The CSI Journal on Computer Science & Engineering

The CSI Journal on
Computer Science and Engineering
Vol. 15, No. 2, 2018
Pages 31-43
Regular Paper

Energy-Budget-Aware Reliability Management in Multi-Core Embedded Systems with Hybrid Energy Source

Sepideh Safari[1] Mohsen Ansari[1] Mohammad Salehi[2] Alireza Ejlali[1]

[1]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[2]Computer Engineering Department, University of Guilan, Rasht, Iran

Abstract

VLSI technology scaling has resulted in the integration of a larger number of cores in a single chip in successive technology nodes, offering a great potential to realize task-level redundancy for reliability enhancement in safety-critical applications. However, since battery technology no longer advances commensurately with integration density, multi-core platforms may have limited utility in battery-powered embedded systems. In this paper, we propose an energy-budget-aware reliability management (enBudRM) method for multi-core embedded systems featuring hybrid energy source (with renewable and non-renewable energy sources). Our method is composed of two phases. In the offline phase, we only consider battery as the energy source and, according to the available energy-budget and slack time for each execution frame, tasks scheduling and voltage-frequency level are determined such that the tasks timing constraints are met while achieving the given reliability target. To increase the battery lifetime, in the online phase, we exploit released slack time at runtime for further voltage scaling. To compensate for the reliability loss of voltage scaling, we exploit an energy harvester along with the battery to enable executing more task replicas. Our experiments show that our energy budgeting method (the offline phase) compared to other approaches reduces the energy consumption on average by 57% (up to 80%). Also, by using harvester we can achieve up to 45% (on average 35%) battery energy saving, resulting in a higher battery life.

Keywords: Hard real-time embedded systems, Multi-core platforms, Fault tolerance, Dynamic task replication, Energy budgeting.

1. Introduction

With the advance of VLSI technology, in order to improve performance and energy efficiency, multi-core platforms are becoming the mainstream in embedded systems [1], [2], [3], [4]. Multi-core platforms provide opportunities to implement real-time embedded systems with low energy consumption and high reliability requirements [5], [6], [7]. However, scaling VLSI technology aggravates manufacturing process variations, soft error rate, and battery efficiency gap (the growth rate gap between application complexity and battery technology) [8]. Process variations lead to variations in the frequency and leakage power of different cores on a chip or across different chips [9], [10], [11], [62]. Scaling VLSI technology also aggravates reliability issues of on-chip systems, such as soft errors that are transient faults (bit-flips) in the underlying hardware due to high energy particle strikes [7], [12]. Furthermore, since the battery technology is not keeping in pace with integration density, multi-core platforms may have limited utility in battery-powered embedded systems [8], [13]. One way to conquer the process variation-induced performance and power variability is to use multiple voltage-frequency levels, e.g. through exploiting Dynamic Voltage and Frequency Scaling (DVFS) [10], [14], [15]. However, scaling supply voltage down further increases soft error rate, with a resultant significant reduction in system reliability [16], [17]. Beside others, task-level redundancy (e.g. redundant multithreading [18], [19], and replication [20]) is a predominant technique to mitigate soft errors in multi-core processors [21], [22]. However, such techniques may impose significant energy overhead to the system, which has to be carefully taken into account for system design, especially in embedded systems with limited energy sources.

In a nutshell, energy management techniques (e.g., DVFS) decrease system reliability, and reliability management techniques (e.g. task replication) consume extra energy. Besides, in the battery-operated devices, such as portable surveillance systems, the limited energy source will be eventually exhausted, and then, the battery needs to be recharged or replaced before the device stops working. Moreover, in some applications, recharging or replacing the battery is time-consuming or even impossible. In order to extend the lifetime of such systems, energy harvesting methods, e.g. solar and piezoelectric harvesters [23], [24], can be used along with battery [25]. Although the harvesting energy obtained from environmental sources such as solar is unlimited, it is time-variant, i.e. the amount of harvested energy during the operation of the system is not constant and also it is not predictable. Therefore, since hard real-time applications need to guarantee the energy availability during the operation of the system, they necessarily require exploiting battery along with energy harvester.

In this paper, we consider the limitations of battery and energy harvester as well as reliability requirements of embedded systems, and propose an energy-budget-aware reliability management (enBudRM) method for hard real-time embedded systems. This method consists of an offline phase and an online phase. In the offline phase, battery is considered as the sole energy source of the system and based on the battery charge and the required lifetime for the system, the available energy budget is distributed between the applications execution frames. Then, considering the amount of the available energy budget for each frame and the required reliability level, the application tasks are scheduled. In the online phase, to increase the system lifetime we use an energy harvester, aiming at using the battery as less as possible. It means that if the energy provided by the harvester is enough for executing the application, we use the harvester instead of the battery, resulting in an energy saving in the battery. The other reason for using energy harvester at runtime is to provide energy budget for reliability management techniques to compensate the reliability degradation due to applying DVFS. When DVFS is used, due to decreasing the supply voltage, system reliability is degraded. In this case, we should exploit more task replicas to improve the reliability. To do this, the energy harvester is used to enable executing more replicas. The main contributions of this paper are:

- Presenting an energy-budget-aware reliability management (enBudRM) method to meet the given reliability target through determining the level of task replication based on the amount of energy budget assigned to each execution frame.

- Proposing the concept of energy budgeting in a system with a hybrid energy source including a limited energy budget like battery (that may not be recharged or replaced in the operational area) and a rechargeable energy source like solar.

- Considering the effects of process variations on performance and power consumption of different cores in multi-core platforms when mapping and scheduling tasks.

- Demonstrating how energy harvester can help to increase battery lifetime while compensating reliability degradations at runtime through providing additional energy.

2. RELATED WORK

Criteria for evaluating energy harvesting systems are different from that for battery powered systems. Harvesting energy is distinct from battery energy in two ways: i) harvesting energy is an unlimited supply which can allow the system to last forever (if appropriately used), unlike the battery which is a limited resource, ii) availability and measurement of harvesting energy is uncertain while the energy stored in the battery can be known deterministically. Therefore, methods which are used to manage battery energy are not always applicable to energy harvesting systems. In addition, most power management schemes for battery-powered systems only account for the dynamics of the energy consumers (e.g., CPU) but not the dynamics of the energy supply. To reduce energy consumption, battery-powered systems should operate at the lowest performance level, while energy harvesting systems do not need necessarily do this and can provide an enhanced performance depending on the available energy [26].

2.1 Battery-Powered Devices

Some research works such as [17], [27], [28], [29], [30], [31], [32] have focused on energy management in fault-tolerant single-processor real-time embedded systems. Some research works, e.g., [6], [33], [34], [35] studied a standby-sparing hardware redundancy technique to tolerate transient faults while saving energy. These works have not considered multiple faults per task execution. Many previous works in the context of multi-core systems either propose energy reduction management techniques without considering reliability (e.g., [36], [37], [38]) or focus on reliability management without considering energy consumption (e.g., [39], [40], [41]). Recently, research works have been focused on both energy and reliability considerations in multi-core systems. Some works, e.g. [42], [43], [44] have proposed multi-core architectures target low-energy consumption and fault tolerance. These works require hardware modification or redesign, and hence, cannot be used by the current off the-shelf processors, while our proposed technique is general and can be exploited by any multi-core processor that supports DVFS. [45], [46], [47], [48], [49] have proposed energy-management techniques for task-level redundancy in multi-core systems. [45] has proposed both individual-recovery and shared-recovery based reliability aware power management heuristics. [45] and [46] have considered only one faulty execution for each task, while for many applications a high level of reliability can be achieved by tolerating multiple faulty tasks [47], [48], [50], [51]. [47] considers periodic independent real-time tasks and determines the degree of replication (number of replicas) and frequency assignment for each task, as well as task-to-core allocations, in such a way to achieve the target reliability levels with minimum energy consumption. [5] has proposed an N-Modular Redundancy (NMR) technique where without considering variations in tasks software vulnerability, assigns same number of copies to each task. However, in our proposed method by considering

tasks software vulnerability we assign appropriate number of replicas to each task (not the same number of replicas for each task), resulting in a reduced energy consumption.

2.2 Energy Harvesting Devices

Several research works have been carried out in power minimization techniques for energy harvesting systems. Few works such as [54] and [55] have considered task scheduling in the context of non-real-time energy harvesting systems. [56] has proposed an offline DVFS algorithm where it was assumed that harvested energy from the ambient energy source is constant. In this work, the variability of the energy source is ignored which is not the case in real applications. [23] has proposed task scheduling techniques for energy harvesting systems. Also the authors in [57] have proposed lazy scheduling algorithm that executes task as late as possible at full speed, reducing deadline miss rates when compared with the classical earliest deadline first (EDF) algorithm. In this paper, tasks slack time is not exploited for energy savings. In order to utilize the slack times for energy saving, [24] has proposed an energy-harvesting-aware DVFS algorithm which slows down tasks when the harvested energy is not sufficient, otherwise, the tasks are executed at the full speed. This work only considers one task instead of considering all tasks in the ready task queue. [13] uses an adaptive scheduling and DVFS algorithm for real-time energy harvesting systems under timing and energy constraints. To do this, it distributes workload of all tasks evenly over time. [58] has considered a realistic model for the battery charging and discharging and presented a load matching task scheduling algorithm for energy harvesting real-time embedded systems. [59] has proposed an energy management technique in the operating system layer and also has proposed an adaptive task scheduler to maximize quality of service of periodic firm real-time applications. [26] has proposed an adaptive duty cycling algorithm that allows energy harvesting sensor nodes to autonomously adjust their duty cycle according to energy availability in the environment. [25] has considered hard real-time single processor systems with two renewable and non-renewable energy sources. In order to reduce the costs, they present two DVS controllers to minimize the energy attained from the non-renewable energy source.

All above research works target task scheduling and DVFS for energy harvesting real-time systems with a single-core processor. However, recent research works have started to move towards multi-core processor [60], [61]. Also none of the previous works consider reliability constraints and multi-core platforms in their system models. [60] has proposed a task mapping, scheduling and power management method for multi-core real-time embedded systems with energy harvester. This method is based on task utilization and mathematically proves that by allocating the new task to the core with the lowest utilization, the lowest overall energy dissipation can be achieved. How-ever, it has more than 50% deadline miss rate. [61] has proposed an algorithm to reduce the deadline miss rate in [60].

In this paper, we address the use of multi-core platforms to achieve high reliability with low energy overhead for hard real-time embedded systems with hybrid energy source consisting of renewable and non-renewable energy sources (i.e. battery and energy harvester).

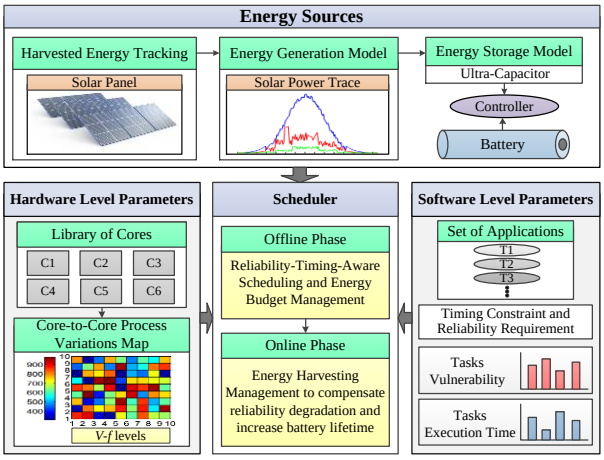


Fig. 1. An overview of the proposed system.

3. SYSTEM MODELS AND PROBLEM FORMULATION

In this paper, we consider a multi-core system featuring homogenous cores where the cores are affected by manufacturing process variations (i.e. the maximum frequency and static power of cores may vary from core to core [62]). In such a multi-core system, due to the variations in operational frequency, an identical task has different execution time and reliability on different cores. Therefore, in our proposed method we consider the effects of process variation in hardware and also software vulnerability. Fig. 1 shows the overview of our system model. Scheduler receives different inputs from hardware and software levels and system energy and gives tasks scheduling in offline and online phases.

3.1. Hardware and Application Model

We focus on a multi-core processor consisting of M homogenous cores $\{C1, C2, \ldots, CM\}$. The cores can operate at multiple voltage and frequency (V-f) levels. Each voltage level contains one or more cores and the cores may have different maximum frequencies and variant static power due to process variations.

A large number of real-time embedded systems operate on a cyclic basis, i.e. they execute certain real-time tasks repetitively (e.g. capturing some sensor data, processing data and finally generating some control signals [25]). These applications are executed in a time frame, i.e. the application tasks should be executed before a deadline. We assume that

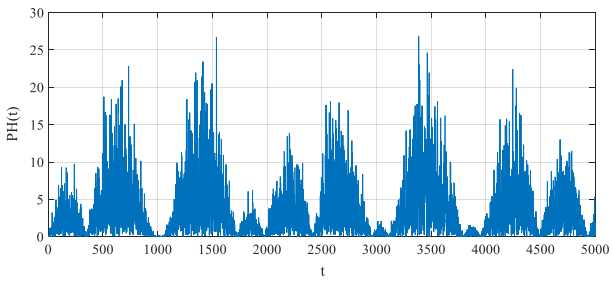


Fig. 2. Solar power trace in continuous scale [57]

each execution frame consists of a set of independent periodic real-time tasks ψ, each task $\tau \in \psi$ has a characteristic triplet (w, d, T). w is the maximum number of CPU clock cycles that the task needs for execution. d is the task deadline and T is the period of the task, in this paper we consider that $d=T$.

3.2. Reliability Models

We consider transient faults, i.e. bit upsets in the underlying hardware. Such transient faults occurrences are typically assumed to follow a Poisson process with the rate λ. The fault rate varies exponentially with the supply voltage V changes. Therefore, the raw fault rate $\lambda(V)$ corresponding to the supply voltage V can be written as follows:

$$\lambda(V) = \lambda_0 10^{\frac{V_{max} - V}{\Delta}} \quad (1)$$

where λ_0 is the fault rate corresponding to the maximum voltage ($V=V_{max}$) and Δ is a parameter that determines the amount of increase in the fault rate when the voltage decreases by one level. In our evaluations, we consider $\lambda_0=10^{-6}$ and $\Delta=1$V [5], [52]. A transient fault in the underlying hardware may finally result in a software failure. To measure the software failures due to transient faults, we use a state-of-the-art software reliability model called the *Function Vulnerability Index* (*FVI*) [52], [53]. This measures the software failure probability and accounts for both spatial and temporal vulnerabilities of different instructions (see details in [53]). Therefore, the software failure rate due to transient faults can be modeled as $\lambda(V)\times FVI$. Therefore, following [52], the reliability of a task execution is computed as:

$$R(\tau) = e^{-\lambda(V)\times FVI\times\frac{w}{f}} \quad (2)$$

When n copies of a task are executed, reliability of the task execution is the probability of that at least one of the task copies finishes successfully and can be written as:

$$R(\tau, n) = 1-(1-R(\tau))^n \quad (3)$$

It should be noted that, due to the Function Vulnerability Index, different applications have different reliability. Therefore, the given reliability target can be achieved through using distinct redundancy techniques. Using dissimilar redundancy techniques reduces the power consumption of less-vulnerable applications.

3.3. Power and Energy Model

We consider that the power consumption of a core at the voltage and frequency level V-f is determined using Eq. 4 [6]; where P_{Static} and $P_{Dynamic}$ are the static power (mainly consumed by sub-threshold leakage current I_{sub}) and dynamic power (mainly consumed due to circuit switching activities), $\alpha_{0\to1}$ is the circuit activity factor, and C_{sw} is the average switched capacitance.

$$P(V,f) = P_{Static} + P_{Dynamic} = I_{sub}V + \alpha_{0\to1}C_{sw}V^2 f \quad (4)$$

When DVFS is used, each task τ is executed at the voltage-frequency level V-f that may be less than V_{max}-f_{max}, and hence, the actual execution time of the task is prolonged from w/f_{max} to w/f (w is the task's clock cycles). Therefore, the total energy which is consumed to execute the task τ and the V-f level can be computed as [6]:

$$E(w,V,f) = P(V,f)\frac{w}{f} \quad (5)$$

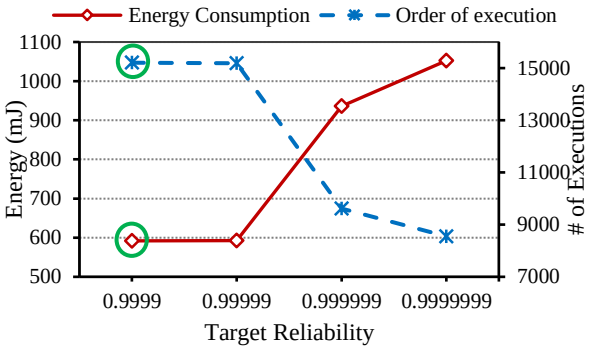


Fig. 3. Trade-off between energy consumption and system life-time (the number of executions).

3.4. Energy Harvesting Model

Energy supply has always been a crucial issue in designing battery-powered systems because the lifetime and utility of the systems are limited by how long the batteries are able to sustain the operation. Since when the system starts running out of the battery power, the validity of data begins to degrade. Therefore, harvesting energy from environment has been proposed to supplement or completely replace battery supplies to enhance system lifetime and to reduce the maintenance cost of replacing batteries periodically [26].

In the rest of this section, at first we present a detailed analysis of different parts of an energy harvesting part in Fig. 1 to illustrate the harvest modules, including the harvested energy tracking block, the energy generation model, and the energy storage model.

Harvested Energy Tracking Block: This block is used to measure the energy received from the harvesting device, such as the solar panel. Such information is useful for determining the energy availability profile and for adapting system performance based on the profiled energy.

Energy Generation Model: This part provides a model of energy availability for the system that may be used for power management techniques. The data measured by the energy tracking block is used by this block to predict future energy availability [26]. Fig. 2 shows an energy generation profile measured by tracking the output current from a solar cell. We can observe that although the energy profile varies from day to day, it follows a general pattern over several days [26]. The trace of the power source $P_H(t)$ is generated by a random number generator as:

$$P_H(t) = \left| 10 \times N(t) \times \cos\left(\frac{t}{70\pi}\right) \times \cos\left(\frac{t}{100\pi}\right) \right| \quad (6)$$

where $N(t)$ is a normally distributed random variable with the mean 0 and variance 1. As Fig. 2 shows, the obtained power trace $P_H(t)$ exhibits stochastic, deterministic and periodic behavior. Since the harvesting energy usually has a stochastic behavior, the systems that exploit such energy sources have to adapt to the dynamic nature of the energy attained from the environment. This obviates the need of appropriate energy management techniques that efficiently use the harvesting energy in order to increase battery lifetime.

Energy Storage Model: This block represents the model for the energy storage technology. Naturally, due to the changes in environmental conditions like temperature, humidity, angle of sunlight incidence and cloud density, the amount of

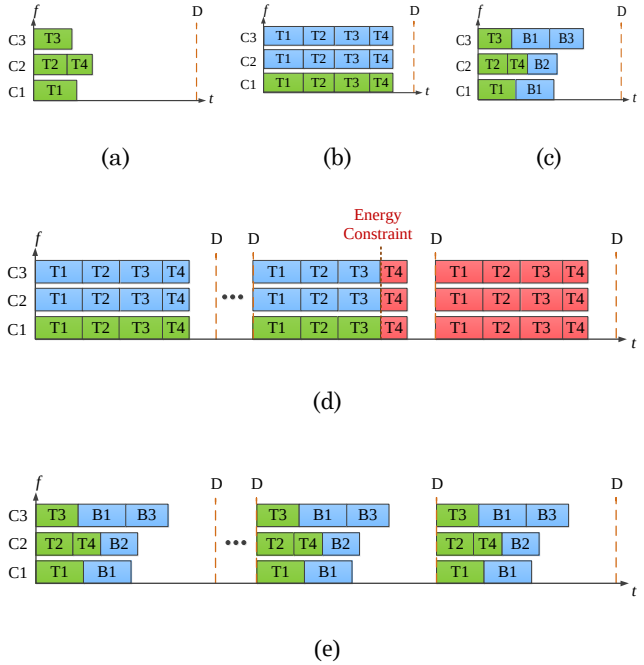


Fig. 4. a) Execution of the sample application in Section 4.2. a) when no fault tolerant technique is used, b) under triple modular redundancy (TMR), c) under an energy budget-aware task replication, d) frame-based execution of Fig. 4b, and e) frame-based execution of Fig. 4c.

harvested power varies over time and solar energy source is unstable in nature. Also, all the generated energy may not be used instantaneously. Therefore, to cope with mentioned problem, the harvesting system will usually have some energy storage technology (e.g., batteries and ultra-capacitors) which can be used to buffer solar energy collected by PV cells [26]. In this paper we assume that the energy storage is ideal, i.e. it is assumed that it can be fully charged and discharged no matter how many charge/discharge cycles it has gone through [13].

Considering $P_H(t)$ as the power harvested from environment by the energy harvesting module at time t, the harvested energy $E_H(t1,t2)$ during time a particular interval $[t1,t2]$ is calculated by Eq. 7. We assume that we have k execution frames in our system and the amount of energy obtained during each frame is:

$$EH_{frame}(t_i, t_{i+1}) = \int_{t_i}^{t_{i+1}} P_H(t)dt \qquad (7)$$

In this paper, the capacity of harvesting storage module is denoted as E_{cap}. Therefore, the amount of energy which can be saved during the harvesting period in the super-capacitor is less than or equal to E_{cap}, i.e.:

$$0 \le \sum_{frame=1}^{k} EH_{frame} \le E_{cap} \qquad (8)$$

In order to ensure that there is no overflow, the initial energy in the super capacitor (EI_{frame}) at the beginning of a frame should be smaller than the battery capacity. To compute the harvester energy consumption in each execution frame we use Eq. 9. In this equation N and B show the number of primary and replica tasks respectively. Also assume that the energy consumption of each task is e_i.

$$ED_{frame} = \sum_{i=1}^{N+B} (x_i + bx_i)e_i \qquad (9)$$

x_i is the main task and bx_i is task replica. Since the energy consumption of the system in each execution frame has to be less than the total available energy, we have:

$$ED_{frame} \le EI_{frame} + EH_{frame} + E_{battery}, \quad 0 \le frame \le k \qquad (10)$$

where $E_{battery}$ is the amount of the energy in battery that is assigned to the frame.

4. MOTIVATIONAL ANALYSIS

4.1. Frame-Based Energy Budgeting

Many frame-based embedded applications are mobile and dependent upon only a limited energy source like battery [56]. Since a system with a limited energy budget should execute the application for a specified period of time, it is important to cautiously consume energy. When task replication is used for fault tolerance, in addition to main tasks, there are task replicas that consume extra energy. When execution frames including task replicas are executed for a specified period of time, in some conditions, the system may exhaust the total available energy before finishing the job. This causes the system to stop working or to fail. Also, due to the limited energy budget, there may not be sufficient energy to provide a high replication level, e.g. full triple modular redundancy (TMR), for all applications. Therefore, the total energy source is distributed among different frames in a way such that each frame has its own energy budget.

By estimating the available energy budget for each execution frame, we guarantee that each frame meets its target reliability according to its own energy budget without exhausting the total energy budget of the system for a long time operation. As an example for energy budgeting in frame-based task execution, let us consider that we have a multi-core platform with a 9000J of energy (e.g. a NiMH battery).

Fig. 3 shows the energy consumption and the number of executions for different target reliability values when task replication is used for fault tolerance. As Fig. 3 shows, when the system requires lower target reliability, it consumes lower energy for executing each frame, since it requires executing fewer number of task replicas. Therefore, the system can perform for a longer duration. For example, when the target reliability is 0.9999 each frame consumes 600mJ of energy, and hence, the system can execute the application for 9000J/600mJ=15000 times (this case is marked in Fig. 3). On the other hand, when the target reliability is high, each frame consumes more energy and it executes the frame for a shorter duration. Therefore, there is a tradeoff between the achievable reliability target and the number of frame executions (i.e. the duration of the system operation) under a given energy budget.

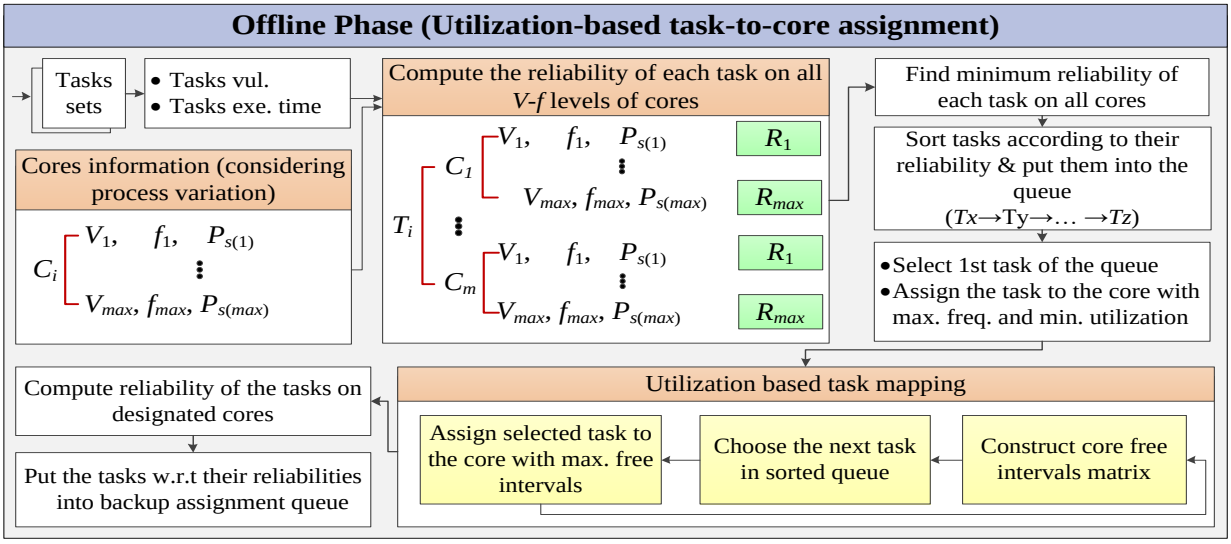


Fig. 5. Utilization-based task-to-core assignment (lines 1-16 of Algorithm 1).

4.2. Illustrative Example

We provide an illustrative example to show how our proposed method works in order to meet target reliability by considering energy budgeting. To do this, we assign specified energy budget to each execution frame. This energy budget should be sufficient for main tasks execution. However, the number of task replicas depends on the remaining amount of energy in the hybrid energy source (i.e. battery and harvester). In this example, we consider a multi-core system with three cores. For simplicity, it is assumed that cores are homogenous and the effects of process variations are not considered. Also, we consider that we have four periodic frame-based tasks T1, T2, T3 and T4 that have the same period and their period is equal to their deadline. We assume that the tasks have reliability in an ascending order as: T1→T2→T3→T4 (i.e. T1 has the lowest reliability and it is more vulnerable to fault occurrence). We consider the same tasks ordering in scheduling.

Fig. 4a shows single-task execution where no fault tolerant technique is used. For this figure we assume that the total energy consumption of tasks is below the frame energy budget. We also assume that the reliability of single task execution do not satisfy the target reliability. One of the approaches to improve the reliability is spatial redundancy (e.g. spatial TMR) which executes three replicas for each task in parallel. We assume that by the use of TMR (Fig. 4b) the target reliability is met but each frame consumes more energy than its assigned energy budget. However, according to Eq. 3, tasks have different reliability due to their various execution time and *FVI* factor. Hence, all tasks in the same frame do not need the same redundancy level (i.e. it is not necessary to use TMR for all tasks). Therefore, in our proposed method when the basic reliability of the system (when no redundancy is used) is below the target reliability, we cautiously determine the number of task replicas in order to meet both the reliability and energy budget constraints. For example in Fig. 4c, we assume that T1 needs two replicas, T2 and T3 need one replica and T4 does not need any replica. Since our proposed method has fewer task replicas in comparison with TMR, it consumes less energy, and also it meets the reliability target. Fig. 4d shows consecutive execution of Fig. 4b where each frame consumes energy more than its energy budget. Hence, it should borrow energy from its successor frames. Repetition of this condition leads the total energy budget to be exhausted. However, our proposed method is energy-budget conservative which assigns replicas to the tasks until the system has enough energy budget and timing constraints are not violated. Therefore, as a result of consecutive execution of frames in Fig. 4e, not only our proposed method meets the frame's energy budget but also each frame saves energy and can give some energy to other frames.

5. OUR PROPOSED METHOD

The problem of scheduling tasks on a multi-core system under optimization goals (achieving target reliability) and constraints (energy-budget and timing constraints) is known as a NP-hard problem [5], [47]. Therefore, we present a heuristic to provide a method for energy reduction. Our proposed enBudRM method consists of an offline phase and an online phase that are developed in Section 5.1 and Section 5.2, respectively.

5.1. Offline Phase of enBudRM

In the offline phase of our proposed method, we assume that the system relies only on the battery as the energy source. Assuming that the system performs a couple of execution frames, we share the energy of the battery among the frames so that each frame has its own energy budget. Then according to the assigned energy budget to each execution frame, we select the appropriate number of replicas and voltage-frequency (*V-f*) level for each task. Afterwards, considering variations in performance and power properties of cores due to process variations and also the cores utilization, the tasks are scheduled. The overview of the system operations in offline phase is shown in Fig. 5 and Fig. 6. The offline phase takes the characteristics of a frame-based task set (ψ) (i.e. tasks clock cycles (w), tasks vulnerability (*FVI*)), cores information (i.e. *V-f* levels and static power considering process variations), energy budget of each frame, target reliability of

Algorithm 1. The Offline Phase of Our enBudRM Method

INPUT: Set of tasks Ψ, tasks clock cycles (w), tasks vulnerability (*FVI*), core-to-core map, set of free cores C_M and their *V-f* levels, reliability and timing constraints, energy_budget constraint (battery)
OUTPUT: Task mapping and *V-f* assignment, determine the number of replicas, and the tasks scheduling.
BEGIN
1. **for** *all* $\tau \in \Psi_Q$ **do** // loop over all ready tasks
2. **for** *all* $c \in C_M$ **do** // loop over all free cores
3. **for** *all available V-f levels for c* **do**
4. $\tau.R=R(\tau,1)$; // Eq. 3
5. **end for;**
6. **end for;**
7. **end for;**
8. $\Psi_Q.sort()$; // sort tasks w.r.t reliability
9. **while** $\Psi_Q \neq \emptyset$ **do**
10. $\tau= \Psi_Q.remove()$; // remove the task with the lowest reliability
11. $c= min_{utilization}\{C_M\}$ **and** $max_{frequency}\{C_M\}$;
12. $\tau.f=c.f_{max}$; // operating frrequency
13. $\tau.k=1$; // the number of replicas
14. $c.add(\tau)$;
15. $\Psi_{BQ}.add(\tau)$; // tasks backup assignment queue
16. **end while;**
17. $E_{total}= \Sigma E(\tau)$; // total energy of tasks in Ψ_{BQ} (Eq. 4)
18. $R_{total}= \Pi R(\tau, \tau.k)\ \forall \tau \in \Psi_{BQ}$; // total reliability of tasks in Ψ_{BQ} (Eq. 2)
19. $\Psi_{BQ}.sort(R)$; // sort tasks w.r.t reliability
20. **while** ($R_{total} < R_{target}$ & $\Psi_{BQ} \neq \emptyset$) **do**
21. $\tau= \Psi_{BQ}.remove()$;
22. $c=min_{utilization}\{C_M\}$;
23. **if** $\tau.w/c.f \leq free\_interval(c)$ **and** $E(\tau) \leq E_{Budget}-E_{total}$ **then**
24. c.add(τ);
25. $\tau.k=\tau.k+1$;
26. $E_{total}= E_{total}+E(\tau)$; // update total energy
27. $R_{total}=\Pi R(\tau, \tau.k)\ \forall \tau \in \Psi_{BQ}$; // update total reliability
28. $\Psi_{BQ}.sort()$; // sort tasks w.r.t reliability
29. **else**
30. return infeasible;
31. **end if;**
32. **end while;**
33. $\Psi_{EQ}.sort(E)$; // sort tasks w.r.t energy consumption
34. **while** $\Psi_{EQ} \neq \emptyset$ **do**
35. $\tau= \Psi_{EQ}.remove()$; // task with max. energy consumption
36. **for** $f=\tau.c.f_{min}$ *to* $\tau.c.f_{max}$ **do**
37. **if** $\tau.w/f - \tau.w/\tau.c.f_{max} \leq free\_interval(\tau.c)$ **then**
38. $\tau.f=f$;
39. $R_{DVFS}= \Pi R(\tau, \tau.k)\ \forall \tau \in \Psi_{BQ}$; // τ is executed under f
40. **if** $R_{DVFS} \geq R_{target}$ **then break**; // break the for loop
41. **else** // schedule a replica
42. $c=min_{utilization}\{C_M\}$;
43. **if** $\tau.w/c.f_{max} \leq free\_interval(c)$ **and** $E(\tau) \leq E_{budget}-E_{total}$ **then**
44. $c.add(\tau)$; // inser a replica
45. $\tau.k=\tau.k+1$;
46. $E_{total}= E_{total}+E(\tau)$; // update total energy
47. $R_{total}=\Pi R(\tau, \tau.k)\ \forall \tau \in \Psi_{BQ}$; // update total reliability
48. **break;** //break the for loop
49. **end if;**
50. **end if;**
51. **end if;**
52. **end for;**
53. **end while;**
54. **shift** the replicas to the end of the frame;
END

the system and timing constraints as input, and gives tasks mapping and scheduling.

Algorithm 1 shows the pseudo-code of the offline phase of our enBudRM method. In this phase a frame-based task set (ψ) is scheduled on a multi-core system. At first, the reliability of the tasks under all *V-f* levels on all cores are determined in lines 1-7. Then, the tasks are sorted in an ascending order according to their minimum reliability in line 8. Through lines 9-16 the algorithm iterates until there is no task in the task set. In each iteration, it chooses the task with the minimum reliability (line 10) and the finds the core with the lowest utilization and the maximum frequency to improve the task's reliability (line 11). It has been mathematically proven that utilization-based task mapping achieves the lowest overall energy consumption [61]. The algorithm assigns the task τ to the selected core c. Then the operating frequency of the task $\tau.f$ is set to the maximum frequency of the core (line 12). We use the variable $\tau.k$ with the initial value of 1 to determine the number of replicas for the task τ (at first the main task has no replicas). Also, in each iteration of the while loop, the algorithm computes the task reliability on the designated core and put it into the backup assignment queue (ψ_{BQ}) in line 15 (the overview of these operations is shown in Fig. 5).

After mapping the tasks to the cores, the total energy consumption of the tasks and the total reliability are computed in lines 17-18. As we mentioned before, we use task replication to meet reliability target. However, when the energy-budget is limited, replicas should be assigned consciously. Therefore, in assigning replicas we consider timing constraints, target reliability and energy-budget simultaneously. To do this, at first the tasks in the backup assignment queue (ψ_{BQ}) are sorted according to their reliability in ascending order in line 19. Then, the algorithm iterates until there is no task in the backup assignment queue and the total reliability becomes higher than the target reliability (lines 20-32). In each iteration of the while loop, we take the task with the minimum reliability from the backup assignment queue for replica assignment (line 21).

In replica assignment, we use the utilization-based policy (line 22) to map the replica to the less-utilized core. Here, if timing and energy-budget constraints are met in line 23, the replica is assigned to the selected core. Also, after each replica assignment in line 24, the variable τ.k increases in line 25 and the total energy consumption and the total reliability are updated in lines 26-27. Since the reliability of the tasks is changed due to the replica assignment, the task is put again into the backup assignment queue and ψ_{BQ} is then sorted according to the tasks reliability in line 28. In each replica assignment, if the timing and energy budget constraints are not met, it means that the algorithm cannot assign a replica to the task with minimum reliability and as a result, the target reliability is not met. Therefore it returns infeasible and terminates in lines 29-30. If the total reliability meets the reliability target, we exploit the available slack times to apply DVFS for energy saving through lines 33-53. At first, the algorithm sorts the tasks with respect to their energy consumption and put them in the energy queue (ψ_{EQ}) in line 33. Then, the algorithm iterates until there is no task in ψ_{EQ} (lines 34-54) and chooses the task with the maximum energy for applying DVFS in line 35. The algorithm applies DVFS beginning from the minimum frequency level (line 36) (i.e. it assigns the minimum frequency to each selected task). Then the algorithm computes the required execution time for applying DVFS, i.e. the difference between the task's execution time under the current frequency step ($\tau.w/f$) and the task's execution time in the maximum frequency step ($\tau.w/\tau.c.f_{max}$) (line 37). If there is enough time for applying DVFS (line 37), the task's new frequency ($\tau.f$) is set to the current frequency.

Since by applying DVFS, the reliability is decreased due to the transient fault rate increase [17], [63], extra replica assignments may be required. If after applying DVFS the total reliability (R_{DVFS}) satisfies the target reliability, the algorithm applies DVFS without inserting more replicas and breaks the for loop and chooses the next task in ψ_{EQ} (lines 40-41). However, after applying DVFS, if the total reliability

Algorithm 2. The Online Phase of Our enBudRM Method

INPUT: Offline scheduling, reliability and timing requirements, cores *V-f* levels, energy budget constraint (battery and harvester), task's offline *V-f* level, *MinPossible_V-f*.
OUTPUT: Tasks scheduling and *V-f* level.
BEGIN //Upon finishing a task or cancelling a replica on a core c
1. $\tau = \Psi_{Exe}.remove(c)$; //select the next task in the schedule of c
2. **if** $\tau.f_{offline} > \tau.f_{MinPossible}$ **then**
3. **if** $\tau.w/f_{MinPssible} - \tau.w/\tau.f_{offline} \leq free\_interval(\tau.c)$ **then**
4. $\tau.f = \tau.f_{MinPossible}$;
5. $R_{runtime} = \Pi R(\tau, \tau.k)\ \forall \tau \in \Psi_{Exe}$; // τ is executed under $\tau.f_{MinPossible}$
6. **if** $R_{runtimr} \leq R_{target}$ **then** // insert a replica
7. $c = min_{utilization}\{C_M\}$;
8. **if** $\tau.w/c.f_{max} < free\_interval(c)$ **and** $E(\tau) \leq E_{Harvester}$ **then**
9. $c.add(\tau)$;
10. **else**
11. $\tau.f = \tau.f_{offline}$;
12. **end if;**
13. **end if;**
14. **end if;**
15. **end if;**
END

decreases below the target reliability, the algorithm assigns a replica to the selected task. In order to do this, it chooses the core with the minimum utilization (c) in line 42. If the timing and energy budget constraints are met (line 43), the algorithm inserts a replica to c and then updates the number of task's replicas, the total energy consumption and the total reliability, and finally breaks the for loop in lines 45-48. However, if the constraints are not met, it scales up the *V-f* level and continues with a higher *V-f* level. At last, when the tasks mapping, scheduling and replica assignment are finished, the algorithm shifts replicas to the end of the frame to reduce possible execution overlaps between the main and replica tasks (line 54). For shifting replicas, at first we should find the length of the replicas which is the difference between the end time of the last replica and the start time of the first replica. Then by computing the shifting length, the new start time of the replicas after shifting is the difference between the deadline and the shifting length. Fig. 6 shows the overview of that part of the offline phase of our proposed method that determines the number of replicas and applies DVFS.

5.2. Online Phase

The overview of the system operations in the online phase is shown in Fig. 7. In this phase we have an adaptive fault tolerance management that gets tasks scheduling (achieved by the offline phase) and the hybrid energy source parameters and determines the number of replicas in runtime along with applying DVFS and DPM. In the online phase, the adaptive fault tolerance management decides how to use the hybrid energy for frame execution to increase battery lifetime. To do this, whenever the amount of the available harvesting energy is enough for task execution, the required energy for the task execution is provided by the harvester which leads to save more energy in the battery.
At run-time, some replica tasks may be dropped due to successful execution of the main tasks, resulting in some dynamic slack times. The system uses the dynamic slack to apply DVFS for saving more energy in the battery. However, DVFS may have negative effects on the system reliability [17], [63], primarily because of the increased transient fault rates at low supply voltage and frequency levels. This can be

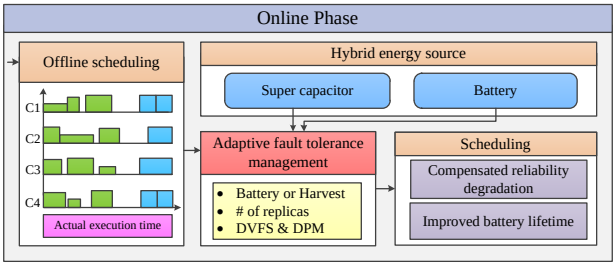


Fig. 7. Overview of the system operations in online phase.

encountered through inserting additional replicas. Allocating additional replicas consumes excessive energy that may violate the energy budget of the frames. The energy that is achieved through energy harvesting can be used for executing additional replicas in the online phase. Therefore, the hybrid energy source helps the system to compensate reliability degradation by executing more replicas. This leads to increase the battery lifetime and to compensate reliability degradation. Algorithm 2 shows the pseudo-code of the online phase of our enBudRM method. In applying DVFS in runtime, due to the computational and timing overheads, we cannot check all frequency levels to choose the best level for energy saving. Therefore, instead of the online phase, the computations are done in the offline phase as follows. In the offline phase, without considering timing and energy budget constraints, we consider assigning one more replica than those that have already been assigned to each task. Accordingly, we determine the minimum possible frequency level that satisfies the reliability requirements. By considering one more replica to each task, a lower frequency level can be used, while the reliability requirement is still satisfied. The minimum frequency values are saved in an array ($f_{MinPossible}$) that is used by Algorithm 2 in the online phase. At runtime, due to early completion of a task or a replica cancellation, some dynamic slack times may release which can be used to apply DVFS for energy saving.
In line 1, the next task after the released slack time in execution queue (ψ_{Exe}) is chosen for applying DVFS. The algorithm finds the minimum possible frequency level of the selected task in $f_{MinPossible}$ array (determined in the offline phase). Here, considering the minimum possible frequency level for the task, we check two cases: *i*) if the offline frequency level of the selected task ($\tau.f_{offline}$) is equal to its minimum possible frequency level ($\tau.f_{MinPossible}$). This means that this frequency level is the minimum value that can be achieved and it is not required to apply DVFS. ii) if the minimum possible frequency level of the task is smaller than the current frequency level. In this case, we can apply DVFS (line 2). For applying DVFS, the algorithm checks the timing constraint. For satisfying the timing constraints, the difference between the task execution time at the minimum frequency level ($\tau.w/f_{MinPssible}$) and its current execution time ($\tau.w/\tau.f_{offline}$) must be more than the core's free time interval (line 3). If so, the algorithm sets the frequency of the task to $f_{MinPssible}$. Then the reliability constraint is checked. If the total reliability Runtime is not violated, it means that DVFS can be applied without inserting more replicas. However, if the total reliability after decreasing the frequency becomes bellow the target reliability, one more replica is assigned. For inserting the replica, the algorithm follows the utilization-based policy and chooses the core with the minimum utilization (c) in line

Table 1: Power and performance characteristics of the LEON3 processor

Voltage and Frequency Level [Volt, MHz]	Power Consumption (mW)		
	Static	Dynamic	Total
[0.72, 490]	13,34	140,62	153,96
[0.85, 650]	14,44	228,78	243,22
[0.97, 730]	15,08	336,81	351,89
[1.10, 850]	15,69	463,17	478,86
[1.23, 970]	16,47	641,39	657,86

7. After checking the timing and energy constraints in line 8, the algorithm inserts one more replica and applies DVFS to save more energy in the battery and updates the total reliability and energy in line 9. Otherwise, we do not apply DVFS since the energy for the extra replica is provided by battery. Indeed, we do not want to use the energy of the battery for inserting one more replica in the online phase. In this case, the frequency level of the task is scaled up to the $f_{offline}$. Finally the output of this phase leads to increase lifetime of battery by saving more energy in the battery.

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1. Experimental Setup

We evaluated our proposed enBudRM method for various benchmark applications executed on a system-level multi-core simulator, which uses precise power and performance characteristics of LEON3 processor [64] obtained through detailed ASIC synthesis. We used the information of synthesizing a LEON3 processor using VHDL implementation in Synopsys Design Compiler with TSMC 45nm low-power standard cell library and junction temperature of 125°C. Also, by considering that the system supports DVFS under five different voltage and frequency levels between [0.72Volt, 490MHz] and [1.23Volt, 970MHz], the power results are shown in Table 1.

We conducted experiments on different real-life applications of the embedded MiBench benchmark suite [65]. Also, we assumed that the *FVI* factor varies between 0.1 to 0.3 [52], and the clock cycles vary between 2K and 20K at the maximum *V-f* level.

Harvester Parameters: We denote the length of a harvesting period as *HP*. Let *frame* be the number of execution frames in a harvesting period. In our implementation, *HP* is one day and the duration of each frame is one second that is the short term prediction. For each frame, the prediction algorithm provides EH_{frame} which is the amount of harvested energy in that frame (Eq. 7). The long term prediction predicts the total harvested energy for the whole day. However, we want the exact information for the next time slot (each one second). Therefore we used short term prediction that predicts the exact amount of harvesting energy for the next one second [59].

Varying Solar Profile: In another set of experiments, we show the performance of our method under different energy harvesting profiles. These profiles are obtained from [66] and show different power profiles on different days at the same location as weather condition changes starting at midnight (Fig. 8).

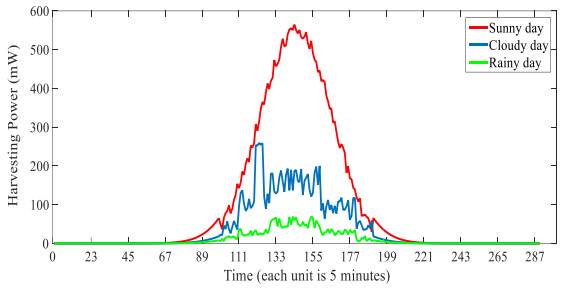


Fig. 8. Varying solar profiles [59]

6.2. Comparison with State-of-the-Art Methods

To the best of our knowledge, this paper is the first one that describes the concept of energy budgeting in hard-real time multi-core systems. We compared our proposed enBudRM method with a TMR technique where tasks are scheduled based on [67] and DVFS is used to achieve energy saving. We compared our enBudRM method with [67] -TMR for: *i*) the worst-case execution condition where all tasks become faulty and no harvester is used which is the offline phase of our proposed method; *ii*) the average-case execution condition including fault-free scenario by considering the energy harvester where the harvester helps the system to improve battery lifetime and compensate reliability degradation. This is the online phase of our proposed method.

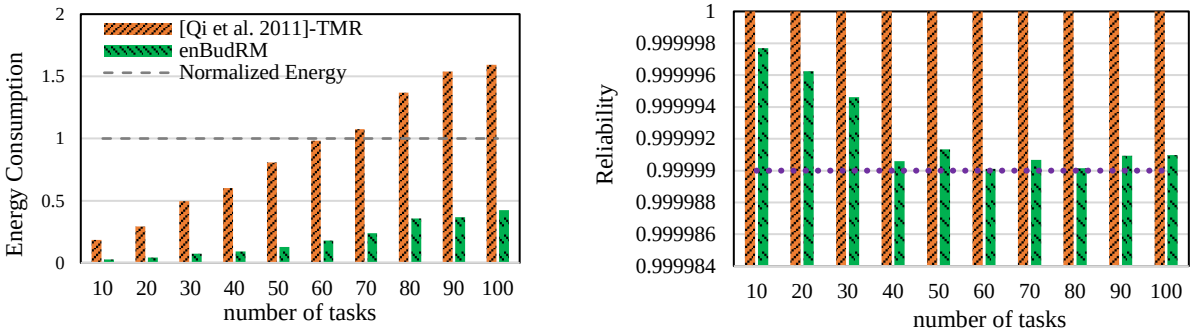
6.2.1. Worst-Case Execution Condition Analysis

In the worst-case execution scenario all replica tasks are considered to be executed as well as the main tasks due to faults in all the main tasks. Although this condition is pessimistic, it leads to consuming the maximum possible energy by the system. Also in this scenario we do not use harvester to verify that the system meets the energy-budget constraint of battery in design time.

Fig. 9 shows the energy consumption and the reliability of our enBudRM method compared to [67]-TMR when the target reliability is 0.99999. Fig. 9a shows the normalized energy consumption of our enBudRM method and [67]-TMR when the execution frame includes 10 tasks to 100 tasks in a multi-core system with four cores. In this case, we assign specified energy-budget to each execution frame that is normalized to 1 (normalized energy is shown by the dash line).

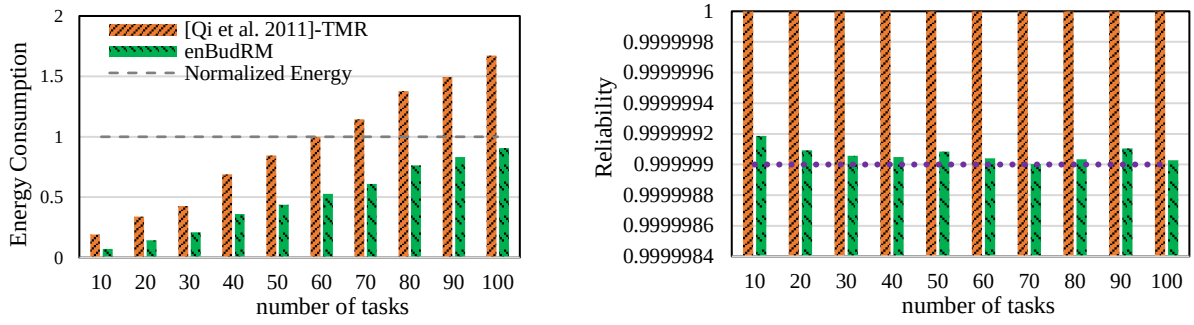
When the system runs fewer number of tasks (under a lower workload), due to our consciously replica assignment, the application needs fewer number of task replicas. However [67]-TMR executes three equal tasks for each task. Therefore, our enBudRM method consumes fewer energy in comparison with [67]-TMR (it meets the energy-budget constraint).

Also, as Fig. 9b shows, our proposed method meets the reliability target that is shown by dot line. When the number of tasks growths, e.g. the system runs 100 tasks (in higher workloads), the enBudRM method needs more replicas to satisfy the reliability target, which leads to consume more energy.



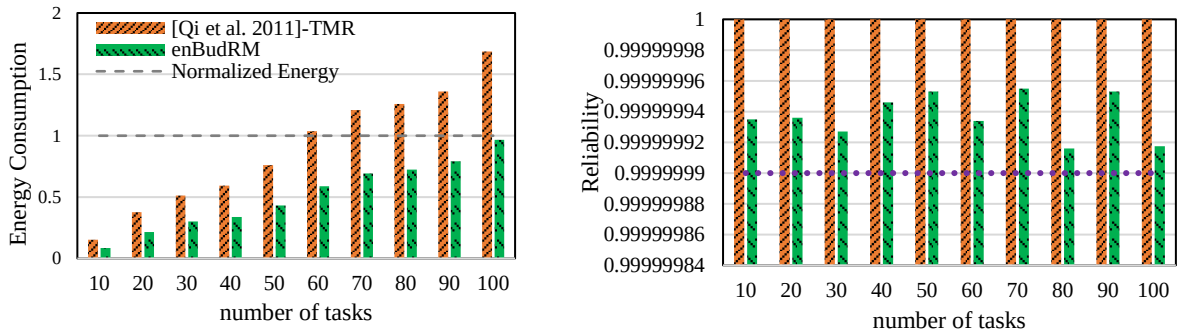
(a) Energy consumption (b) Reliability

Fig. 9. Energy consumption and reliability in a system with *R_target*=0.99999



(a) Energy consumption (b) Reliability

Fig. 10. Energy consumption and reliability in a system with *R_target*=0.999999



(a) Energy consumption (b) Reliability

Fig. 11. Energy consumption and reliability in a system with *R_target*=0.9999999

Although our enBudRM method consumes more energy, but it also meets the energy-budget constraint and reliability target in Fig. 9b. [67]-TMR violates the energy-budget. Considering the energy consumption and reliability results in Fig. 9, we can conclude that as our expectation, our proposed enBudRM method meets the reliability and energy-budget constraints in all condition by consuming fewer energy in comparison with [67]-TMR.

We follow the experiments by considering higher reliability targets, i.e. 0.999999 and 0.9999999 in Fig. 10 and Fig. 11, respectively. In all experiments, our enBudRM method meets the reliability target by consuming lower energy in comparison with [67]-TMR, while [67]-TMR achieves higher reliability level (more than that is needed) which leads to consuming more energy.

6.2.2. Average-Case Execution Condition with Using Harvester

We study the average-case execution condition where both faulty and fault-free execution scenarios were considered. Also in both scenarios we used energy harvester. In our experiments, transient faults were generated by a Poisson process where to simulate transient fault rates in different voltage levels we used the model of Eq. 1 with $\lambda_0=10^{-6}$ and Δ=1V [6]. Since transient faults are rare in nature, the online part of our enBudRM method achieves further energy reduction beyond what is achieved through the offline phase at design time.

We considered that tasks actual execution time vary between their worst-case execution time (*WC*) and the best-case execution time (*BC*). The ratio *BC*/*WC* for each task was generated randomly using the uniform distribution in the range [0.5, 1] to investigate the impact of tasks early completion. Also, the actual execution time of each task is uniformly distributed from *BC* and *WC*.

In order to show the effectiveness of using energy harvester in the online phase, we conducted other experiments. In our experiments we assumed that each execution frame includes 100 tasks and the system has four cores and the target reliability is equal to 0.99999. Fig. 12 shows the energy consumption of a fault-free frame execution (replica cancellation) during one day beginning from midnight in the online phase. In Fig. 12 the energy consumption of offline phase is normalized to one and is shown by dot line. We compared the results of online phase with offline normalized

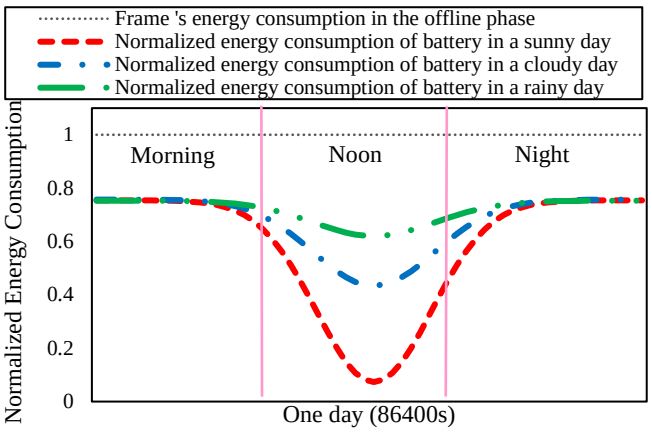


Fig. 12. Energy consumption in fault-free scenario using harvester

energy to show the improvement in battery energy consumption of our proposed method. Fig. 12 shows the normalized energy consumption of frame execution in different conditions including sunny, cloudy and rainy days. In our simulations we used the results of Fig. 8 which shows the varying solar profile in sunny, cloudy and rainy days. By considering the provided energy during a sunny day, as Fig. 12 shows, in the beginning of the day (at midnight) the system receives fewer energy from harvester. In this figure the energy consumption of a frame in online phase is less than the offline phase, that is because of early or successful completion of the main tasks which leads to battery energy saving. However, at noontime, the harvester can provide more energy and as Fig. 12 shows most of the required energy for frame execution can be provided through harvester and the energy consumption of the battery decreases which leads to an increased battery lifetime. Also, at the end of the day, the harvester provides lower energy for the system. Therefore battery should provide most of the energy for frame execution and results follow the same scenario as the first of the day (morning), i.e. lower energy saving in the battery.

We repeated our experiments by harvested energy of cloudy and rainy days. As Fig. 8 shows the received energy in a cloudy or rainy days are less than a sunny day. Therefore, in Fig. 12 the energy saving of battery is less than a sunny day. It means that for example in a cloudy day, the system should receive most of its energy for frame execution through the battery. In conclusion by using harvester in online phase, our enBudRM method provides up to 43% (on average 35%) improvement in battery energy consumption in compared with the offline phase.

7. CONCLUSIONS

In this paper, we presented an energy-budget-aware reliability management (enBudRM) method in a multi-core platform with hybrid energy source consisting of battery and energy harvester. Our proposed method has an offline and online phases. For the offline phase of enBudRM, we explained energy budgeting concept. In the offline phase, at first, based on the total energy in the battery and the required number of frame executions, the energy is distributed among all execution frames so that each frame has its own energy-budget. Then by considering the timing and energy-budget constraints, the appropriate number of replicas, tasks scheduling and *V-f* level of tasks are determined in a way that achieves reliability target. In the online phase we use energy harvester along with battery to increase the battery lifetime and also to provide energy for compensating reliability degradation by inserting more replicas. Our proposed method can increase the battery lifetime by 45%.

References

[1] J. Lee, B. Yun, and K. G. Shin, “Reducing Peak Power Consumption in Multi-Core Systems without Violating Real-Time Constraints,” IEEE Trans. on Parall. and Distr. Syst., vol. 25, no. 4, pp. 1024-1033, April 2014.

[2] A. Munir, S. Ranka, and A. Gordon-Ross, “High-Performance Energy-Efficient Multicore Embedded Computing,” IEEE Trans. on Parall. and Distr. Syst., vol. 23, no. 4, pp. 684-700, 2012.

[3] J. Henkel, V. Narayanan, S. Parameswaran, and J. Teich, “Run-Time Adaption for Highly-Complex Multi-Core Systems,” Int’l Conf. on Hardware/Software Codesign and Syst. Synthesis (CODES+ISSS), pp. 1-8, 2013.

[4] A. Singh, M. Shafique, A. Kumar, and J. Henkel, “Mapping on Multi/Many-Core Systems: Survey of current and emerging trends,” 50th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1-10, 2013.

[5] M. Salehi, A. Ejlali, and B. Al-Hashimi, “Two-Phase Low-Energy N-Modular Redundancy for Hard Real-Time Multi-Core Systems,” IEEE Trans. on Parall. and Distr. Syst., no. 99, 2015.

[6] A. Ejlali, B. Al-Hashimi, and P. Eles, “Low-Energy Standby-Sparing for Hard Real-Time Systems,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 3, pp. 329 - 342, March 2012.

[7] R. Melhem, D. Mosse, and E. Elnozahy, “The Interplay of Power Management and Fault Recovery in Real-Time Systems,” IEEE Trans. on Computers, vol. 53, no. 2, pp. 217-231, Feb 2004.

[8] A. Hemani, “Charting the EDA Roadmap,” IEEE Circuits and Devices Magazine, pp. 5-10, November/December 2004.

[9] K.K. Rangan, M. Powell, G.-Y. Wei, and D. Brooks, “Achieving Uniform Performance and Maximizing Throughput in the Presence of Heterogeneity," IEEE HPCA, pp. 3-14, 2011.

[10] S. Dighe, S. Vangal, P. Aseron, and S. Kumar, “Within-Die Variation-Aware Dynamic-VoltageFrequency-Scaling with Optimal Core Allocation and Thread Hopping,” IEEE J. of Solid-State Circuits, vol. 46, no. 1, pp. 184-193, Jan. 2011.

[11] M.K. Tavana, A. Kulkarni, A. Rahimi, T. Mohsenin, and H. Homayoun, “Energy-Efficient Mapping of Biomedical Applications on Domain-Specific Accelerator under Process Variation,” IEEE ISLPED, 2014.

[12] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique, M. Tahoori, and N. Wehn, “Reliable On-Chip Systems in the Nano-Era: Lesson Learnt and Future Trends,” IEEE DAC, 2013.

[13] S. Liu, Q. Wu, and Q. Qiu, “An Adaptive Scheduling and Voltage/Frequency Selection Algorithm for Real-time Energy Harvesting Systems,” DAC, 2009.

[14] J. Luo and N. K. Jha, “Static and Dynamic Variable Voltage Scheduling Algorithms for Real-Time Heterogeneous Distributed Embedded Systems,” Int.’l Conf. on VLSI Design, 2002 .

[15] M.R. Guthaus, J. S. Ringenberg, and D. Erns, “Power Optimization of Variable-Voltage Core-Based systems,” IEEE Trans. On Computer-Aided Design, 1999.

[16] 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.

[17] D. Zhu, R. Melhem, and D. Mossé, “The Effects of Energy Management on Reliability in Real-Time Embedded Systems,” IEEE/ACM Int. Conf. on Computer Aided Design, 7-11 Nov. 2004.

[18] A. Shye, T. Moseley, V. Janapa Reddi, J. Blomstedt, and D. A. Connors, “Using Process-Level Redundancy to Exploit Multiple Cores for Transient Fault Tolerance,” IEEE DSN, pp. 297– 306, 2007.

[19] S.S. Mukherjee, M. Kontz, and S.K. Reinhardt, “Detailed Design and Evaluation of Redundant Multithreading Alternatives,” IEEE ISCA, 2002.

[20] M. A. Haque, H. Aydin, and D. Zhu, “Energy-aware Task Replication to Manage Reliability for Periodic Real-Time Applications on Multicore Platforms,” Int'l Green Computing Conf. (IGCC), 2013.

[21] E. Normand, “Single Event Upset at Ground Level,” IEEE Trans. on Nuclear Science, vol. 43, no. 6, pp. 2742-2750, Dec. 1996.

[22] T. Langley, R. Koga, and T. Morris, “Single-Event Effects Test Results of 512MB SDRAMs,” IEEE Radiation Effects Data Workshop, pp. 98-101, Jul. 2003.

[23] C. Moser, D. Brunelli, L. Thiele, L. Benini, “Real-Time Scheduling for Energy Harvesting Sensor Nodes,” MICS Scientific Conf. and SNF Panel Review, 2006.

[24] S. Liu, Q. Qiu, and Q. Wu, “Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting,” DATE, 2008.

[25] M. Mohaqeqi, M. Kargahi, and M. Dehghan, “Adaptive Scheduling of Real-Time Systems Cosupplied by Renewable and Nonrenewable Energy Sources,” ACM Trans. on Embedded Computing Syst. (TECS) - Special Section on ESTIMedia, vol. 13, no. 1, Nov. 2013 .

[26] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava and V. Raghunathan, “Adaptive Duty Cycling for Energy Harvesting Systems,” int’l symposium on Low power electronics and design (ISLPED), New York, 2006.

[27] Y. Liu, H. Liang, and K. Wu, “Scheduling for Energy Efficiency and Fault Tolerance in Hard Real-Time Systems,” Design, Automation (DATE), pp. 1444-1449, 2010.

[28] B. Zhao, H. Aydin, D. Zhu, “Enhanced Reliability-Aware Power Management through Shared Recovery Technique,” Int’l Conf. on Computer Aided Design, 2009.

[29] D. Zhu, “Reliability-Aware Dynamic Energy Management in Dependable Embedded Real-Time Systems,” IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 04-07, April 2006.

[30] D. Zhu, and H. Aydin, “Reliability-Aware Energy Management for Periodic Real-Time Tasks,” IEEE Trans. on Computers, vol. 58, no. 10, pp. 1382-1397, 2009.

[31] B. Zhao, H. Aydin, and D. Zhu, “Shared Recovery for Energy Efficiency and Reliability Enhancements in Real-Time Applications with Precedence Constraints,” ACM Trans. on Design Automation of Electronic Syst., vol. 18, no. 2, 2013.

[32] B. Zhao, H. Aydin, and D. Zhu, “Energy Management under General Task-Level Reliability Constraints,” 18th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012.

[33] M. Khavari Tavana, M. Salehi, A. Ejlali, “Feedback-Based Energy Management in a Standby-Sparing Scheme for Hard Real-Time Systems," IEEE Real-Time Syst. Symposium, RTSS, 2011.

[34] M.A. Haque, H. Aydin, and D. Zhu, “Energy-Aware Standby-Sparing Technique for Periodic Real-Time Applications,” IEEE Int’l Conf. on Computer Design (ICCD), pp. 9-12 Oct. 2011.

[35] M.A. Haque, H. Aydin, and D. Zhu, “Energy Management of Standby-Sparing Systems for Fixed-Priority Real-Time Workloads,” Green Computing Conf. (IGCC), pp. 27-29 June 2013.

[36] J. Cong, and K. Gururaj, “Energy Efficient Multiprocessor Task Energy Efficient Multiprocessor Task," Design, Automation and Test in Europe Conf. and Exhibition (DATE), April 2009.

[37] X. Qi, and D. Zhu, “Energy Efficient Block-Partitioned Multicore Processors for Parallel Applications," J. Comput. Science Tech., vol. 26, no. 3, p. 418–433, May 2011.

[38] S. Saha, J. S. Deogun, and Y. Lu, “Adaptive Energy-Efficient Task Partitioning for Heterogeneous Multi-Core Multiprocessor Real-Time Systems," Int’l Conf. High Performance Computing and Simulation (HPCS), July 2012.

[39] H. Su, D. Zhu, and D. Mosse, “Scheduling Algorithms for Elastic Mixed-Criticality Tasks in Multicore Systems,” IEEE 19th Int’l Conf. Embed. Real-Time Computing Syst. and Applications (RTCSA), Aug. 2013.

[40] S.-H. Kang, H. Yang, K. Sungchan, I. Bacivarov, S. Ha, and L. Thiele, “Reliability-Aware Mapping Optimization of Multi-Core Systems with Mixed-Criticality,” Design, Automation and Test in Europe Conf. and Exhibition (DATE), March 2014.

[41] S. Rehman, F. Kriebel, M. Shafique, and J. Henkel, “Reliability Driven Software Transformations for Unreliable Hardware,” IEEE Trans. Comput.-Aid. Des. Integr. Circuits Syst., vol. 33, no. 11, pp. 1597-1610, Nov. 2014.

[42] T. Miller, N. Surapaneni, and R. Teodorescu, “Flexible Error Protection for Energy Efficient Reliable Architectures,” 22nd Int’l Symp. Comput. Arch. and High Performance Comput. (SBAC-PAD), Oct. 2010.

[43] R. Jeyapaul, F. Hong, A. Rhisheekesan, A. Shrivastava, and K. Lee, “UnSync-CMP: Multicore CMP Architecture for Energy-Efficient Soft-Error Reliability,” IEEE Trans. Parall. Distr. Syst., vol. 25, no. 1, pp. 254-263, Jan. 2014.

[44] R. Vadlamani, J. Zhao, W. Burleson, and R. Tessier, “Multicore Soft Error Rate Stabilization Using Adaptive Dual Modular Redundancy,” Design, Automation and Test in Europe Conf. and Exhibition (DATE), March 2010.

[45] Y. Guo, D. Zhu, and H. Aydin, “Reliability-Aware Power Management for Parallel Real-Time Applications with Precedence Constraints,” Int’l Green Computing Conf. and Workshops (IGCC), July 2011.

[46] X. Qi, D. Zhu, and H. Aydin, “Global Scheduling Based Reliability-Aware Power Management for Multiprocessor Real-Time Systems,” J. Real-Time Syst., vol. 47, no. 2, pp. 109-142, March 2011.

[47] M.A. Haque, H. Aydin, and D. Zhu, “Energy-Aware Task Replication to Manage Reliability for Periodic Real-Time Applications on Multicore Platform," Int’l Green Computing Conf. (IGCC), June 2013.

[48] D. Zhu, R. Melhem, and D. Mosse, “Energy Efficient Redundant Configurations for Real-Time Parallel Reliable Servers,” J. Real-Time Syst., vol. 41, no. 3, pp. 195-221, April 2009.

[49] T. Wei, P. Mishra, K. Wu, and H. Liang, “Fixed-Priority Allocation and Scheduling for Energy-Efficient Fault Tolerance in Hard Real-Time Multiprocessor Systems,” IEEE Trans. Parall. Distr. Syst., vol. 19, no. 11, pp. 1511-1526, Nov. 2008.

[50] D. Pradhan, “Fault Tolerant Computer System Design,” Prentice-Hall, 1996.

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

[52] M. Salehi, M. K. Tavana, S. Rehman, F. Kriebel, M. Shafique, A. Ejlali, and J. Henkel, “DRVS: Power-Efficient Reliability Management through Dynamic Redundancy and Voltage Scaling under Variations,” IEEE/ACM Int’l Symposium on Low Power Electronics and Design (ISLPED), pp. 22-24, July, 2015.

[53] S. Rehman. A. Toma, F. Kriebel, M. Shafique, J.-J. Chen, J. Henkel. Reliable Code Generation and Execution on Unreliable Hardware under Joint Functional and Timing Reliability Considerations, In IEEE Real-Time Embed. Tech. App. Symp. (RTAS), pp. 273-282, 2013.

[54] V. Gutnik and A. Chandrakasan, “An Efficient Controller for Variable Supply-Voltage Low Power Processing,” Symposium on VLSI Circuits, pp. 158-159, 1996.

[55] P. Koch, “How to Interface Energy Harvesting Models with Multiprocessor Scheduling Paradigms,” 1st Int’l Conf. on Wireless Communication, 2009.

[56] A. Allavena, and D. Mosse, “Scheduling of Frame-Based Embedded Systems with Rechargeable Batteries,” Workshop on Power Management for Real-time and Embedded Syst., 2001.

[57] C. Moser, D. Brunelli, L. Thiele, and L. Benini, “Lazy Scheduling for Energy-Harvesting Sensor Ndoes," 5th Working Conf. Distr. and Parall. Embedded Syst., 2007.

[58] S. Liu, J. Lu, Q. Wu, and Q. Qiu, “Load-Matching Adaptive Task Scheduling for Energy Efficiency in Energy Harvesting Real-Time Embedded Systems,” 16th ACM/IEEE Int’l Symposium on Low Power Electronics and Design, 2010.

[59] H. Kooti, N. Dang, D. Mishra, and E. Bozorgzadeh “Energy Budget Management for Energy Harvesting Embedded Systems,” IEEE Int’l Conf. on Embedded and Real-Time Computing Syst. and Applications, 2012.

[60] J. Lu, and Q. Qiu, “Scheduling and Mapping of Periodic Tasks on Multi-Core Embedded Systems with eEnergy Harvesting,” Int’l Green Computing Conf. and Workshops (IGCC), 2011.

[61] Y. Xiang and S. Pasricha, “Run-Time Management for Multicore Embedded Systems with Energy Harvesting," IEEE trans. Very Large Scale Integration (VLSI) Syst., vol. 23, no. 12, pp. 2876-2889, December, 2015.

[62] E. Humenay, D. Tarjan, and K. Skadron, "Impact of Process Variation on Multicore Performance Symmetry," Conf. Design, Autom. Test, Apr. 2007.

[63] V. Degalahal, L. Li, V. Narayanan, M. Kandemir, and M.J. Irwin, "Soft Errors Issues in Low-Power Caches," IEEE Trans. Very Large Scale Integr. (VLSI), vol. 13, no. 10, pp. 1157–1166, 2005.

[64] http://www.gaisler.com/doc/leon3_product_sheet.pdf. [Online].

[65] M.R. Guthaus, J. S. Ringenberg, and D. Ernst, "MiBench: A free, Commercially Representative Embedded Benchmark Suite," IEEE Int'l Workshop Workload Characterization (WWC), pp. 3-14, Dec. 2001.

[66] "National Renewable Energy Lab.," Apri 2011. [Online]. Available: http://www.nrel.gov.

[67] X. Qi, D. Zhu, and H. Aydin, "Global scheduling based reliability-aware power management for multiprocessor real-time syst., vol. 47, no. 2, pp. 109–142, March 2011.

[68] J. Liu, P . Chou, N. Bagherzadeh, and F . Kurdahi, "Power-Aware Scheduling under Timing Constraints for Mission-Critical Embedded Systems," 38th Design Automation Conf. (DAC), June, 2001.

Sepideh Safari received the M.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2016. She is currently pursuing her PhD degree with Sharif University of Technology, Tehran, Iran. She is now the member of VLSI Laboratory at the department of computer engineering, Sharif University of Technology. Her research interests include low-power design, multi-/many-core systems, and energy management in fault-tolerant real-time systems.
Email: ssafari@ce.sharif.edu

Mohsen Ansari received the M.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2016. He is currently working toward the PhD degree in computer engineering form Sharif University, Tehran, Iran. He is now the member of Embedded Systems Research Laboratory (ESR-LAB) at the department of computer engineering, Sharif University of Technology. His research interests include low-power design of embedded systems, peak power management in fault-tolerant embedded systems, and multi-/many-core systems with a focus on dependability/reliability.
Email: mansari@ce.sharif.edu

Mohammad Salehi received the PhD degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2016. He is currently an assistant professor of computer engineering at University of Guilan, Rasht, Iran. From 2014 to 2015, he was a visiting researcher in the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany. His research interests include design of low-power, reliable and real-time embedded systems with a focus on dependability and energy efficiency in cyber-physical systems and Internet of Things (IoT).
Email: mohammad.salehi@guilan.ac.ir

Alireza Ejlali received the PhD degree in computer engineering from Sharif University of Technology in, Tehran, Iran, in 2006. He is currently an associate professor of computer engineering at Sharif University of Technology. From 2005 to 2006, he was a visiting researcher in the Electronic Systems Design Group, University of Southampton, Southampton, United Kingdom. In 2006, he joined Sharif University of Technology as a faculty member in the department of computer engineering and from 2011 to 2015 he was the director of Computer Architecture Group in this department. His research interests include low power design, real-time embedded systems, and fault-tolerant embedded systems.
Email: ejlali@sharif.edu

Paper Handling Data:

Submitted: 15.12.2017
Received in revised form: 10.05.2018
Accepted: 05.07.2018
Corresponding author: Dr. Alireza Ejlali,
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran