Proc. of 12^{th} *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Apr. 2006*

Reliability-Aware Dynamic Energy Management in Dependable Embedded Real-Time Systems

Dakai Zhu
Department of Computer Science
University of Texas at San Antonio
San Antonio, TX, 78249
dzhu@cs.utsa.edu

Abstract

Recent studies show that, voltage scaling, which is an efficient energy management technique, has a direct and negative effect on system reliability because of the increased rate of transient faults (e.g., those induced by cosmic particles). In this work, we propose schemes that explore dynamic slack for energy savings while taking system reliability into consideration. The proposed schemes dynamically schedule an additional recovery to recuperate the reliability loss due to energy management. Based on the amount of available slack, the application size and the fault rate changes, we analyze when it is profitable to reclaim the slack for energy savings without sacrificing system reliability. Checkpoint technique is further explored to efficiently use the slack. Analytical and simulation results show that, the proposed reliability-aware energy management schemes can achieve comparable energy savings as ordinary energy management schemes while preserving system reliability. The ordinary energy management schemes that ignore the effects of voltage scaling on fault rate changes could lead to drastically decreased system reliability.

1 Introduction

The performance of modern computing systems has increased at the expense of dramatically increased power consumption. For battery-operated embedded systems (e.g., PDAs and cell phones), the increased power consumption reduces their operation time. Many hardware and software techniques have been proposed to manage power consumption in modern computing systems and power aware computing has become an important research area recently. As an efficient energy management technique, *voltage scaling*, which reduces system supply voltage for lower operation frequencies [29, 30], has been used extensively in the recently proposed power management schemes [1, 17, 19, 23].

Due to the effects of cosmic ray radiation, transient faults may occur during the execution of an application, especially for systems deployed in vulnerable environments (such as in deep space). Since the critical charge required to maintain proper circuit state is proportional to system supply voltage [24], when system supply voltage is reduced, the critical charge decreases which leads to dramatically increased transient fault rates [34]. Therefore, scaling down voltages and frequencies for energy savings has a severe effect on system reliability [7, 26, 34] and should be carefully evaluated before it is applied, especially for mission critical embedded real-time applications, such as satellite and surveillance systems, where both high level of reliability and low energy consumption are important.

To obtain a certain level of system reliability in the worst case, only static slack in a system has been explored as temporal redundancy traditionally. However, as real-time applications exhibit large variations in actual execution time, and in many cases, only consume a small fraction of their worst case execution time [8], large amount of dynamic slack is available during run-time. As mentioned earlier, simply reclaiming this dynamic slack for energy savings through voltage scaling technique could dramatically reduce system reliability due to increased failure rates as well as extended execution time [7, 34]. Therefore, special considerations are needed when exploiting dynamic slack for energy savings.

In this work, we propose schemes that utilize dynamic slack for energy savings while taking system reliability into consideration. Specifically, the proposed schemes *dynamically* schedule *an additional recovery* using dynamic slack to recuperate the reliability loss due to energy management. To the best of our knowledge, this is the first work that addresses the complications of exploring dynamic slack for both energy and reliability.

The remainder of this paper is organized as follows. The models and problem description are presented in Section 2. Reliability-aware dynamic energy management is proposed and analyzed in Section 3 and Section 4 explores check-

pointing techniques to efficiently use dynamic slack. The simulation results are presented and discussed in Section 5. Section 6 addresses the closely related work and Section 7 concludes the paper.

2 Models and Problem Description

2.1 Power Model

For embedded systems, the power is consumed mainly by the processor, memory, I/O interfaces and underlying circuits. While the power consumption is dominated by dynamic power dissipation, which is quadratically related to supply voltage and linearly related to frequency [3], the static leakage power is ever-increasing and cannot be ignored, especially with increased levels of integration [27]. To incorporate all the power consuming components in an embedded system while keeping the power model simple, we assume that the system has only two states: *sleep* and *active* states. However, different supply voltages/frequencies may be employed in active state to deliver different levels of performance.

Considering the almost linear relation between supply voltage and operating frequency [3], *voltage scaling* reduces the supply voltage for lower frequencies [18]. In this paper, we use frequency changes to stand for changing both supply voltage and frequency and adopt the power model developed in [35]:

$$P = P_s + \hbar(P_{ind} + P_d) \quad (1)$$
$$= P_s + \hbar(P_{ind} + C_{ef} f^m) \quad (2)$$

where P_s is the *sleep power*[1], P_{ind} is the *frequency-independent active power*[2] and P_d is the *frequency-dependent active power*[3]. Considering the large overhead of turning on/off a system [2], we assume the system is always on (in either *sleep* or *active* state) and P_s is not manageable. $\hbar$ equals 0 if the system is in sleep state and $\hbar$ equals 1 otherwise. The effective switching capacitance C_{ef} and the dynamic power exponent m (in general, larger than or equal to 2) are system/application dependent constants [3] and f is the processing frequency. For easy discussion, normalized frequencies are used and the maximum frequency f_{max} is assumed to be 1 (with corresponding normalized supply voltage $V_{max} = 1$). The maximum frequency-dependent active power is denoted by P_d^{max} and we assume $P_s = \alpha P_d^{max}$ and $P_{ind} = \beta P_d^{max}$.

Considering the energy consumption related to the sleep power P_s is fixed for a given time period (e.g., within the deadline D), in what follows, we focus on the energy consumption that comes from active power. Though voltage scaling can reduce energy consumption due to reduced frequency-dependent active power P_d, the computation will take more time and more energy will be consumed due to the effects of frequency-independent active power P_{ind}. Therefore, lower voltages/frequencies may not result in less energy consumption and there exists a minimum energy-efficient voltage/frequency pair [9]. From Equation 1, it is easy to find out that the *energy efficient frequency* is [35]:

$$f_{ee} = \sqrt[m]{\frac{\beta}{m-1}} \quad (3)$$

For energy consideration, we should never run at a frequency below f_{ee}, since doing so consumes more energy. For simplicity, we assume that $f_{ee} \geq f_{low}$, where f_{low} is the lowest frequency in the system, and define the *minimum energy efficient frequency* as $f_{min} = \max\{f_{low}, f_{ee}\} = f_{ee}$. Moreover, frequency is assumed to be able to change continuously[4] from f_{max} to f_{min}.

2.2 Fault Model

During the execution of an application, a fault may occur due to various reasons, such as hardware failures, software errors and the effect of cosmic ray radiations. Since *transient* faults occur much more frequently than *permanent* faults [4, 12, 13], in this paper, we focus on transient faults, especially the ones caused by cosmic ray radiations, and explore temporal redundancy to tolerate them. It is assumed that faults are detected using sanity or consistency checks when a task completes [20].

Traditionally, transient faults have been modeled to follow Poisson distribution with an average arrival rate λ [31]. However, considering the effects of voltage scaling on transient faults [7, 34], the average arrival rate λ will depend on system processing frequency and supply voltage. Therefore, the fault rate at frequency f (and its corresponding voltage level) can be *generally* modeled as

$$\lambda(f) = \lambda_0 g(f) \quad (4)$$

where λ_0 is the average fault rate corresponding to the maximum frequency $f_{max} = 1$ (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* (which is the smallest charge required to cause a soft error in a circuit node) [10]. Moreover, the critical charge is proportional to system supply voltage [24]. When the system supply voltage is reduced, the critical charge decreases and a lower energy particle

[1] It is used to maintain basic circuits, keep the clock running etc.

[2] It consists of the components of memory and processor power that can be efficiently removed by putting systems to sleep and is independent of system supply voltage and frequency [5, 22].

[3] It includes processor dynamic power and any power that depends on system supply voltage and frequency [3, 27].

[4] For discrete frequency levels, we can use two adjacent levels to emulate the execution at any frequency [11].

could strike the sensitive region in a semiconductor device and cause a soft error. Considering the fact that the number of low-energy particles is two magnitude higher than that of the high-energy particles [36], in our analysis and simulations, we focus on the exponential fault rate model proposed in [34] :

$$\lambda(f) = \lambda_0 g(f) = \lambda_0 10^{\frac{d(1-f)}{1-f_{min}}} \quad (5)$$

Here, the maximum average fault rate is assumed to be $\lambda_{max} = \lambda_0 10^d$, which corresponds to the lowest frequency f_{min} (and supply voltage V_{min}), where d (> 0) is a constant. That is, reducing the supply voltage and frequency for energy savings results in exponentially increased fault rates and larger d indicates that the fault rate is more sensitive to voltage scaling. However, the reliability-aware energy management schemes proposed in this paper are very generic and do not rely on this specific fault model.

2.3 Problem Description

In this work, we consider a real-time application that consists of a set of *aperiodic* tasks. The worst case execution time (WCET) of task T_i at the maximum frequency f_{max} is assumed to be c_i and T_i should finish execution before its deadline D_i ($i = 1, \cdots, n$). Due to early completion of tasks, slack will exist during the execution of the applications [8]. **For a given amount of slack S, we focus on the problem of how to use S for energy savings without sacrificing system reliability, while taking the effects of voltage scaling on fault rates into consideration**.

The reliability of a real-time system depends on the correct execution of all tasks in an application. For the application that consists of n tasks, its reliability is $R = \prod_{i=1}^{n} R_i$, where R_i is the probability of task T_i being executed correctly. Without loss of generality, we assume that, when all tasks use their WCETs and are executed at the maximum frequency f_{max}, the reliability of the application, $R_0 = \prod_{i=1}^{n} R_i^0$, is *satisfactory*. Here, $R_i^0 = e^{-\lambda_0 c_i}$ is the probability of task T_i being executed correctly at frequency f_{max} with execution time c_i. In order to preserve the reliability of an application, for simplicity, we focus on maintaining the reliability of individual tasks in this work. That is, we propose schemes to keep the probability of task T_i being correctly executed no less than R_i^0 ($i = 1, \cdots, n$).

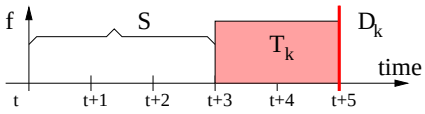
In next Section, we propose a reliability-aware dynamic energy management scheme and analyzes its performance on both reliability and energy consumption for single tasks when the amount of available dynamic slack S is no less than c_k, the size of the next task T_k. When S is smaller than c_k, checkpointing techniques are further explored to efficiently use the available dynamic slack in Section 4.

3 Reliability-Aware Dynamic Energy Management

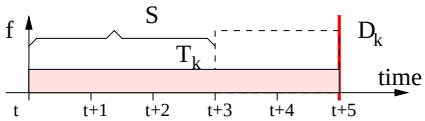
Although sophisticated dynamic power management schemes that explore tasks' statistical information have been proposed [1, 17], we will focus on *greedy* scheme for it's simplicity. Exploring other advanced schemes is beyond the scope of this paper and will be considered in our future work. We first illustrate the problem of ordinary greedy power management on reliability in Section 3.1. Then Section 3.2 presents the new reliability-aware greedy energy management scheme and the analysis.

3.1 Ordinary Greedy Power Management

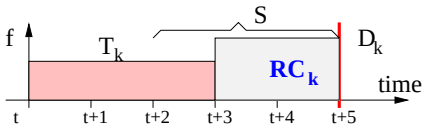
In *ordinary greedy* power management, all the available dynamic slack will be used to scale down the processing of the next task for energy savings [1, 17]. For example, as shown in Figure 1a, due to the early completion of previous tasks, there are 3 units of available dynamic slack at time t, that is, $S = 3$. The WCET of the next ready task T_k is $c_k = 2$. Recall that D_k is the deadline of task T_k.



a. at time t, slack S is available



b. Greedy power management



c. Reliability−aware greedy scheme

Figure 1. Ordinary and Reliability-Aware Greedy Schemes.

Suppose that $\beta = 0.1$ (i.e., $P_{ind} = 0.1 P_d^{max}$) and $m = 3$, we have the minimum energy efficient frequency $f_{ee} = 0.37$ (recall that $f_{max} = 1$, Section 2). Therefore, all the available dynamic slack S can be allocated to task T_k and the processing speed of T_k can be reduced from $f_{max} = 1$ to $f = \frac{2}{2+3} = 0.4$ as shown in Figure 1b. From Equation 1, it is easy to find that scaling down the processing of T_k could save 63% of the *active energy*[5].

However, as discussed in Section 2, with reduced processing frequency and supply voltage, the processing of task

[5]Notice that the sleep power P_s is not manageable and we focus on active energy consumption in this work.

T_k is more susceptible to transient faults [7, 34]. Suppose that the exponent in the fault rate model is $d = 2$ (see Equation 5 in Section 2), the probability of having fault(s) during the execution of task T_k at the reduced speed will be:

$$\begin{aligned} & \rho_k = 1 - R_k = 1 - e^{-\lambda_0 10^{\frac{d(1-f)}{1-f_{min}}}(S+c_k)} \\ = \;& 1 - e^{-\lambda_0 10^{\frac{d(1-f)}{1-f_{min}}}\frac{c_k}{f}} = 1 - e^{-\lambda_0 c_k 10^{\frac{2(1-0.4)}{1-0.37}}\frac{1}{0.4}} \\ \approx \;& 1 - (R_k^0)^{200} = 1 - (1-\rho_k^0)^{200} \approx 200\rho_k^0 \end{aligned} \quad (6)$$

where ρ_k^0 is the probability[6] of having fault(s) when task T_k uses its WCET at the maximum processing frequency f_{max}. That is, though 63% active energy is saved by scaling down the processing of task T_k, it leads to approximately 200 times higher in the probability of failure! The increase in the probability of failure during the processing of individual tasks will degrade the overall system reliability, which is unbearable, especially for mission-critical systems where the requirement for high levels of reliability is strict.

3.2 Reliability-Aware Greedy Scheme

In order to recuperate the reliability loss due to energy management, we propose the *reliability-aware greedy (RA-Greedy)* power management scheme, which dynamically schedules a *recovery* for the task to be scaled by energy management. Here, the recovery task takes the form of simple re-execution (thus has the same size of the task to be recovered) and will be executed (if needed) at the maximum frequency $f_{max} = 1$.

Notice that, in this section, the amount of dynamic slack S is assumed to be no less than c_k, the size of next task T_k. After reserving c_k units of dynamic slack for the recovery task, the remaining dynamic slack ($S - c_k$, if any) can be allocated to T_k for energy savings. Therefore, the execution of task T_k will have $c_k + (S - c_k) = S$ units of time and be processed at a reduced frequency $f_k = \frac{c_k}{S}$. For example, as shown in Figure 1c, a recovery task RC_k is scheduled for task T_k which uses 2 units of dynamic slack. The remaining 1 unit of dynamic slack allows task T_k to run at a lower frequency $f = \frac{2}{2+1} = 0.66$ and save energy.

3.2.1 System Reliability under RA-Greedy

With the additional recovery task RC_k, the reliability R_k of task T_k will be the summation of the probability of primary task T_k being executed correctly and the probability of having fault(s) during T_k's execution while RC_k being executed correctly. Notice that, if the execution of the primary task T_k is faulty, the recovery task RC_k will be executed at the maximum frequency f_{max} and the probability of its *fault-free* execution is $e^{-\lambda_0 c_k} = R_k^0$. Therefore, we have:

$$R_k = e^{-\lambda(f_k)S} + \left(1 - e^{-\lambda(f_k)S}\right) R_k^0 > R_k^0 \quad (7)$$

where $\lambda(f_k)$ is the fault rate at the reduced frequency f_k. From the above equation, we can see that, under the RA-Greedy scheme, with the help of the additional recovery task RC_k, the reliability of task T_k is always better than R_k^0 regardless different fault rate increases (i.e., different values of d in Equation 5) and the reduced processing frequency f_k of the primary task T_k. That is, when the amount of dynamic slack is no less than the size of the next task, by dynamically scheduling a recovery task before applying energy management schemes, the RA-Greedy scheme can achieve better reliability for individual tasks, and thus preserve system reliability.

3.2.2 Expected Energy Consumption

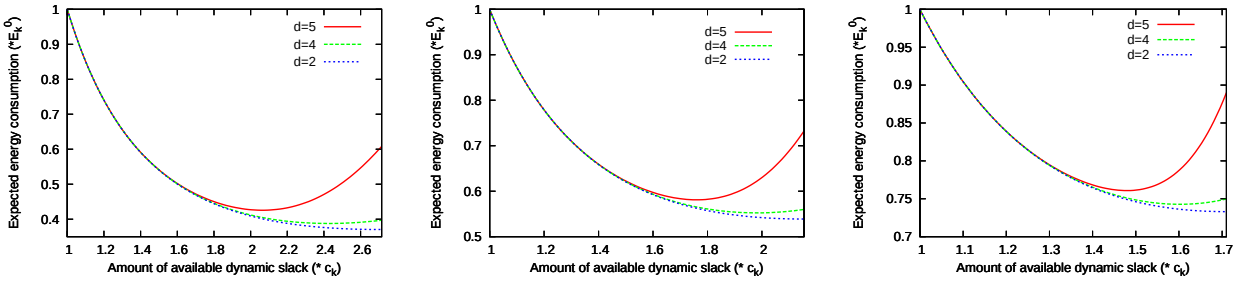
Suppose that the energy consumption to execute task T_k for time c_k at the maximum frequency f_{max} is $E_k^0 = (P_s + P_{ind} + P_d^{max})c_k = (\alpha + \beta + 1)P_d^{max}c_k$. Considering the probability of RC_k being executed, the *expected energy consumption* for processing task T_k will be:

$$\begin{aligned} E_k &= (P_s + P_{ind} + C_{ef} f_k^m)S + (1 - e^{-\lambda(f_k)S})E_k^0 \\ &= E_k^0 \left[1 - e^{-\lambda(f_k)S} + \frac{(\alpha + \beta + \frac{c_k^m}{S^m})\frac{S}{c_k}}{1 + \alpha + \beta}\right] \end{aligned} \quad (8)$$

Intuitively, the more the available dynamic slack is allocated for energy management, the lower the processing frequency can be for executing task T_k, and thus more energy savings can be obtained. However, due to the limitation of the minimum energy efficient frequency f_{ee}, the maximum amount of dynamic slack that should be allocated to task T_k for energy management is limited, which can be easily calculated as $\frac{c_k}{f_{ee}} - c_k = US_{max} - c_k$, where US_{max} ($= \frac{c_k}{f_{ee}}$) is the maximum amount of dynamic slack that may be used when processing T_k. When more dynamic slack than US_{max} is available, part of the slack will be saved for future tasks due to energy consideration.

Moreover, with reduced processing frequency and supply voltage, the fault rate increases and the execution of T_k takes more time, which results in higher probability of having fault(s) during the execution of T_k. Therefore the probability of recovery task RC_k being executed increases, which may overshadow the energy savings and lead to more expected energy consumption. However, considering the exponential component in Equation 8, it is hard to obtain a simple closed formula for the optimal amount of dynamic slack that minimizes the expected energy consumption. In what follows, we present some analytical results to illustrate the relation between the expected energy consump-

[6]Note that, ρ_k^0 is a small number (usually $< 10^{-4}$).



a. $\beta = 0.1$ b. $\beta = 0.2$ b. $\beta = 0.4$

Figure 2. The normalized expected energy consumption vs. the amount of available dynamic slack.

tion, the amount of available dynamic slack and the fault rate changes due to energy management.

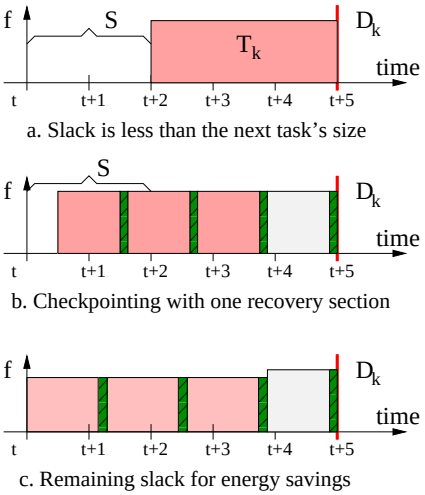
Without loss of generality, in the analysis, we assume $c_k = 1$ and $\lambda_0 = 10^{-6}$ (which corresponds to 100,000 FITs, *failure in time* in terms of errors per billion hours of use per megabit, that is a reasonable fault rate as reported [25, 37]). Moreover, we assume $\alpha = 0$ (i.e., $P_s = 0$) and $m = 3$. Figure 2 shows the expected energy consumption for executing task T_k, normalized to E_k^0, versus the amount of available dynamic slack under different frequency-independent power (i.e., β) and fault rate changes (d). Notice that, one unit (c_k) of dynamic slack is reserved for the recovery task. From Section 2, for different frequency-independent power $\beta = 0.1, 0.2$ and 0.4, the corresponding energy efficient frequency are $f_{ee} = 0.37, 0.46$ and 0.58, which in turn limits the maximum amount of dynamic slack used by RA-Greedy scheme US_{max} to be $2.70c_k$, $2.17c_k$ and $1.72c_k$, respectively.

From the figures we can see that, when the amount of available dynamic slack is more than c_k ($= 1$), the size of the next task T_k, dynamic slack is available for energy management and the expected energy consumption to execute T_k is less than E_k^0 (i.e., energy savings is expected). As the amount of available dynamic slack increases, more slack is available for energy management and the expected energy consumption for executing task T_k generally decreases. However, when the fault rate increases dramatically with reduced supply voltages (e.g., $d = 5$), as more dynamic slack is available and the reduced frequency approaches f_{ee}, more expected energy may be consumed due to the increased probability of recovery task being executed. In this case, the optimal amount of dynamic slack to minimize expected energy consumption is less than US_{max}.

Notice that, when the fault rate change is not that severe (e.g., $d \leq 4$), the maximum amount of dynamic slack US_{max} limited by f_{ee} is very close to the optimal amount of slack that minimizes the expected energy consumption. Considering the difficulty of finding the close formula for the optimal amount of slack, for simplicity, in this work, the amount of dynamic slack that will be allocated for energy management is only limited by f_{ee} (i.e., up to US_{max} amount of dynamic slack will be used). Moreover, for higher frequency-dependent active power (i.e., $\beta = 0.4$), f_{ee} increases and US_{max} decreases, which results in less energy savings (note the difference in the scale of Y-axis of Figure 2).

Instead of scheduling the whole recovery task, checkpoints may be employed to efficiently use dynamic slack for more energy savings [14, 16], which is especially useful for the case where the amount of available dynamic slack S is less than c_k, the size of next ready task T_k.

4 Checkpoints with Less Dynamic Slack



a. Slack is less than the next task's size

b. Checkpointing with one recovery section

c. Remaining slack for energy savings

Figure 3. Reliability-Aware Energy Management with Checkpoints.

Checkpointing techniques insert checkpoints during the execution of an application. Within a checkpoint, the state of a system is checked and correct states are saved to a stable storage [20]. When faults are detected, the execution is rolled back to the latest correct checkpoint by exploring the temporal redundancy [14, 15].

For example, Figure 3a shows that there are 2 units of dynamic slack available at time t, which is less than $c_k = 3$, the size of the next ready task T_k. If the overhead of employing one checkpoint is $r = 0.125$ and 3 checkpoints are inserted, Figure 3b illustrates the case of one recovery section being scheduled. Here there is 0.5 units of remaining dynamic slack, which can be used to scale down the processing of the primary task sections for energy savings as shown in Figure 3c.

In this work, for a given checkpoint overhead r, we focus on the problem of **finding the minimum amount of dynamic slack needed for energy savings while preserving system reliability**. For easy discussion, we assume that $r = \gamma \cdot c_k$. Suppose that n checkpoints are inserted during the execution of task T_k. The size of the recovery section will be $\frac{c_k}{n}$ and we have:

$$S \geq n \cdot r + (r + \frac{c_k}{n}) = (n\gamma + \gamma + \frac{1}{n})c_k \tag{9}$$

In order for n to have a real (non-imaginary) solution, we can easily find that the minimum amount of slack needed due to timing constraints is $S^{time}_{min} = (\gamma + 2\sqrt{\gamma})c_k$ and the optimal number of checkpoints is $n_{opt} = \left\lfloor \sqrt{\frac{1}{\gamma}} \right\rfloor$ or $n_{opt} = \left\lceil \sqrt{\frac{1}{\gamma}} \right\rceil$. However, considering the integer property of n_{opt} and the energy overhead incurred by checkpoints, the minimum amount of slack needed for energy savings S^{energy}_{min} should be larger than S^{time}_{min} as illustrated in Section 4.2.

Notice that the amount of dynamic slack considered in this section is less than c_k, the size of next task T_k. Therefore, to employ checkpoints for energy management, we need to have $S^{time}_{min} = (\gamma + 2\sqrt{\gamma})c_k < c_k$. That is, for the case we considered in this Section, the checkpoint overhead needs to satisfy $\gamma < \gamma_{max} = 0.17$ due to timing constraints.

4.1 Reliability with Checkpoints

With the optimal number of checkpoints n_{opt} and one recovery section, the amount of available slack for energy management will be $S - (n_{opt} + 1)r - \frac{c_k}{n_{opt}}$, which can be used to scale down the execution of the primary sections. Therefore, the reduced frequency to execute the primary sections will be

$$f_{ckpt} = \frac{c_k + n_{opt} \cdot r}{S + c_k - r - \frac{c_k}{n_{opt}}} \tag{10}$$

and each primary section will take $t_{section} = \frac{S + c_k - r - \frac{c_k}{n_{opt}}}{n_{opt}}$ time units. From Section 2, the fault rate at frequency f_{ckpt} will be $\lambda(f_{ckpt}) = \lambda_0 10^{\frac{d(1-f_{ckpt})}{1-f_{min}}}$ and the probability of having fault(s) during the execution of one primary section is $\rho_{section} = 1 - e^{-\lambda(f_{ckpt})t_{section}}$. Notice that, the recovery section is executed at f_{max} and the probability of having fault(s) during the execution of the recovery section is $\rho_{recovery} = 1 - e^{-\lambda_0(r + \frac{c_k}{n_{opt}})}$. Therefore, the reliability of executing task T_k is

$$R^{ckpt}_k = (1 - \rho_{section})^{n_{opt}} + n_{opt} \cdot \rho_{section}(1 - \rho_{section})^{n_{opt}-1}(1 - \rho_{recovery}) \tag{11}$$

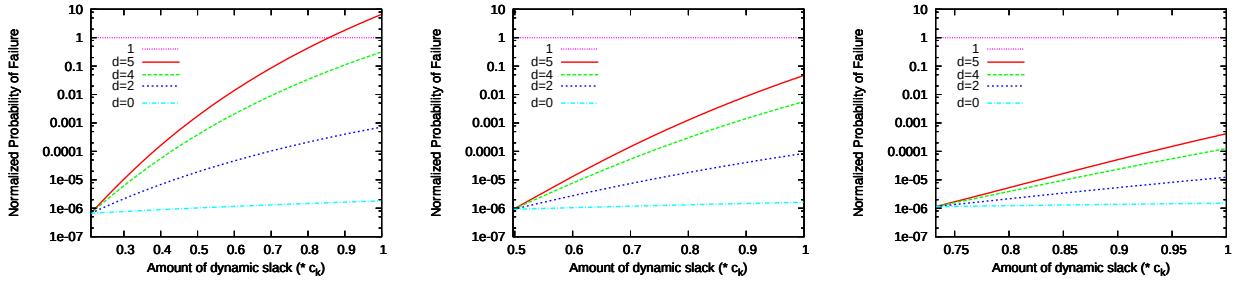
where the first part is the probability of all primary sections being executed correctly and the second part is the probability of having fault(s) during the execution of one primary section while the recovery section being executed correctly.

From Equation 11, R^{ckpt}_k is determined by the amount of available dynamic slack S, checkpoint overhead r and fault rate changes d. For a given checkpoint overhead, more dynamic slack leads to lower reduced frequency for the primary sections, which in turn leads to higher probability of failure and lower reliability R^{ckpt}_k. However, due to the complexity of Equation 11, it is hard to find the close formula for S to ensure $R^{ckpt}_k \geq R^0_k$.

Figure 4 shows the normalized probability of failure, $\frac{1-R^{ckpt}_k}{1-R^0_k}$, when executing task T_k with different amount of available dynamic slack under different checkpoint overheads. The same as before, we assume $m = 3$, $\lambda_0 = 10^{-6}$ and $c_k = 1$. Moreover, β is assumed to be 0.1 and we have $f_{ee} = 0.37$. Thus, the maximum amount of dynamic slack limited by f_{ee} for energy management is larger than $c_k = 1 > S$. Therefore, for given checkpoint overhead $r = \gamma c_k$, the amount of dynamic slack considered will be in the range of S^{time}_{min} $(= \gamma + 2\sqrt{\gamma})$ and 1.

From the figure, we can see that, with one recovery section, the normalized probability of failure to execute task T_k is lower than 1 most of the time, which means that higher reliability than R^0_k is achieved. The exception comes from the case where the checkpoint overhead is low (i.e., $\gamma = 0.01$; see Figure 4a) which leaves more slack for energy management and the reduced frequency is close to f_{ee} when the amount of dynamic slack is around 1. With the exponent of fault rate model being $d = 5$, the fault rate at f_{ee} is 10^5 time higher than $\lambda_0 = 10^{-6}$ and leads to worse than R^0_k reliability. However, with moderate fault rate increase (e.g., $d \leq 4$), exploring checkpoints with one recovery section before applying energy management always obtains higher reliability for executing task T_k.

Moreover, the faster the fault rate increases (i.e., larger d) with reduced frequencies and supply voltages, the higher the probability of failure and the lower the reliability. Assuming constant fault rate (e.g., $d = 0$) is too optimistic and could lead to lower reliability than expected when exploring slack for energy management, which is the same observation as our previous results [34].



a. $\gamma = 0.01$ b. $\gamma = 0.05$ c. $\gamma = 0.10$

Figure 4. The normalized probability of failure vs. the amount of available dynamic slack.

4.2 Expected Energy Consumption with Checkpoints

With reduced frequency f_{ckpt}, the energy consumption for executing each primary section is $E_{section} = \left(\alpha + \beta + \left(\frac{f_{ckpt}}{f_{max}}\right)^m\right) P_d^{max} t_{section}$. The energy consumption for executing the recovery section is $E_{recovery} = (\gamma + \frac{1}{n_{opt}})E_k^0$. Considering the probability of recovery section being executed, the expected energy consumption for executing task T_k will be

$$E_k^{ckpt} = n_{opt} E_{section} + (1 - (1 - \rho_{section})^{n_{opt}}) E_{recovery} \quad (12)$$

where the first part is always consumed and is the energy for executing the primary sections (including the checkpoints), and the second part is the expected energy consumption for executing the recovery section.

Due to the overhead of checkpoints, in order to obtain energy savings (i.e., $E_k^{ckpt} < E_k^0$), there is a minimum amount of dynamic slack S_{min}^{energy} needed for energy management. Again, due to the complexity of Equation 12, it is hard to get the close formula for S_{min}^{energy} and we illustrate the relation between S_{min}^{energy} and checkpoint overhead γ in the following analysis.

Figure 5 shows the normalized expected energy consumption, $\frac{E_k^{ckpt}}{E_k^0}$, when executing task T_k with different amount of available dynamic slack under different checkpoint overheads. The same parameters as in Section 4.1 are used here. For different fault rate changes (i.e., different values of d), due to the low probability of recovery section being executed (lower than 10^{-5} even when $d = 5$), the expected energy consumption is almost the same for a given checkpoint overhead and amount of available dynamic slack. Therefore, we only show the normalized expected energy consumption for the worst case of $d = 5$.

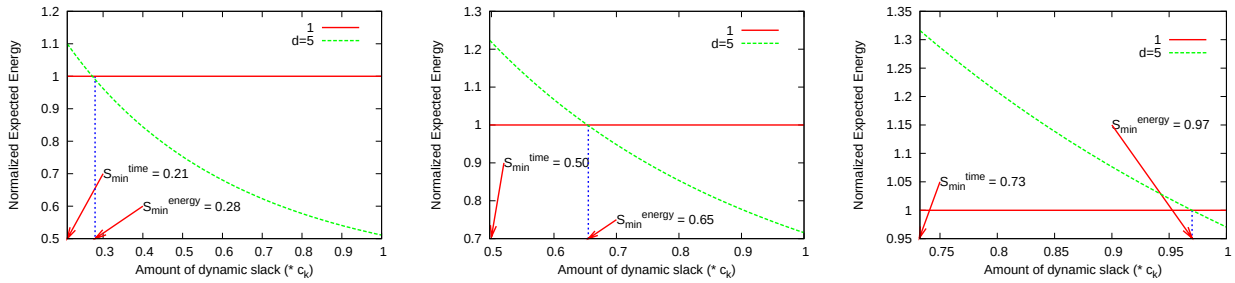
From the figures, we can see that, though it is feasible to employ checkpoints when the amount of dynamic slack is larger than S_{min}^{time}, due to the energy overhead of checkpoints, no energy savings could be obtained until the amount of slack is more than S_{min}^{energy}. The smaller the checkpoint overhead, the lower the value of S_{min}^{energy} and the more energy savings could be obtained for a given amount of dynamic slack. When the checkpoint overhead is large (e.g., $\gamma = 0.1$, which is close to $\gamma_{max} = 0.17$; Figure 5c), almost no energy savings could be obtained and checkpoints should not be employed.

Considering both reliability (Section 4.1) and energy savings (Section 4.2), checkpoints should not be employed for energy management when checkpoint overhead is relatively large (e.g., $\gamma > 0.1$). Moreover, when checkpoint overhead is relatively small (e.g., $\gamma = 0.01$), though more energy savings could be obtained with more available dynamic slack, limitation may exist on the amount of employed dynamic slack due to reliability consideration, especially for dramatical fault rate increases with reduced frequencies and supply voltages (e.g., $d = 5$).

We have analyzed the performance of the proposed schemes for a single task. In what follows, to illustrate the merits of our proposed reliability-aware energy management schemes and see how they performs for overall system reliability and energy savings, we present simulation results for dependable real-time applications that consist of a set of aperiodic tasks. We compare the energy savings as well as system reliability of the new proposed schemes with ordinary energy management schemes.

5 Simulation Results and Discussion

In the simulations, we consider four different schemes: a) *no power management (NPM)*, which is used as the baseline for comparison; b) *ordinary greedy power management (Greedy)*, which allocates all available dynamic slack for next ready task to save energy without considering system reliability; c) *reliability-aware greedy power management (RA-Greedy)*, which dynamically allocates a recovery task for next ready task before applying greedy power management. When the amount of available dynamic slack is less than the size of next ready task, the slack is not used and saved for future tasks; d) *reliability-aware power management with checkpoints (Ckpt)*, which is the same as RA-Greedy except that checkpoints are employed when the



a. $\gamma = 0.01$ b. $\gamma = 0.05$ c. $\gamma = 0.10$

Figure 5. The normalized expected energy consumption vs. the amount of available dynamic slack.

amount of available dynamic slack is less than the size of next ready task.

For the system parameters, as discussed in Section 2, we use normalized frequency with $f_{max} = 1$ and assume frequency can be changed continuously. Moreover, corresponding to the analysis in Section 3 and 4, we assume $\alpha = 0$, $\beta = 0.1$ and $m = 3$. That is, we focus on active power. For the effects of different values of β on energy management, see [35] for more discussions. The same as in Section 3, we assume that faults follow a Poisson distribution with an average fault rate as $\lambda_0 = 10^{-6}$ at f_{max} (and corresponding V_{max}). We vary the values of d (as $0, 2$ and 5 respectively) for different changes in fault rates due to the effects of frequency and voltage scaling [7]. An application fails if *any* task in the application fails and there is no recovery or both the task and its recovery fail.

The number of tasks in an application is randomly generated between 5 and 20, where the WCETs of tasks are uniformly distributed in the range of 1 and 10. When every task in an application uses its WCET, we assume that the application finishes just in time and the system reliability is satisfactory. To emulate the run-time behaviors of tasks, a parameter σ is used as an application-wide average over worst execution time, which also indicates the amount of dynamic slack available on average during execution. Smaller values of σ imply more dynamic slack. The value of σ_i for task T_i in the application is generated from a uniform distribution with an average value of σ. The actual execution time of T_i follows a similar uniform distribution with an average value of $\sigma_i \cdot c_i$, where c_i is the WCET of task T_i. For each result point in the graphs, 100 task sets are generated and each task set is executed $100,000$ times, and the result is the average of all the runs.

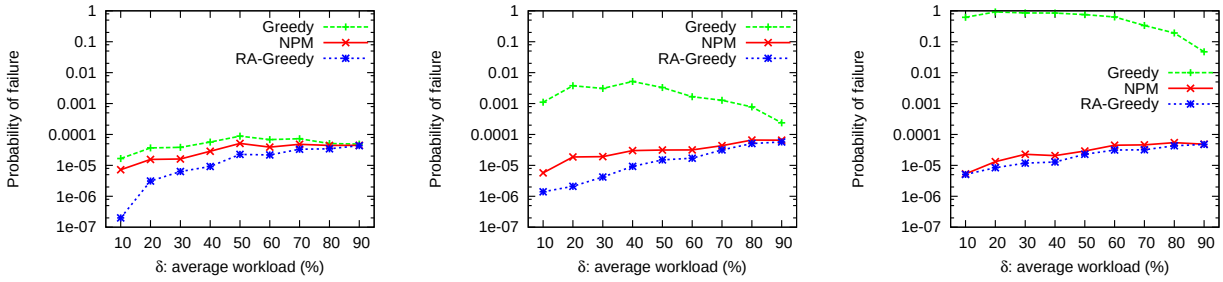
5.1 Performance of RA-Greedy

First, we compare the performance of *Greedy* and *RA-Greedy* on reliability and energy consumption. For different fault rate changes, Figure 6 shows the probability of failure when executing the applications with different average system loads (i.e., different amounts of dynamic slack).

Notice that, the fault rate under *NPM* is always $\lambda_0 = 10^{-6}$, which is not affected by the different fault rate changes (i.e., different values of d). Therefore, from Figure 6, we can see that for a given average system load, the probability of failure under *NPM* is roughly the same. When the average system load increases, the applications run longer and the probability of failure under *NPM* increases linearly. Note the log scale of Y-axis in Figure 6.

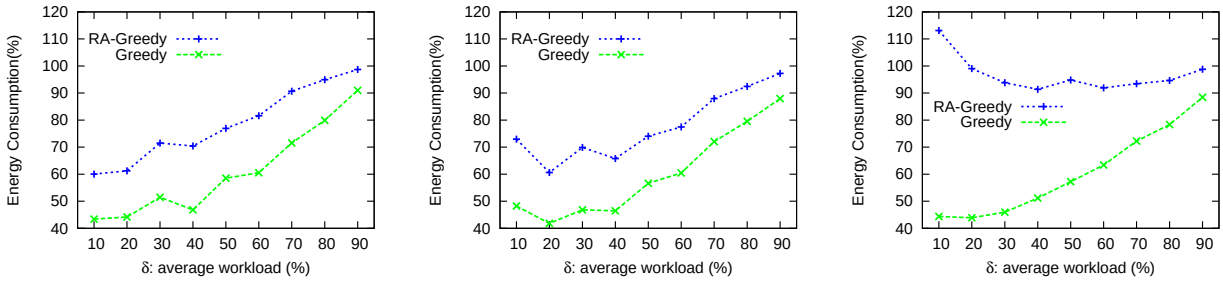
From the figure, it can be seen that the *Greedy* scheme results in higher probability of failure than *NPM* even when $d = 0$ (i.e., constant fault rate), which comes from the extended execution of tasks due to energy management. When fault rate increases with reduced frequencies and supply voltages (i.e., $d > 0$), the probability of failure under *Greedy* scheme increases exponentially as d increases. For example, when $d = 5$, *Greedy* scheme almost always leads to system failure (with probability of failure close to 1), especially for the case of low average system loads where more dynamic slack exists. When the average system load increases, the probability of failure under *Greedy* scheme increases first and then decreases, the reason is because of the limitation of $f_{ee} = 0.37$. When the average system load is extremely low (e.g., $\sigma < 20\%$), tasks in an application always run at f_{ee} and the probability of failure mainly depends on the execution time, which increases as average system load increases. However, as average system load continues to increase, less slack is available and tasks need to run at higher frequencies than f_{ee}, which has lower fault rates and thus leads to lower probability of failure. Moreover, from Figure 6, we can also see that *RA-Greedy* scheme always has a lower probability of failure (i.e., higher system reliability) than *NPM* regardless the fault rate changes, which coincides with the analysis in Section 3.2.1.

Figure 7 shows the corresponding normalized energy consumption for *Greedy* and *RA-Greedy* schemes with the one consumed by *NPM* as a baseline. As *Greedy* scheme does not consider system reliability when reclaiming dynamic slack for energy savings, the normalized energy consumption for *Greedy* scheme only depends on the average system load and is roughly the same for different fault rate



a. $d = 0$ b. $d = 2$ c. $d = 5$

Figure 6. The probability of failure vs. different average system loads.



a. $d = 0$ b. $d = 2$ c. $d = 5$

Figure 7. The normalized expected energy consumption vs. average system loads.

changes. For *RA-Greedy* scheme, by providing an additional recovery for maintaining system reliability, it consumes from 10% to 20% more energy than *Greedy* scheme when the fault rate only increases moderately with reduced frequencies and supply voltages (i.e., $d \leq 2$). However, when the fault rate increases dramatically (e.g., $d = 5$), the probability of failure for the original scaled-down execution is close to 1 when the average system load is low (see Figure 6c) and the recovery task is almost always executed, which leads to higher energy consumption than *NPM* (Figure 7c). Therefore, when the fault rate increases dramatically with reduced frequencies and supply voltages, it will be more energy efficient to use less dynamic slack for energy management to keep the fault rate at a reasonable level.

5.2 Effects of Checkpoints

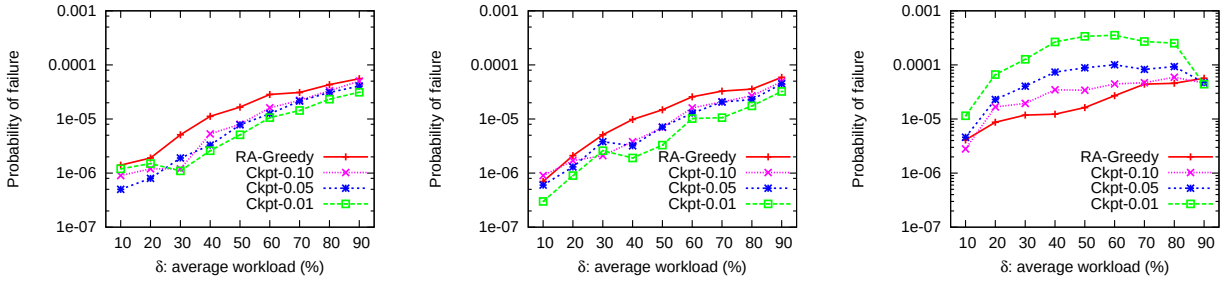
Considering the checkpoint overhead could be very small [21], we use $r = 0.01, 0.05, 0.1$, which corresponds to *Ckpt-0.01*, *Ckpt-0.05* and *Ckpt-0.10* in the following figures, respectively. Recall that the size of tasks is in the range of $[1, 10]$ inclusively, which leads to the average $\gamma = 0.002, 0.01$ and 0.02, smaller than the ones we used in the analysis in Section 4.

Figure 8 shows the probability of failure for the schemes of *RA-Greedy* and *Ckpt* with different checkpoint overheads. From the figure, when the fault rate increase is moderate (i.e., $d \leq 2$), *Ckpt* achieves slightly better system reliability (lower probability of failure) by providing an additional recovery when the amount of available dynamic slack is less than the size of the next ready task. When the checkpoint overhead is smaller, *Ckpt* has more chances to use the dynamic slack and generally gets better system reliability. However, when the fault rate increase is high (e.g., $d = 5$), the additional recovery is almost always executed and overall probability of failure increases due to the execution overhead of checkpoints. Smaller checkpoint overhead leads to higher probability of using checkpoints and thus results in higher probability of failure.

Figure 9 further shows the corresponding normalized energy consumption for *RA-Greedy* and *Ckpt* with different checkpoint overheads. With additional chances for energy management, *Ckpt* with smaller checkpoint overhead consumes less energy, and all of them is less than the one consumed by *RA-Greedy*. The same reason as before, due to the limitation of f_{ee} and higher failure rates, the normalized energy consumption decreases first and then increases as the average system load increases. All schemes consumes more energy than *NPM* when $d = 5$ and $\sigma \leq 10\%$.

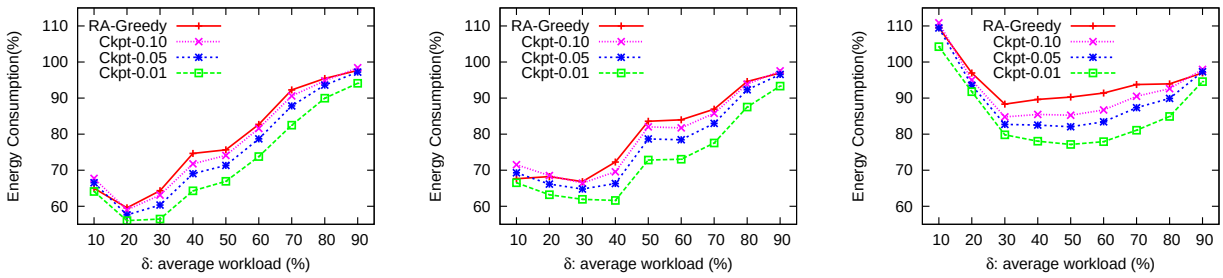
6 Closely Related Work

Using the primary/backup recovery model, Unsal *et al.* proposed to postpone the execution of backup tasks to minimize the overlap of primary and backup execution and thus the energy consumption [28]. The optimal number of checkpoints, evenly or unevenly distributed, to achieve



a. $d = 0$ b. $d = 2$ c. $d = 5$

Figure 8. The probability of failure with checkpoints.



a. $d = 0$ b. $d = 2$ c. $d = 5$

Figure 9. The normalized expected energy consumption with checkpoints.

minimal energy consumption while tolerating one transient fault was explored by Melhem *et al.* in [16]. Elnozahy *et al.* proposed an *Optimistic TMR* scheme that reduces the energy consumption for traditional TMR systems by allowing one processing unit to slow down provided that it can catch up and finish the computation before the application deadline [6]. The optimal frequency settings for OTMR was further explored in [35]. Assuming a Poisson fault model, Zhang *et al.* proposed an adaptive checkpointing scheme that dynamically adjusts checkpoint intervals for energy savings while tolerating a fixed number of faults for a single task [31]. The work is further extended to a set of periodic tasks [33], and moreover, faults within checkpoints are also considered [32].

Most of the previous research either focused on tolerating fixed number of faults [6, 16] or assumed constant fault rate [31, 32, 35] when applying frequency and voltage scaling for energy savings. The work reported in this paper is different from all previous work in that we address the system reliability problem when exploring dynamic slack for energy savings, while explicitly taking the effects of energy management on fault rates into consideration.

7 Conclusions

As fault rates generally increase with reduced supply voltages, energy management exploring slack time through voltage scaling will reduce system reliability, which is undesirable, especially for mission critical applications (e.g., satellite and surveillance systems), where system reliability is as important as (or even more important than) energy consumption. Considering the effects of voltage scaling on fault rates, we propose *reliability-aware* dynamic energy management schemes that preserve system reliability while exploring dynamic slack for energy savings.

By scheduling an additional recovery task before reclaiming dynamic slack for energy management, the proposed reliability-aware energy management scheme ensures that the system reliability achieved is higher than the case when there is no power management. Checkpointing techniques are further explored to more efficiently use the dynamic slack when the slack is not enough to schedule a recovery for a whole task. The performance of the proposed schemes is analyzed and evaluated through simulations for both system reliability and energy savings. The results show that, the proposed schemes can achieve comparable energy savings as ordinary energy management schemes while preserving system reliability. Ignoring the effects of energy management on fault rates is too optimistic and the ordinary energy management schemes could lead to drastically decreased system reliability.

Acknowledgements

The author would like to thank Prof. Rami Melhem and Prof. Daniel Mossé for their inspiring discussions and suggestions in the early stages of this work. Also, the author would like to thank the anonymous reviewers whose com-

ments helped to improve the paper.

References

[1] H. Aydin, R. Melhem, D. Mossé, and P. Mejia-Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proc. of The* 22^{th} *IEEE Real-Time Systems Symposium*, Dec. 2001.

[2] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. *The case for power management in web servers*, chapter 1. Power Aware Computing. Plenum/Kluwer Publishers, 2002.

[3] T. D. Burd and R. W. Brodersen. Energy efficient cmos microprocessor design. In *Proc. of The HICSS Conference*, Jan. 1995.

[4] X. Castillo, S. McConnel, and D. Siewiorek. Derivation and caliberation of a transient error reliability model. *IEEE Trans. on computers*, 31(7):658–671, 1982.

[5] Intel Corp. Mobile pentium iii processor-m datasheet. Order Number: 298340-002, Oct 2001.

[6] E. (Mootaz) Elnozahy, R. Melhem, and D. Mossé. Energy-efficient duplex and tmr real-time systems. In *Proc. of The* 23^{rd} *IEEE Real-Time Systems Symposium*, Dec. 2002.

[7] 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*, 24(6):10–20, 2004.

[8] R. Ernst and W. Ye. Embedded program timing analysis based on path clustering and architecture classification. In *Proc. of The International Conference on Computer-Aided Design*, pages 598–604, Nov. 1997.

[9] X. Fan, C. Ellis, and A. Lebeck. The synergy between power-aware memory systems and processor voltage. In *Proc. of the Workshop on Power-Aware Computing Systems*, 2003.

[10] P. Hazucha and C. Svensson. Impact of cmos technology scaling on the atmospheric neutron soft error rate. *IEEE Trans. on Nuclear Science*, 47(6):2586–2594, 2000.

[11] T. Ishihara and H. Yauura. Voltage scheduling problem for dynamically variable voltage processors. In *Proc. of The 1998 International Symposium on Low Power Electronics and Design*, Aug. 1998.

[12] R.K. Iyer and D. J. Rossetti. A measurement-based model for workload dependence of cpu errors. *IEEE Trans. on Computers*, 33:518–528, 1984.

[13] R.K. Iyer, D. J. Rossetti, and M.C. Hsueh. Measurement and modeling of computer reliability as affected by system activity. *ACM Trans. on Computer Systems*, 4(3):214–237, Aug. 1986.

[14] C. M. Krishma and A. D. Singh. Reliability of checkpointed real-time systems using time redundancy. *IEEE Trans. on Reliability*, 42(3):427–435, 1993.

[15] H. Lee, H. Shin, and S. Min. Worst case timing requirement of real-time tasks with time redundancy. In *Proc. of Real-Time Computing Systems and Applications*, 1999.

[16] R. Melhem, D. Mossé, and E. (Mootaz) Elnozahy. The interplay of power management and fault recovery in real-time systems. *IEEE Trans. on Computers*, 53(2):217–231, 2004.

[17] D. Mossé, H. Aydin, B. R. Childers, and R. Melhem. Compiler-assisted dynamic power-aware scheduling for real-time applications. In *Proc. of Workshop on Compiler and OS for Low Power*, Oct. 2000.

[18] T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *Proc. of Int'l Symposium on Low Power Electronics and Design*, Aug. 1998.

[19] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proc. of* 18^{th} *ACM Symposium on Operating Systems Principles (SOSP'01)*, Oct. 2001.

[20] D. K. Pradhan. *Fault Tolerance Computing: Theory and Techniques*. Prentice Hall, 1986.

[21] F. Quaglia and A. Santoro. Nonblocking checkpointing for optimistic parallel simulation: Description and an implementation. *IEEE Trans. on Parallel and Distributed Systems*, 14(6):593–610, 2003.

[22] Rambus. Rdram. http://www.rambus.com/, 1999.

[23] S. Saewong and R. Rajkumar. Practical voltage scaling for fixed-priority rt-systems. In *Proc. of the* 9^{th} *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2003.

[24] N. Seifert, D. Moyer, N. Leland, and R. Hokinson. Historical trend in alpha-particle induced soft error rates of the alpha^{TM} microprocessor. In *Proc. of the* 39^{th} *Annual International Reliability Physics Symposium*, 2001.

[25] Tezzaron Semiconductor. Soft errors in electronic memory: A white paper. available at http://www.tachyonsemi.com/about/papers/, 2004.

[26] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proc. of the International Conference on Dependable Systems and Networks*, 2002.

[27] A. Sinha and A. P. Chandrakasan. Jouletrack - a web based tool for software energy profiling. In *Proc. of Design Automation Conference*, Jun 2001.

[28] O. S. Unsal, I. Koren, and C. M. Krishna. Towards energy-aware software-based fault tolerance in real-time systems. In *Proc. of The International Symposium on Low Power Electronics Design (ISLPED)*, Aug. 2002.

[29] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced cpu energy. In *Proc. of The First USENIX Symposium on Operating Systems Design and Implementation*, Nov. 1994.

[30] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proc. of The* 36^{th} *Annual Symposium on Foundations of Computer Science*, Oct. 1995.

[31] Y. Zhang and K. Chakrabarty. Energy-aware adaptive checkpointing in embedded real-time systems. In *Proc. of IEEE/ACM Design, Automation and Test in Europe Conference(DATE)*, 2003.

[32] Y. Zhang and K. Chakrabarty. Task feasibility analysis and dynamic voltage scaling in fault-tolerant real-time embedded systems. In *Proc. of IEEE/ACM Design, Automation and Test in Europe Conference(DATE)*, 2004.

[33] Y. Zhang, K. Chakrabarty, and V. Swaminathan. Energy-aware fault tolerance in fixed-priority real-time embedded systems. In *Proc. of International Conference on Computer Aided Design*, Nov. 2003.

[34] D. Zhu, R. Melhem, and D. Mossé. The effects of energy management on reliability in real-time embedded systems. In *Proc. of the International Conference on Computer Aidded Design (ICCAD)*, Nov. 2004.

[35] D. Zhu, R. Melhem, D. Mossé, and E.(Mootaz) Elnozahy. Analysis of an energy efficient optimistic tmr scheme. In *Proc. of the* 10^{th} *International Conference on Parallel and Distributed Systems (ICPADS)*, Jul. 2004.

[36] J. F. Ziegler. Terrestrial cosmic ray intensities. *IBM Journal of Research and Development*, 42(1):117–139, 1998.

[37] J. F. Ziegler. Trends in electronic reliability: Effects of terrestrial cosmic rays. available at http://www.srim.org/SER/SERTrends.htm, 2004.