Offline Replication and Online Energy Management for Hard Real-Time Multicore Systems

Farimah R. Poursafaei, Sepideh Safari, Mohsen Ansari, Mohammad Salehi, Alireza Ejlali
ESRLab, Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
{poursafaei,ssafari,mansari,mohammad_salehi}@ce.sharif.edu, ejlali@sharif.edu

Abstract— **For real-time embedded systems, energy consumption and reliability are two major design concerns. We consider the problem of minimizing the energy consumption of a set of periodic real-time applications when running on a multi-core system while satisfying given reliability targets. Multi-core platforms provide a good capability for task replication in order to achieve given reliability targets. However, careless task replication may lead to significant energy overhead. Therefore, to provide a given reliability level with a reduced energy overhead, the level of replication and also the voltage and frequency assigned to each task should be determined cautiously. The goal of this paper is to find the level of replication, voltage and frequency assignment, and core allocation for each task at design time, in order to achieve a given reliability level while minimizing the energy consumption. Also, at run-time, we find the tasks that have finished correctly and cancel the execution of their replicas to achieve even more energy saving. We evaluated the effectiveness of our scheme through extensive simulations. The results show that our scheme provides up to 50% (in average by 47%) energy saving while satisfying a broad range of reliability targets.**

Index Terms–**Real-Time Embedded Systems, Task Replication, Energy Management, Multicore Systems.**

I. INTRODUCTION

One of the primary design constraints of real-time embedded systems is energy consumption. More energy consumption may lead to reduced battery life as well as higher temperature. Specifically, in embedded systems, the problem of managing energy is worse since most of these systems have limited power supplies. There are two well-studied techniques to face with the problem of high energy consumption: *Dynamic Power Management (DPM)* [1] [2] and *Dynamic Voltage and Frequency Scaling (DVFS)* [1] [3] [4]. DPM is a general technique that can be applied to any component of the system. It puts the system components into a low-energy state when they are idle [2] [5]. DVFS reduces voltage and frequency in order to save energy [4] [6]. It has been shown that DPM provides less energy saving compared to DVFS in similar conditions [7] [8]. However, since DVFS prolongs tasks execution time, exploiting DVFS in real-time embedded system may lead to violation of timing constraints associated with real-time tasks.

One other issue affecting computer systems is their susceptibility to faults which may lead to different run-time errors. Generally, faults are classified into three different types: *transient, intermittent* and *permanent* faults [9] [10]. What may lead to a transient fault is electromagnetic interference or cosmic radiation [11] [12] which can be tolerated by time redundancy technique like re-execution of an affected task [9] [10]. Intermittent faults never go away entirely and oscillate between being dormant and active. When these faults are dormant, the component functions normally, while at some intervals when they are active, they result in malfunctioning of the affected component. Permanent faults are caused by manufacturing defects or circuit wear-out. In order to confront with permanent faults, the only way is through hardware redundancy techniques [9] [10]. It has been shown that transient faults occur more frequently than the other two types [11] [12]. Therefore, confronting transient faults is of great importance.

Embedded systems are frequently used in hard real-time applications where the demand to reliability is very high. Managing energy and achieving fault tolerance in these systems are often at the odd due to the fact that fault tolerance mainly requires exploiting redundant resources [13] [14] [15] [16] [17]. When DVFS is chosen as a means of energy management technique, its negative impact on reliability should be considered, as the current researches suggest [18] [19]: the transient fault rate (and thus the corresponding soft errors) increases exponentially as we decrease the supply voltage and processing frequency for saving energy. Therefore, in hard real-time applications, the degradation of reliability because of applying energy management techniques should be considered.

Using available slack time in the system for both reliability and energy management have been considered by a set of techniques called *Reliability-Aware Power Management (RAPM)* [20] [21]. The main idea of these works is to preserve system's original reliability through scheduling redundant executions and then exploiting the remaining slack time for energy management. In addition to these techniques, there is another framework proposed for periodic real-time tasks running on a single core which is called *reliability-oriented energy management* [22]. In this work, the main goal is to achieve different reliability levels with minimum energy consumption. The difference of this work in comparison to RAPM is that the desired reliability levels can be different from the original reliabilities of the tasks. This offers a flexibility that can be exploited for energy management.

In this paper, we focus on a reliability-oriented energy management solution for multi-core systems. The increase in the number of cores in the emerging many-core systems

978-1-4673-8047-8/15/$31.00 ©2015 IEEE

makes abundant opportunities to exploit task replication as a proper option for reliability management. In [23], the authors suggested that by scheduling multiple copies of a task on different cores in a multicore system, the probability of finishing at least one copy successfully increases significantly. Also, it has been proved that replication has some intrinsic features that make it an efficient means of managing reliability [23]. For one thing, task replication makes high reliability targets achievable. Moreover, replication helps in reducing energy consumption by reducing the supply voltage and processing frequency of the copies of the tasks in addition to achieving reliability targets.

Considering the benefits of task replication, we proposed an offline replication and online energy management scheme for hard real-time applications running on multicore systems. Our scheme consists of two phases: an *offline* phase, and an *online* phase. In the offline phase, the degree of replication and the voltage and frequency assignment for each task with respect to a given reliability level is determined. Moreover, by specifying the task to core allocation, the system configuration and the tasks execution sequence are determined. At run-time, an online manager controls the system and prevents the unnecessary execution of any task that at least one of its copies has finished successfully. The required number of copies for each task to meet its reliability target is specified statically in the offline phase, while the online manager controls the number of copies of each task.

The main contribution of this paper is to find the efficient configuration of the system in terms of the number of copies for each task along with the voltage and frequency assignment of tasks offline (at design time), and to manage the execution of the tasks' copies to prevent unnecessary executions online (at run time).

The rest of this paper is organized as follows. In Section II, some models and assumptions are presented. Section III covers the analysis of the impact of adjusting the degree of replication. Problem definition and the proposed solution will be presented in Section IV. Evaluations and results are presented in Section V. Finally, Section VI concludes the paper.

II. Models and Assumptions

A. Workload Model

Our workload consists of a set of N periodic real-time tasks $T=\{\tau_1, \tau_2, ..., \tau_N\}$. Each task τ_i, under the maximum available processing frequency f_{max}, has the worst-case execution time WC_i and period P_i which is equal to its relative deadline. The workload is assumed to be executed on M similar cores. Each core has F different frequency levels. We use normalized frequency f with respect to the maximum available processing frequency $f_{\max}$ ($0<f\leq f_{\max}=1.0$). The cost of switching between the frequencies (and their corresponding voltages) has been assumed to be negligible. Each task τ_i has k_i copies which will be executed on $k_i \leq M$ distinct cores. The allocated tasks to each core will be scheduled by *Earliest-Deadline-First (EDF)* scheduling which is proved to be optimal on a single core [24].

B. Energy Consumption Model

We consider that the total energy consumption of the system, E_{total}, consists of the static and dynamic energy components. Each core in the system can operate in active or idle states. In active state the core executes tasks and it consumes dynamic and static energy. When there is no task to be executed the cores goes to the idle state and consumes only static energy. The energy consumption of a core is computed as [25]:

$$E_{total} = E_s + E_d \tag{1}$$

where the static energy which is dominated by the leakage current of the system is denoted by E_s, and the dynamic energy is denoted by E_d.

We assume that all cores in the system are equipped with DVFS. During the execution of each task, its supply voltage V_i, may be less than the maximum supply voltage V_{max}. We consider the normalized supply voltage ρ_i as:

$$\rho_i = \frac{V_i}{V_{\max}} \tag{2}$$

The dynamic energy consumption of each core ($E_d(\tau_i)$) when executing the task τ_i, at the scaled supply voltage V_i can be written as:

$$E_d(\tau_i) = C_{eff} V_i^2 f_i \left(\frac{WC_i}{\rho_i} \right) \tag{3}$$

where C_{eff} is a constant that expresses effective switching capacitance, and V_i and f_i are the supply voltage and operational frequency respectively. WC_i/ρ_i is the scaled task execution time due to DVFS. Since in DVFS, voltage has a linear relationship with frequency, we have $\rho_i=V_i/V_{max}=f_i/f_{max}$ where V_{max} is the maximum supply voltage corresponding to the maximum processing frequency f_{max}. Therefore, Eq. 3 can be rewritten as [25]:

$$E_d(\tau_i) = C_{eff} V_{\max}^2 f_{\max} \rho_i^2 WC_i \tag{4}$$

Since we have no direct control over $C_{eff}V_{max}^2f_{max}$, we can use the normalized energy consumption by removing $C_{eff}V_{max}^2f_{max}$. Therefore, the normalized dynamic energy consumption of each core $NE_d(\tau_i)$, while executing the task τ_i can be written as:

$$NE_d(\tau_i) = \rho_i^2 WC_i \tag{5}$$

In our evaluations, we set the static energy consumption E_s, equal to 5% of the maximum dynamic energy like the assumption that has been adopted in similar works [23] [25].

C. Fault Model

The arrival rate of faults can be modeled by an exponential distribution [18]. It has been shown that the average arrival rate of faults increases as the frequency decreases through DVFS [18] [19]. If the average arrival rate of faults is λ, and the average fault rate at the maximum frequency is λ_0, at frequency f, the fault rate is [19]:

$$\lambda(f) = \lambda_0 10^{\frac{d(1-f)}{(1-f_{\min})}} \tag{6}$$

where d is a parameter called *sensitivity factor* and the typical values for d ranges from 2 to 6 [19] [21] [26]. Following [22], we assume that at the maximum available frequency, the value of λ_0 is equal to 10^{-6}.

The probability of executing a task successfully is called the task's reliability. At frequency f_i, the reliability of a task ($R_i(f_i)$) is:

$$R_i(f_i) = e^{-\lambda(f_i)\frac{WC_i}{f_i}} \tag{7}$$

Consequently, the *Probability of Failure (PoF)* of a task is:

$$PoF_i(f_i) = 1 - R_i(f_i) \tag{8}$$

When we have several copies of each task, it is enough to have at least one instance which has been finished successfully. So, at the end of execution of each task a test is conducted to reveal whether any fault has occurred or not [9] [10].

III. Degree of Replication Analysis

Having in mind the previous discussion, when we have several copies of a task, the execution will be unsuccessful only if all copies encounter faults. So, in general, the probability of failure of a task with k copies decreases exponentially:

$$PoF_i^{(k)} = \left(PoF_i(f)\right)^k \tag{9}$$

Based on [23], it can be seen that to face with negative impact of voltage and frequency scaling on the probability of failure, we can use replication and we may even get some energy savings through parallel execution. Through Eq. 9, given a certain PoF target (PoF_{targ}), we can find the minimum number of copies for a task to satisfy the reliability targets as follows [23]:

$$PoF_{t\arg} \le \left(PoF_i(f)\right)^k \tag{10}$$

$$k \ge \left\lceil \frac{\log\left(PoF_{t\arg}\right)}{\log\left(PoF_i(f)\right)} \right\rceil \tag{11}$$

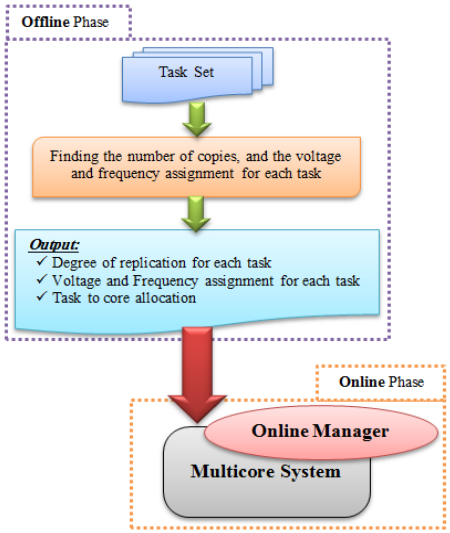

Fig. 1. Overview of our proposed scheme.

We can conclude that for a given reliability target, different number of copies and voltage and frequency levels can satisfy the need; On the other hand, the energy consumption of these different combination of the number of copies, and supply voltages and processing frequencies that are able to achieve the same reliability targets differ significantly [23]. It should be noted that the number of copies of a task should not exceed the total number of available cores in the system. So, in the selection of the execution frequency of a task, the upper bound on the number of copies should be considered. This constraint may lead to have fewer numbers of copies with probably higher frequency to achieve the reliability target.

IV. Problem Definition and Solution

Given a multicore system consisting of M similar cores, and a set of N periodic tasks with their task-level reliability targets, the first part of the problem is to determine the number of copies and the voltage and frequency assignment for each task in order to reduce the energy consumption, while the allocation of tasks to core needs to guarantee that all timing constraints are met. Moreover, no more than one copy of a task can be allocated to the same core. The second part of the problem is to save further energy during the actual execution of the tasks through the addition of an online manager to the system; the responsibility of the online manager is to apply DPM. The online manager needs to find the first copy of each task that has finished successfully and cancel the execution of the remaining part of the other copies on the other cores. Based on the rare nature of the fault occurrence, most of the time, there is no need to execute all copies of a task completely. So, as soon as a task finishes successfully, the online manager stops the execution of the remaining part of the copies and avoids dynamic energy consumption and further energy can be saved.

Our solution for dealing with this problem is through exploiting a combination of offline replication and online

energy management; our proposed scheme consists of two phases: the offline phase is solved by *Energy-Efficient Replication (EER)* which is a proficient heuristic-based algorithm presented in [23]. In the online phase, the main objective is to add an online control over the underlying system to gain further energy savings. An overview of procedures taken place in our proposed scheme is shown in Fig. 1. The details of each of these two phases are presented in the following subsections.

Offline Phase: It has been proven that problem with which we are confronted in offline phase, belongs to the class of NP-hard problems. Therefore in order to solve it, the heuristic-based algorithm EER has been proposed [23]. In our offline phase of tackling with the presented problem, our scheme follows the EER algorithm. In this phase, at first, the *Energy-Frequency-Reliability (EFR)* table for each task τ_i, will be constructed. Entries in each row of this table consist of a frequency level (from the set of all available frequency F in the system), the minimum number of copies required to satisfy the reliability target of the task according to that specific frequency level, the corresponding overall energy consumption considering all copies, and the total processor time needed by all the copies. The rows in this table are placed in a way that the frequency levels are sorted in descending order, with the maximum available frequency at index 1 of the EFR table. After constructing the EFR table, the algorithm tries to partition the workload among the set of available M cores in the system. For allocating tasks to cores, a modified version of *First-Fit-Decreasing (FFD)* algorithm has been adopted. Some of the notable modifications are as follows: along with the original scheme in FFD heuristic, the total utilization of the assigned workload to each core should not be more than one, due to schedulability condition of EDF scheduling. Moreover, the allocation should not assign copies of the same task on the same core. Otherwise, if a permanent fault occurs at a core, more than one copies of a task will be corrupted. In the partitioning of the tasks among the cores, in the first attempt, the EER algorithm starts with the most energy-efficient configuration in which each task executes at its least frequency level. If this trial is successful, the optimal solution for the problem will be found and the EER algorithm will be terminated; if not, the algorithm moves into its next step. In this step, first we need to know whether the problem has a feasible solution at all or not. So, the procedure continues by checking the other extreme where all tasks are running at their maximum available frequency in the system. If the algorithm cannot find any solution in this configuration either, the task set is not schedulable at all, and the algorithm is terminated by a message. Otherwise, the EER algorithm continues with the relaxation phase. In its relaxation phase, EER starts with a feasible configuration (where at the first run, each copy of each task is running at the maximum frequency). In each of its iterations, the algorithm tries to select a task and reduce its frequency. Selection of the tasks for reducing frequency is according to two of the proposed heuristics in [23] which will be introduced later. If the new configuration of the system is feasible, the algorithm moves to another round of relaxation and the new configuration is committed to, otherwise, the task will be omitted from the set of eligible tasks, will not be considered for the further iterations, and the system backtracks to its previous feasible configuration. Also, in a case when a task reaches its minimum frequency level in its EFR table, it will be omitted from the set of eligible tasks for the next iterations. This procedure continues until there are no more tasks in the set of eligible tasks. Finally, the last

Algorithm 1: Offline Replication and Online Energy Management

```
1.  Start the offline phase
2.  /* In the offline phase, the EER algorithm is applied */
3.     Construct the EFR tables for all tasks
4.     for each task τ_i in the task set do
5.        /*assume that for each task τ_i, the most energy-efficient
          frequency-level in its corresponding EFR table is denoted by
          level-ee_i */
6.        CurrentLevel[τ_i] ← level-ee_i
7.     end for
8.     SysConfig ← Partition the currentLevel with modified FFD
9.     if (feasible(SysConfig)) then
10.       return the CurrentLevel and the SysConfig
11.       exit offline phase
12.    end if
13.    for each task τ_i in the task set do
14.       CurrentLevel[τ_i] ← 1;
15.       eligible[τ_i] ← true;
16.    end for
17.    SysConfig ← Partition the currentLevel with modified FFD
18.    if (!feasible(SysConfig)) then
19.       return error;  /* No feasible solution exists */
20.       exit
21.    end if
22.    while (∃j (eligible[τ_j] is true)) do
23.       Choose the most eligible task according to LEF or LPF
24.       /*τ_i is chosen for relaxation*/
25.       CurrentLevel[τ_i]++;  /*goes to the less frequency-level*/
26.       SysConfig ← Partition the currentLevel with modified FFD
27.       if(!feasible(SysConfig)) then
28.          CurrentLevel[τ_i]--; /*goes back to the last feasible-level*/
29.          eligible[τ_i] ← false;
30.       end if
31.    end while
32.    return the CurrentLevel and the SysConfig
33. End the offline phase
34. /*The CurrentLevel and the SysConfig are given to the online
    phase*/
35. Start the online phase
36.    /*In the online phase, the Online Manager tries to apply
       DPM*/
37.    for each task τ_i in the currentLevel do
38.       coreSet[τ_i]← find the set of all cores executing a copies of τ_i
39.       online_Manager controls the system;
40.          find the first instance at which τ_i is completed successful-
             ly on one of the core in coreSet_i
41.          cancel all the remaining part of all other copies of τ_i on all
             other cores in coreSet_i
42.    end for
43. End the online phase
```

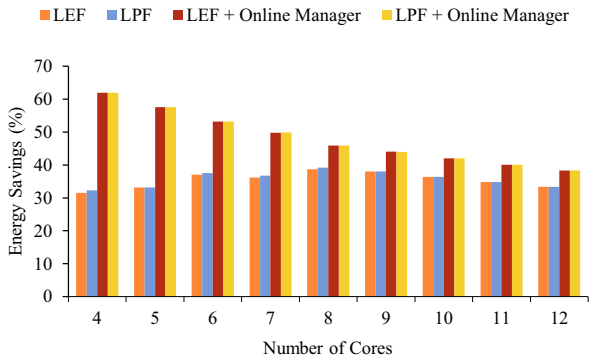

Fig. 2. Impact of number of cores; $U_{tot} = 1.5$, $w = 0.01$.

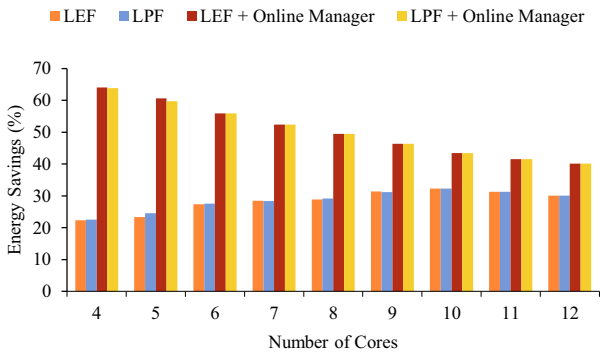

Fig. 3. Impact of the number of cores; $U_{tot} = 1.5$, $w = 0.001$.

feasible configuration of the system is chosen. In the offline phase of our scheme, two of the three heuristics introduced in [23] has been applied in order to choose the most eligible task. These heuristics are as follows:

- *Largest-Energy-First (LEF)*: The most eligible task is the task that will results in the most energy saving by relaxing its frequency to a less frequency level [23].
- *Largest-Power-First (LPF)*: The task that gives the largest energy savings per the additional processing time by one-level reduction of its frequency is chosen as the most eligible task [23].

Online Phase: In the online phase, having the system configuration from the previous phase, the objective of our scheme is to reduce the dynamic energy consumption. Keeping in mind that transient faults are rare in nature, if an online control can be added to the system to cancel the unnecessary copies of the tasks, further energy savings can be obtained. In EER algorithm, the goal is to find a configuration in which the energy is minimized while the reliability target is satisfied [23]. However, there is no need to execute all copies of a task under any circumstances completely. Considering that when a task whose processing frequency is reduced finishes its execution successfully, the online manager cancels the remaining part of its copies, resulting in significant energy saving. Therefore, after finding the proper system configuration in the offline phase, the online manager controls the system as it is running the tasks on different cores. If a task finishes successfully, the online manager stops execution of its copies and then put the system into a low-power state which only consumes static energy (i.e. applying DPM). We have seen that by adding this online manager we can obtain significant energy saving compare to the pure EER algorithm proposed in [23].

Algorithm 1 shows the pseudo code of our offline replication and online energy management scheme. In this Algorithm, the offline phase starts from line 1. In line 3, for each task, the EFR table is constructed. Lines 4-8 show the first trial of the algorithm to find the system configuration. In this trial, each task has the most energy-efficient configuration (i.e. executing at its least frequency level). In lines 9-12, EER algorithm checks whether the most energy-efficient configuration is feasible; if this configuration is feasible, the optimal solution is found, and the corresponding level of frequency for each task as well as the system configuration are given to the online phase; otherwise, in lines 13-21, the offline phase checks whether there exists a feasible configuration of the system at all (with all the tasks running at the maximum frequency). If no configuration is found, the algorithm is terminated by an error message; otherwise, the EER algorithm continues to its relaxation phase. Through lines 22-32, the algorithm tries to find a more suitable configuration of the system by the means of the relaxation phase of EER algorithm. In the relaxation phase, for choosing the most eligible task for frequency reduction, one of the two heuristics (LEF or LPF) based on [23], is selected. The offline phase finishes at line 33. After finding the most energy-efficient configuration of the system in the offline phase, the online phase starts at line 35. Lines 37-42 specify the addition of the online manager to the system. The goal of this manager is to apply DPM and avoid unnecessary execution of different copies of a task. Specifically, line 38 shows the process of finding the set of all cores on which a specific task τ_i is executing. The online manager monitors the cores in this set to find the first copies of τ_i that has been finished successfully, and cancels the unnecessary execution of the other copies, in lines 39-41. Finally, the online phase finishes at line 42.

V. Performance Evaluation

In this section our simulation results are presented. As a baseline, we consider the configuration of the system where all copies of all tasks are running at the maximum available frequency (i.e. f_{max}). We report the energy saving results of our scheme and the pure EER algorithm in [23] in compare to the baseline.

The energy savings of the different schemes is evaluated by the use of an in-house discrete event simulator. For each point in the figures, the average energy savings of 1000 data sets each contains 20 tasks is reported. For producing tasks, we followed the scheme in [27] which produces task according to *UUnifast* scheme. We have 10 different frequency levels in the system and the static energy is set to 5% of the maximum dynamic energy consumption.

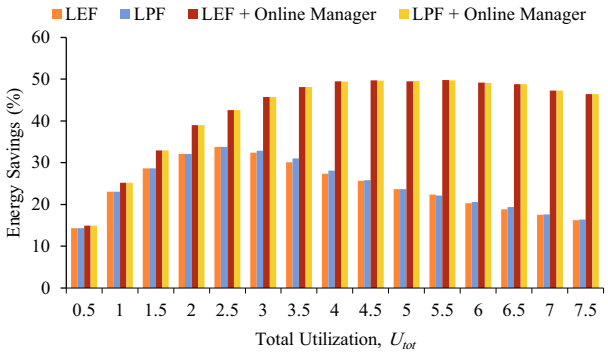

Fig. 4. Impact of system load (tasks utilization); 16 cores, $w = 0.01$.

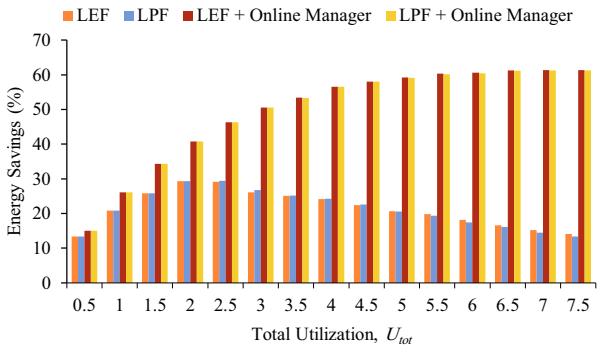

Fig. 5. Impact of system load (tasks utilization); 16 cores, $w = 0.001$.

In our simulations, the results of applying the pure EER algorithm [23] in the offline phase are specified by LEF and LPF (according to the heuristics used for selecting the most eligible task for the frequency reduction). On the other hand, when the online manager is added to the system, the results are specified by "LEF + Online Manager" and "LPF + Online Manager" respectively.

It should be noted that if the given reliability target is the system target reliability ($R_{sys}(t)$), the first thing to do is finding the task-level reliability targets ($R_i(t)$). In such a case, based on the approach proposed in [22] a *uniform reliability scaling* is performed in such a way that, for each task, the ratio $(1-R_i(t))/(1-R_i(t_0))$ is set to a unique value w which is a constant value and is called the *(uniform) PoF scaling factor*. By multiplying the normalized PoF of each task to w, we scale all the original task reliabilities in the same way [22]. In our evaluation, we assume that the PoF of each task is in normalized form, with respect to the PoF of a single task running at the maximum available frequency.

In the first set of the simulations, we investigate the impact of the number of cores on energy savings. The results are shown in Fig. 2 and Fig. 3. In these experiments, the total system load is set to 1.5, and the value of w is set to 0.01 and 0.001, respectively. In these figures, by increasing the number of cores, there is more space for the LEF and LPF to reduce the frequency levels of tasks in addition to the probable increase of the number of copies. In these situations, by increasing the number of cores, the offline phase results in more energy savings as there is more capacity to slowdown the tasks, however because the overlapping parts of the copies become larger (that is the result of extending tasks execution times), in the online phase, the trend of energy savings is decreasing; even though the energy savings is more than the pure offline phase in all cases. Moreover, in Fig. 3, when the w is smaller (w=0.001), the required reliability of the tasks are higher and it leads to executing tasks in higher frequency or probably with more number of copies, therefore leads to less energy savings in total. Notice that when the number of cores grows, at some point, achieving the most energy-efficient configuration of each task becomes feasible. Therefore, for larger number of cores, the energy savings become almost unchanged.

In the second set of the simulations, we evaluate the impact of the total system loads (U_{tot}), on the energy savings. The results have been shown in Fig. 4 and Fig. 5. In these experiments, the system has 16 cores, and the value of w is set to 0.01 and 0.001, respectively. According to these figures, generally when the total utilization of the system is low, the available slack is larger and the chance of running each task at the most energy-efficient configuration is high. On the other hand, when the system load increases, the amount of the available slack becomes lower. Therefore, the ability of the offline phase of our scheme to save energy decreases and that is due to the fact that with high load, the system will have little opportunity to scale tasks. So, these tasks require executing in higher frequency or with more number of copies for sake of satisfaction of the reliability targets, and this leads to more energy consumption. Although when the system utilization moves toward higher values, the trend of energy savings for pure offline phase of the algorithm (i.e. LEF and LPF) is decreasing, by adding the online manager to the existing system, we will get more energy savings. This increasing trend stems from the fact that when system load is high, since there is smaller slack available, the tasks cannot be scaled. Therefore, the overlapping parts of replicated tasks may become shorter, and when the online manager decides to cancel the remaining part of the copies, shorter overlapping parts have been passed and this will result to more energy savings. Moreover, when tasks cannot be scaled, they run at a higher level of frequency, which results in more reliable execution of each copies of each task. Therefore, the chance of finishing tasks successfully increases and there is less need of executing other copies of the tasks completely.

It is noteworthy that both heuristics, LEF and LPF, works much the same way, and result in similar energy savings. This similar behavior is followed when the online phase is added to the pure offline phase. However, in some cases, one of the two heuristics may outperform the other one.

VI. Conclusions

In this paper we considered the problem of managing energy consumption for a set of periodic real-time tasks with specified reliability targets, when running on a multicore system. For tackling with this problem, we introduced an offline replication and an online energy management scheme

consisting of two phases: in the offline phase, we try to find the number of copies as well as processing frequency for each task; while in the online phase, we add an online manager to the existing system of the first phase. The online manager applies DPM to the system, i.e. when a task finishes successfully, the execution of the other copies of the same task is canceled and system will go to a low-energy state. We have seen that by selecting the proper degree of replication and frequency assignment for each task, and adding the online manager, we are able to gain significant energy savings. Therefore, the addition of this online manager is very important in energy saving because of the infrequent nature of the faults affecting the system in general.

ACKNOWLEDGEMENT

The authors of this paper acknowledge Research Vice Presidency of Sharif University of Technology for funding this work under grant no. G930827.

REFERENCES

[1] M. Salehi, and A. Ejlali, "A Hardware Platform for Evaluating Low-Energy Multiprocessor Embedded Systems Based on COTS Devices," *IEEE Trans. Ind. Electron.,* vol. 62, no. 3, pp. 1262-1269, 2015.

[2] L. Benini, A. Bogliolo, and G. De Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,* vol. 8, pp. 299-316, 2000.

[3] T. D. Burd, T. A. Pering, and A. J. Stratakos, "A Dynamic Voltage Scaled Microprocessor System," *IEEE J. Solid-State Circuits (JSSC),* vol. 35, no. 11, pp. 1571-1580, 2000.

[4] P. Pillai, and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," In *SOSP ACM Symp. Operating Syst. Principles*, Dec. 2001.

[5] V. Devadas, and H. Aydin, "Real-Time Dynamic Power Management through Device Forbidden Regions," In *Proc. of the 14th IEEE Real Time and Embedded Technology and Applications Symp. (RTAS'08)*, 2008.

[6] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," *Mobile Computing of the International Series in Engineering and Computer Science,* vol. 353, pp. 449-471, 1996.

[7] S. Aminzadeh, and A. Ejlali, "A Comparative Study of System-Level Energy-Management Methods for Fault-Tolerant Hard Real-Time Systems," *IEEE Trans. Comput.,* vol. 60, no. 9, pp. 1288-1299, 2011.

[8] V. Devadas, and H. Aydin, "On the Interplay of Dynamic Voltage Scaling and Dynamic Power Management in Real-Time Embedded Applications," In *Proc. of the 8th ACM Int'l conf. on Embedded software,* 2008.

[9] I. Koren, and C. M. Krishna, Fault-Tolerant Systems, 2007, Morgan Kaufman.

[10] D. Pradhan, Fault Tolerant Computer System Design, Prentice Hall, 1996.

[11] X. Castillo, S. R. McConnel, and D. P. Siewiorek, "Derivation and Calibration of a Transient Error Reliability Model," *IEEE Trans. Comput.,* vol. 31, pp. 658-671, 1982.

[12] R. K. Iyer, D. J. Rossetti, and M. C. Hsueh, "Measurement and Modeling of Computer Reliability as Affected by System Activity," *ACM Trans. Compu. Syst.,* vol. 4, pp. 214-237, 1986.

[13] A. Ejlali, B. M. Al-Hashimi, and P. Eles, "Low-Energy Standby-Sparing for Hard Real-Time Systems," *IEEE Trans. Comput.-Aided Design of Integr. Circuits and Syst.,* vol. 31, pp. 329-342, 2012.

[14] M. A. Haque, H. Aydin, and D. Zhu, "Energy-Aware Standby-Sparing Technique for Periodic Real-Time Applications," In *Proc. of IEEE Int'l Conf. Comput. Design (ICCD'11),*, 2011.

[15] R. Melhem, D. Mosse, and E. Elnozahy, "The Interplay of Power Management and Fault Recovery in Real-Time Systems," *IEEE Trans.Comput.,* vol. 53, pp. 217-231, 2004.

[16] O. S. Unsal, I. Koren, and C. M. Krishna, "Towards Energy-Aware Software-Based Fault Tolerance in Real-time Systems," In *Proc. of the IEEE Int'l Symp. Low Power Electron. and Design (ISLPED),* Aug. 2002.

[17] Y. Zhang, and K. Chakrabarty, "Energy-Aware Adaptive Check Pointing in Embedded Real-Time Systems," In *Proc. of IEEE/ACM Design, Automation and Test in Europe (DATE)*, 2003.

[18] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner, "Razor: Circuit-Level Correction of Timing Errors for Low Power Operation," *IEEE Micro,* vol. 6, pp. 10-20, 2004.

[19] D. Zhu, R. Melhem, and D. Mosse, "The Effects of Energy Management on Reliability in Real-Time Embedded Systems," In *Proc. of IEEE Int'l Conf. Comput. Aided Design (ICCAD)*, 2004.

[20] B. Zhao, H. Aydin, and D. Zhu, "Enhanced Reliability-Aware Power Management Through Shared Recovery Technique," In *Proc. of IEEE Int'l Conf. on Comput. Aided Design (ICCAD)*, 2009.

[21] D. Zhu, and H. Aydin, "Reliability-Aware Energy Management for Periodic Real-Time Tasks," *IEEE Trans. Comput.,* vol. 58, pp. 1382-1397, 2009.

[22] B. Zhao, H. Aydin, and D. Zhu, "Energy Management under General Task-Level Reliability Constraints," In *Proc. of 18th IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS)*, 2012.

[23] M. A. Haque, H. Aydin, and D. Zhu, "Energy-aware Task Replication to Manage Reliability for Periodic Real-Time Applications on Multicore Platforms," In *Proc. Of the Second Int'l Green Computing Conf. (IGCC)*, 2013.

[24] C.L. Liu, and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *J. of ACM,* vol. 20, pp. 46-61, 1973.

[25] M. Khavari Tavana, M. Salehi, and A. Ejlali, "Feedback-Based Energy Management in a Standby-Sparing Scheme for Hard Real-Time Systems," In *Proc. of the 32nd IEEE Real-Time Syst. Symp. (RTSS)*, Vienna, 2011.

[26] R. Sridharan, and R. Mahapatra, "Reliability Aware Power Management for Dual-Processor Real-Time Embedded Systems," In *Proc. of the 47th IEEE/ACM Design Automation Conf. (DAC'10),* 2010.

[27] E. Bini, and G. C. Buttazzo, "Measuring the Performance of Schedulability Tests," *J. of Real-Time Syst.,* vol. 30, pp. 129-154, 2005.