Dual-Processor Design of Energy Efficient Fault-Tolerant System*

Shaoxiong Hua
Synopsys Inc.
700 E. Middlefield Road
Mountain View, CA 94043
huas@synopsys.com

Pushkin R. Pari
Intel Technology India Pvt. Ltd.
Bangalore, India
pushkin.r.pari@intel.com

Gang Qu
Dept. of ECE and UMIACS
University of Maryland
College Park, MD 20742, USA
gangqu@eng.umd.edu

Abstract

A popular approach to guarantee fault tolerance in safety-critical applications is to run the application on two processors. A checkpoint is inserted at the completion of the primary copy. If there is no fault, the secondary processor terminates its execution. Otherwise, should the fault occur, the second processor continues and completes the application before its deadline. In this paper, we study the energy efficiency of such dual-processor system. Specifically, we first derive an optimal static voltage scaling policy for single periodic task. We then extend it to multiple periodic tasks based on worst case execution time (WCET) analysis. Finally, we discuss how to further reduce system's energy consumption at run time by taking advantage of the actual execution time which is less than the WCET. Simulation on real-life benchmark applications shows that our technique can save up to 80% energy while still providing fault tolerance.

1 Introduction

Real-time embedded systems are widely used in safety-critical applications, such as avionics platform and flight control system. Faults in such applications may cause catastrophic effect. Therefore, these systems must be designed not only to reduce the fault rate but also to provide the capability to tolerate fault should it occurs. Fault tolerance is normally realized on multi-processor system via temporal redundancy or physical redundancy depending on the availability of the task's laxity. A task's laxity is defined as the difference between its deadline and execution time. In temporal redundancy approach, the task's laxity is long enough so that the system can restart the execution either in the same processor or another one.In physical redundancy approach, the task does not have sufficient laxity and multiple (typically two) processors will be executing identical copies of the tasks simultaneously. Tsuchiya et al. [11] proposed several techniques that allow the two copies of the task to begin execution at different times in order to reduce the overlap time between them. Specifically, one processor executes the primary copy of the task and the other one executes the backup copy as late as possible while meeting the deadline. Manimaran and Murthy [7] developed a dynamic scheduling algorithm based on distance concept, backup overloading, and resource reclaiming to improve the multiprocessor system's performance in fault-tolerant. More recently, Rashid et al. [9] described a multiprocessor architecture with one lead processor executing the primary copy and two checkers executing, at a slower speed, the backup copy.

Meanwhile, low energy consumption has become one of the major design objectives for embedded real-time systems, especially battery operated portable devices. Low power/energy consumption can also improve run-time reliability of the circuit and therefore reduce the fault rate. Dynamic voltage scaling (DVS) technique varies the processor's operating voltage and hence clock frequency at run-time based on the workload and other factors. For fault-tolerant system, slack will also arise when fault does not occur. One can exploit this type of slack to reduce the energy consumption. Melhem et al. [8] presented uniform and non-uniform checkpoint placement policies for aperiodic as well as periodic tasks (by using earliest deadline first, EDF, scheduling) that allow a real-time system to recover from failure and minimize the energy consumption during failure-free operation. Unsal et al. [12] introduced non-EDF scheduling heuristics and obtained more energy reduction. Zhang and Chakrabarty proposed an adaptive checkpointing scheme which is combined with a dynamic voltage scheduling scheme to achieve power re-

*This work is partially supported by grant CNS0615222 from the National Science Foundation.

duction and increase the likelihood of timely task completion in the presence of faults [14].

In this paper, we study the energy efficiency of dual-processor system (unlike multiprocessor systems [7, 9]) that executes two copies of each task in different processors. The highlight of our work is an optimal DVS static power management scheme for single periodic task (unlike many previous work on the scheduling of a set of tasks [7, 12] or the placement of checkpoints [8, 14]) assuming the voltage can be changed continuously [13]. We also propose static/dynamic power management algorithms for multiple periodic tasks.

2 Problem Formulation

We consider a real-time fault-tolerant dual-processor system, which supports DVS, executing multiple periodic tasks without any deadline missing.

Application Model The dual-processor system processes multiple periodic tasks. Each periodic task can be characterized by (C_i, D_i, T_i), where C_i is the WCET at the reference voltage V_{max} that supports the maximum speed f_{max}, D_i is the deadline and T_i is the period with $T_i \geq D_i$. The total density of the tasks can be calculated by $\sum_{i=1}^{n} \frac{C_i}{D_i}$ and the utilization is defined as $\sum_{i=1}^{n} \frac{C_i}{T_i}$. We necessarily assume that the density is less than or equal to 1 in order to guarantee the schedulability of the task set.

DVS System Model The system has two processors each of which has its private memory and supports dynamic speed and voltage changes. The system's dynamic power consumption $P_d = C_{ef} \cdot V_{dd}^2 \cdot f$, operating speed (or clock frequency) $f = k \cdot (V_{dd} - V_t)^2 / V_{dd}$ where C_{ef} is the effective switching capacitance, V_{dd} is the voltage, k is a constant and V_t is the threshold voltage. When V_t is small enough to be negligible or can be proportionally adjusted at run time, speed becomes linear to V_{dd} [2]. Therefore, for a task that requires time e at the highest (reference) voltage V_{max} and the highest speed f_{max}, if the processor completes the task at a reduced voltage and speed in time $(e + l)$, the (dynamic) energy consumption E can be expressed as

$$E = P_{max} \cdot e \cdot (\frac{e}{e+l})^2 \tag{1}$$

where $P_{max} = C_{ef} \cdot V_{max}^2 \cdot f_{max}$ is the power consumption at the highest voltage and speed. Note that in this paper we normalize P_{max} to be 1.

Fault and Fault Recovery Model The dual-processor system executes two copies (primary and backup) of each task for fault tolerance and only needs one of the copies to be completed successfully before the deadline. One processor executes the primary copy and the other one executes the backup copy. When the processor completes the primary copy of the task without fault, it will notify the other processor to stop executing the backup copy.

For the fault-tolerant dual-processor system that executes multiple periodic tasks, based on the above models, we consider the problem of *when and at which voltage should two copies of each task be executed in order to reduce the system's total energy consumption while completing at least one copy of each task successfully without any deadline missing even when fault occurs.*

3 Power Management Schemes

Static power management (SPM) scheme schedules tasks in the offline fashion based on their WCET to minimize the energy consumption. Dynamic power management scheme leverages the slack between task's actual execution time and WCET to further reduce energy at run time.

3.1 SPM for Single Periodic Task

Assume that the system executes a single periodic task with deadline $D \leq T$, the period. Processor P1 will execute the primary copy of the task whose workload is e_1 at the reference voltage (highest speed f_{max}), whenever it is ready. Processor P2 executes the corresponding backup copy whose required workload at the highest speed is $e_2 \geq e_1$ [1]. When P1 and P2 only run at the highest speed and voltage, Tsuchiya et al. [11] have proposed a scheduling policy which minimizes the overlap length between the execution time slot of primary copy and backup copy (Figure 1 (a)). Let t_1 be the finishing time of the primary copy at P1 and t_2 be the start time of the backup copy at P2, this no power management (NPM) method indicates that $t_1 = e_1$ and

$$t_2 = \begin{cases} e_1 : & D \geq e_1 + e_2 \\ D - e_2 : & D < e_1 + e_2 \end{cases} \tag{2}$$

The expected energy consumption of the system per period is (recall that $P_{max} = 1$)

$$E = \begin{cases} e_1 + p \cdot e_2 : & D \geq e_1 + e_2 \\ 2 \cdot e_1 + e_2 - D + p \cdot (D - e_1) : & D < e_1 + e_2 \end{cases} \tag{3}$$

where $p \in [0, 1]$ is the fault rate of P1.

[1] For static power management, e_1 and e_2 are the worst case workloads of the task for P1 and P2, respectively. And e_1 is equal to e_2 when we do voltage scheduling. However for dynamic power management e_1 and e_2 are the remaining workloads of the task and e_2 may be greater than e_1 as P2 may start the execution later than P1. Note that the power management schemes introduced in this section can also be used in dynamic power management.

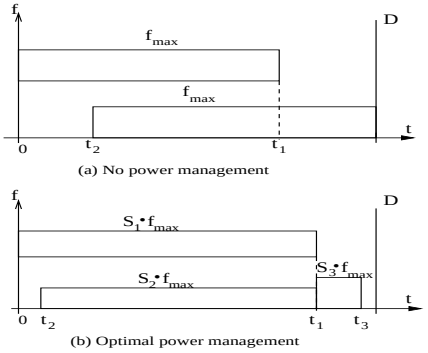


Figure 1. Different static power management schemes for single periodic task

When the dual-processor system, P1 and P2, supports dynamic voltage and frequency scaling, we propose the following optimal power management (OPM) scheme (see Figure 1 (b) for a graphic illustration):

P1: executes from the arrival of the task (time 0) at a fixed speed $S_1 \cdot f_{max}$ till the task's completion at time t_1.

P2: waits till time t_2 (can be 0) and executes at a fixed speed $S_2 \cdot f_{max}$ to P1's completion time t_1. If there is no fault, terminates; Otherwise, continues executing at a fixed speed $S_3 \cdot f_{max}$ to its completion at $t_3 \leq D$.

where $S_i \in [S_{min}, 1]$ and $S_{min} = \frac{f_{min}}{f_{max}}$. The problem becomes **determining** S_1, S_2, S_3, **and** t_2 **such that the total energy consumption is minimized**.

After tedious calculation, we identify that the optimal solution for this optimization problem either happens in the "interior" or one of the following boundaries: $S_1 = 1$, $S_2 = S_{min}$, $S_3 = 1$. We give the optimal "interior" solution when none of the speed S_i hits the boundary S_{min} or 1.

$$S_1 = \frac{k_1 S_3 e_1}{S_3 D - e_2} \quad (4)$$

$$S_2 = k_2 S_3 \quad (5)$$

$$S_3 = \frac{e_2 + e_1 k_3}{D} \quad (6)$$

$$t_2 = 0 \quad (7)$$

where $k_1 = 1 - \sqrt{p}, k_2 = \sqrt{p}$, and $k_3 = \sqrt[3]{(\frac{1}{\sqrt{p}} - 1)^2}$.

3.2 SPM for Multiple Periodic Tasks

Now we consider the system which executes multiple periodic tasks by using EDF algorithm. Each multiple periodic task is characterized by (C_i, D_i, T_i), where C_i is WCET, D_i is the deadline and T_i is the period. We can use the density of the tasks, which is $\sum_{i=1}^{n} \frac{C_i}{min(D_i, T_i)}$, to test its schedulability [5]. Here we assume that $D_i \leq T_i$. When the density $\sum_{i=1}^{n} \frac{C_i}{D_i} \leq 1$, the system is schedulable and we can assign the execution slot $\frac{C_i}{\sum_{i=1}^{n} \frac{C_i}{D_i}}$ to the i-th periodic task on both processors without violating the schedulability. The assigned time slot for each task may not be continuous because it may be preempted by the higher-priority task. In each time slot, we can employ the OPM algorithm to find the execution speeds and start time for the primary and backup copy of the corresponding task in order to reduce the energy consumption.

Note that the OPM algorithm may give the primary copy of the task a completion time earlier than its assigned time slot. Therefore although we only consider the WCET of each task, when fault does not occur on P1 there may still create slack, which is the time difference between the assigned time slot and the completion time. The slack will be released so it or part of it can be used by other tasks.

The static power management algorithm for multiple periodic tasks is shown in Figure 2. Before the task τ_n or τ_o is scheduled to execute, we apply the slack estimation algorithm proposed by Kim et al. [4] that considers the unused times of completed tasks and remaining times for uncompleted tasks to estimate the available time for the task (step 7). Then based on the remaining and available time, we use our OPM algorithm to obtain the appropriate voltage and speed for both processors (step 8).

```
1. On the completion of current task τ_c or the
     preemption of τ_c by a high-priority task τ_n ;
2. update the unused times of completed tasks and
     their remaining times for uncompleted tasks;
3. if τ_c is completed
4.     if the ready queue is not empty
5.         select task τ_o based on EDF;
6. if task τ_n or τ_o is available
7.     estimate the slack for τ_n or τ_o;
8.     use OPM algorithm to set voltage and speed;
9.     start executing the task;
```

Figure 2. Static power management algorithm for multiple periodic tasks.

3.3 Dynamic Power Management

The actual execution time of each task may deviate from its WCET, sometimes in a very large amount, and hence creates slack. Dynamic power management exploits such slack for further energy reduction. The simplicity and therefore low computation complexity

of the above optimal solution enables us to adopt it at the run-time for dynamic power management.

Note that for single periodic task, the slack created by the previous task instance can not be used by the current task instance because their time slots do not overlap. However for multiple periodic tasks, the slack created from one task may be used by other tasks because their execution time may be overlapped due to the preemption. Our dynamic power management algorithm for multiple periodic tasks is almost the same as Figure 2 except that there are additional slack arising from the difference between the actual task execution time and its WCET.

4 Simulation Results

The goal of our simulation includes the followings: (1) validation of the optimality of our static power management algorithm OPM for single periodic task, (2) demonstration of the energy savings by our static power management algorithm for multiple periodic tasks, and (3) showing the energy efficiency of the proposed dynamic power management algorithm.

4.1 Benchmarks and Simulation Setup

Table 1 gives the characteristics of a set of real-life benchmarks for fault tolerant applications that we have collected. These include Computerized Numerical Control (CNC) machine controller applications [3], the Avionics application [6], the Inertial Navigation System (INS) [1], and the videophone application [10]. Each benchmark consists of a set of tasks and the number of tasks is listed in the second column. The third and fourth columns give the range of these tasks' WCETs and periods, respectively. The last column shows the utilization.

Application	# of tasks	WCETS (ms)	Period (ms)	Utilization
CNC	8	$0.035 \sim 0.72$	$2.4 \sim 9.6$	0.489
INS	6	$1.18 \sim 100.28$	$2.5 \sim 1250$	0.736
Avionics	17	$1 \sim 9$	$25 \sim 1000$	0.850
Videophone	4	$1.4 \sim 50.4$	$40 \sim 66.7$	0.984

Table 1. The real-life benchmarks.

The synthetic applications are generated to have different utilization in $(0, 1]$. The fault is randomly generated based on the given fault rate that falls in $[0, 1]$. The task's actual execution time in our simulation follows a discretized normal distribution with average between 0 and WCET (WCET is normalized to 1) and a standard deviation between 0 and 0.25.

The dual-processor system schedules the tasks based on EDF policy and the two processors will execute the tasks in the same order. Both processors support dynamic voltage and frequency scaling and the frequency changes from $f_{min} = 0.3 \cdot f_{max}$ to f_{max} when the voltage changes in the corresponding range. For static power management scheme, we consider one hyperperiod (the least common multiple of task periods) based on the task's WCET. For dynamic power management, we simulate 10,000 hyperperiods for each benchmark with the above actual execution time model.

4.2 Results of the SPM Schemes

We first use a "guided" exhaustive search to verify the optimality of the solution (the value of S_1, S_2, and S_3 for the dual-processor system) provided by our OPM scheme. We set S_1 to go from $S_{min} = 0.3$ to $S_{max} = 1.0$ with an increment of 0.01. For each fixed S_1 of the primary processor, we can easily find the optimal value of S_2 and S_3 for the secondary processor. This gives us the best static scheduling policy for a fixed S_1 . For a single task, we compute its energy consumption by NPM scheme and then simulate its execution with this speed setting, obtain its energy consumption, and calculate its saving over NPM.

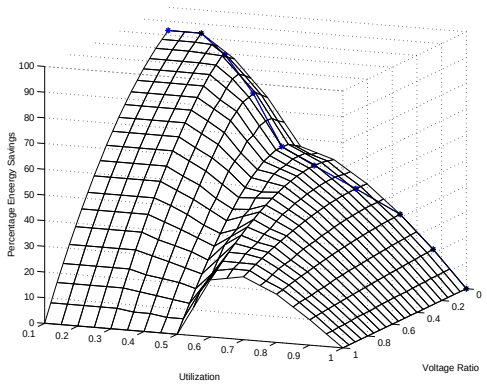
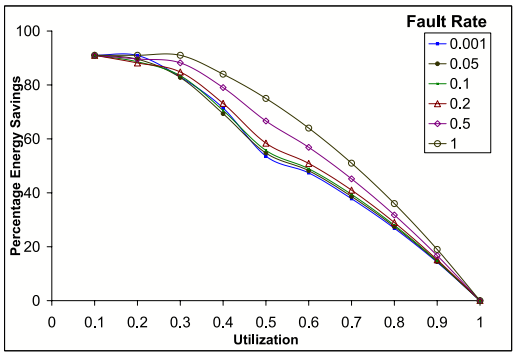


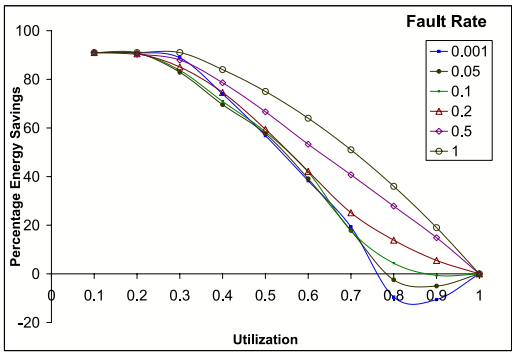
Figure 3. Optimality of OPM: the energy savings of different static scheduling policies over NPM for a single periodic task.

Figure 3 reports the energy saving for different utilization value when the fault rate is 0.001. The voltage ratio R is defined as $\frac{S_1 - S_{min}}{S_{max} - S_{min}}$ and it increases from 0 to 1 as S_1 goes from S_{min} to S_{max}. For each fixed value of utilization, we mark a * on the energy saving by the optimal voltage setting as shown in Figure 3. This optimal voltage setting coincides with the one from our proposed OPM algorithm within the error of voltage increment (which corresponds to a $0.01 * S_{max}$ increase in speed) we set earlier.

Figures 4(a) and 4(b) report the energy saving of SPM schemes over NPM for different utilization and fault rate. For single task, we see that the energy saving increases as the fault rate increases or the utilization decreases (Figure 4(a)). When the utilization is low, there are more slack which is in favor of the voltage scaling based OPM approach. When the fault rate is high, the second processor often continues the execution after the first processor completes its execution. The increase of energy saving in this case is due to the optimal setting of the second processor's speed (S_2 and S_3) by OPM.



(a) The case of single periodic task.



(b) The case of multiple periodic task.

Figure 4. The energy saving of SPM schemes over NPM for different utilization and fault rate.

Figure 4(b) reveals that OPM's performance on energy saving for multiple periodic tasks is similar except when the utilization is high. In that case, there will (always) be tasks available in the ready queue. Since our SPM algorithm (Figure 2) uses all of the available slack for the current task, there is almost zero slack released after its completion even when the fault does not occur. On the other hand, for NPM when fault does not occur, there is some slack released by terminating the execution of processor P2. When there is slack available, NPM delays the execution of the task in P2 in order to minimize the overlap time between the execution of P1 and P2. Therefore when the fault does not occur the slack will be accumulated and the energy consumption will be reduced in NPM, but not in our algorithm.

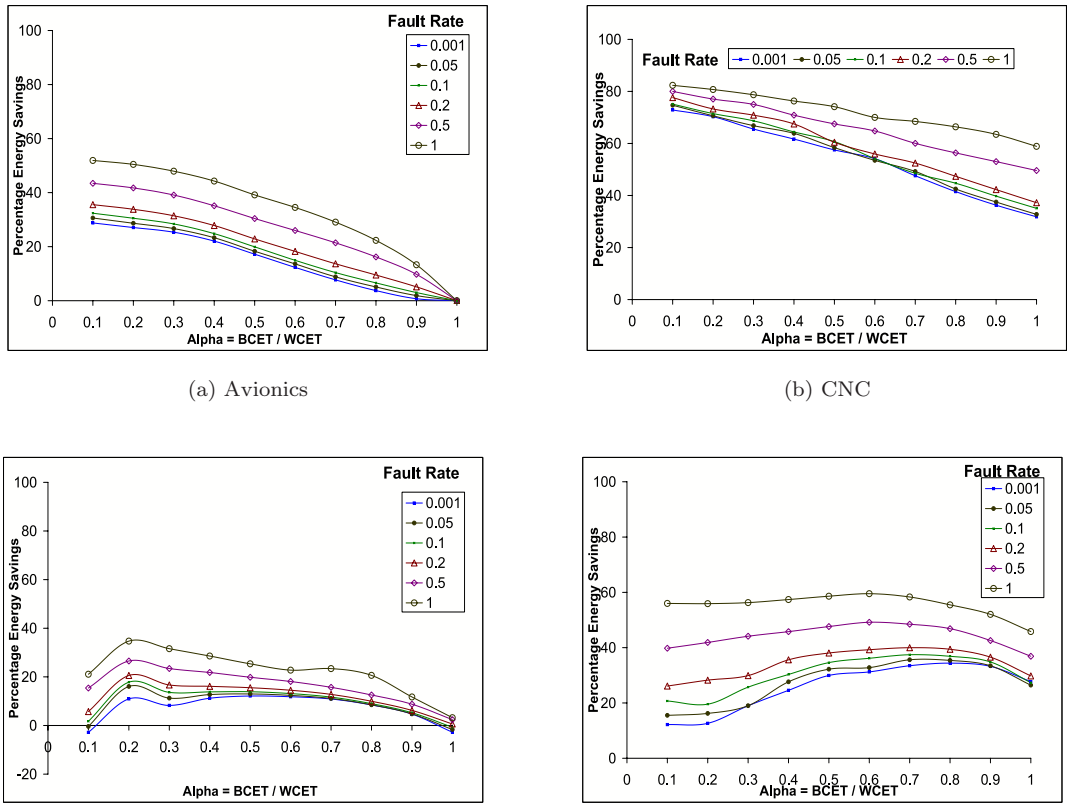
4.3 Results of the Dynamic Scheme

We apply our proposed dynamic power management scheme to the set of real-life benchmarks shown in Table 1. Figure 5 reports our energy savings over NPM for different fault rate and different BCET/WCET ratio. In almost all the cases, our algorithm is more energy efficient. For example, on the CNC benchmark (Figure 5(b)) the energy saving can be as high as 80%. Similar to the analysis of OPM, we see that the higher the fault rate, the more energy saving by our dynamic power management scheme.

In general, when the BCET/WCET ratio increases, the task's actual execution time becomes closer to WCET with less variance. Therefore we expect less energy saving. However, it is interesting to see that in some applications (CNC and INS), significant amount of energy saving is reported even when BCET and WCET become the same. We suspect that the impact of the BCET/WCET ratio to energy saving depends on the structure of the specific application and it is being investigated currently. In the Videophone application, when the BCET is close to WCET or much less than WCET, our algorithm performs slightly worse than NPM. This is caused by the high utilization of this application for the same reason as we have discussed in the previous section 4.2.

5 Conclusions

We consider the dual-processor system that concurrently runs two copies of the same applications in different processors in order to tolerate fault. We have developed voltage scheduling techniques on such a system in order to reduce the total system energy consumption while still tolerating the fault occurrence. Specifically, we propose an optimal static voltage scaling scheme for single periodic task, which can also be used for multiple periodic tasks to save energy. In order to further reduce the energy consumption, the dynamic power management scheme will be applied to exploit the slack arising from the variation of task's execution time during run time. The experiment results confirm our claims and show that our techniques can save up to 80% energy vs. no power management scheme while still providing fault tolerance.



(a) Avionics

(b) CNC

(c) Video Phone

(d) INS

Figure 5. Dynamic power management scheme: energy savings over NPM on real-life benchmarks.

References

[1] A. Burns, K. Tindell, and A. Wellings. "Effective analysis for engineering real-time fixed priority schedulers", *IEEE Tran. on Software Eng.,* Vol. 21, pp. 475-480, 1995.

[2] F. Gruian. "System-Level Design Methods for Low-Energy Architectures Containing Variable Voltage Processors", *Proc. of the Workshop on Power-Aware Computing Systems,* pp. 1-12, 2000.

[3] N. Kim, M. Ryu, S. Hong, M. Saksena, C. Choi, and H. Shin. "Visual Assessment of a Real-Time System Design: a Case Study on a CNC controller", *Proc. of IEEE Real-Time Systems Symposium,* pp. 300-310, 1996.

[4] W. Kim, J. Kim, and S. L. Min. "A Dynamic Voltage Scaling Algorithm for Dynamic-Priority Hard Real-Time Systems Using Slack Time Analysis", *Proc. of Design, Automation and Test in Europe,* pp. 788-794, 2000.

[5] J. W. S. Liu. "Real-Time Systems", *Prentice Hall,* 2000.

[6] C. Locke, D. Vogel, and T. Mesler. "Building a Predictable Avionics Platform in Ada: a Case Study", *Proc. of IEEE Real-Time Systems Symposium,* pp. 181-189, 1991.

[7] G. Manimaran and C.S.R. Murthy. "A Fault-Tolerant Dynamic Scheduling Algorithm for Multiprocessor Real-Time Systems and Its Analysis", *IEEE Tran. on Parallel and Distributed Systems*, Vol. 9, No. 11, pp. 1137-1152, 1998.

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

[9] M.W. Rashid, E.J. Tan, M.C. Huang, and D.H. Albonesi. "Power-Efficient Error Tolerance in Chip Multiprocessors", *IEEE Micro*, pp. 60-70, Nov.-Dec. 2005.

[10] D. Shin, J. Kim, and S. Lee. "Intra-Task Voltage Scheduling for Low-Energy Hard Real-Time Applications", *IEEE Design and Test of Computers,* Vol. 18, No. 2, pp. 20-30, 2001.

[11] T. Tsuchiya, Y. Kakuda, and T. Kikuno. "A New Fault-Tolerant Scheduling Technique for Real-Time Multiprocessor Systems", *International Workshop on Real-Time Computing Systems and Applications,* pp. 197-202, 1995.

[12] O.S. Unsal, I. Koren, and C.M. Krishna. "Towards Energy-Aware Software-Based Fault Tolerance in Real-Time Systems", *International Symposium on Low Power Electronics and Design,* pp. 124-129, 2002.

[13] L. Yuan and G. Qu. "Analysis of Energy Reduction on Dynamic Voltage Scaling-Enabled Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 24, No. 12, pp. 1827-1837, 2005.

[14] Y. Zhang, and K. Chakrabarty. "Energy-Aware Adaptive Checkpointing in Embedded Real-Time Systems", *Proc. Design, Automation and Test in Europe Conference,* pp. 918-923, 2003.