Simultaneous Management of Peak-Power and Reliability in Heterogeneous Multicore Embedded Systems

Mohsen Ansari, Javad Saber-Latibari, Mostafa Pasandideh, and Alireza Ejlali

Abstract— Analysis of reliability, power, and performance at hardware and software levels due to heterogeneity is a crucial requirement for heterogeneous multicore embedded systems. Escalating power densities have led to thermal issues for heterogeneous multicore embedded systems. This paper proposes a peak-power-aware reliability management scheme to meet power constraints through distributing power density on the whole chip such that reliability targets are satisfied. In this paper, we consider peak power consumption as a system-level power constraint to prevent system failure. To balance the power consumption, we also employ a Dynamic Frequency Scaling (DFS) method to further reduce peak power consumption and satisfy thermal constraints on the chip. We illustrate the benefits of our scheme by comparing it with state-of-the-art schemes, resulting in average in 26.5% less peak power consumption (up to 54.3%).

Index Terms— Power Consumption, Reliability, Embedded Systems, Dynamic Frequency Scaling, Thermal Safe Power, Thermal Design Power.

1 Introduction

With the advance of technology scaling, the integration of multiple cores into a chip naturally makes the system suitable for implementation of fault-tolerance mechanisms [1][2][3][4][5][8]. Since power efficiency is one of the desired properties of embedded systems and the use of fault-tolerance techniques can increase power density on a chip, an efficient solution must be used for power reduction in these systems [1][5][9][15][16][17]. Power-aware task scheduling, Dynamic Voltage and Frequency Scaling (DVFS), and Dynamic Power Management (DPM) are widely exploited for average and peak power reduction in embedded systems [1][5][9][40]. Recently, heterogeneous multicore systems provide an effective solution wherein every core can have an individual voltage but it is costly for implementation [4][6]. Due to the heterogeneity, the worst-case execution time and the power consumption of tasks change according to the task-to-core mapping, presenting a new challenge for average/peak power reduction. There is a tradeoff between a global supply voltage for all cores and an individual voltage per core because a global supply voltage is energy inefficient and an individual voltage is very expensive [6][7]. Therefore, it is better to use platforms with multiple voltage islands in embedded systems. In such platforms, processing cores in an island share the same voltage while different islands can have a different supply voltage. Meanwhile, multicore platforms provide a large opportunity for implementation of fault-tolerance techniques to tolerate transient and permanent faults, but it incurs significant power overheads. Due to the Thermal Design Power (TDP) constraint, it is impossible to simultaneously power on all cores on a chip [3]. TDP is the maximum sustainable power that a chip can dissipate safely. Violating TDP makes some cores automatically restart or significantly reduce their performance to prevent a permanent failure [3]. This may affect the timeliness of the system, and hence, designers face a critical problem with system design in deciding how to use multicore platforms in embedded systems. Recently, a new power budget concept called Thermal Safe Power (TSP) provides safe and efficient power constraint [3]. Therefore, it is necessary to keep the power consumption of the cores below TSP in order to avoid excessive temperature increase [3]. TSP is computed in the offline phase for the worst-case scenarios, or unlike TDP in the online phase for a specific mapping of cores. When core heterogeneity or timing guarantees are involved, TSP can also guide task partitioning and mapping decisions. It should be noted that TDP is the chip-level power constraint and TSP is the core-level power constraint. In order to meet the TDP/TSP constraints, some solutions like heat-sink and chip's cooling are proposed while due to their negative effects on the system reliability these solutions are not used in reliable embedded systems [3]. In this paper, we consider TSP as the core-level power constraint.

Consequently, heterogeneous multicore systems are an appropriate solution for power efficiency and reliability improvement since they have a great potential for reducing the power consumption due to per-island DVFS [4]

• *M. Ansari, J. S. Latibari, M. Pasandideh, and A. Ejlali are with the Department of Computer Engineering, Sharif University of Technology, Tehran 14588, Iran (e-mails: mansari@ce.sharif.edu; jsaber@ce.sharif.edu; smpasandideh@ce.sharif.edu; ejlali@sharif.edu).*

Manuscript received 23 Dec. 2018; revised 13 July 2019; accepted 6 September 2019. Date of publication X Y Z; date of current version X Y Z.
(Corresponding author: Alireza Ejlali.)
Recommended for acceptance by X. X.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. X/Y

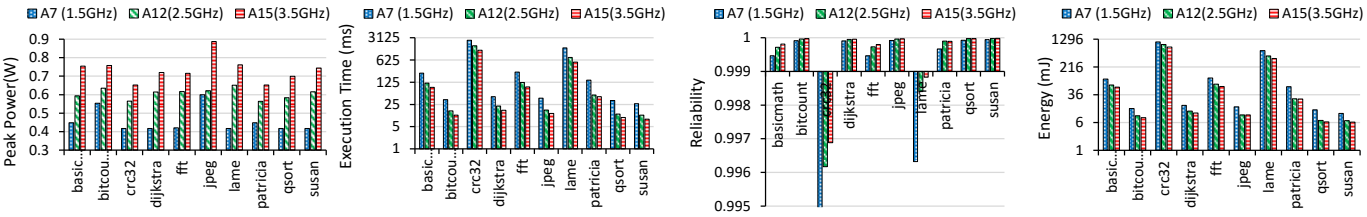


Fig. 1. Peak power consumption, worst-case execution time, reliability, and energy values based on simulations in gem5 [25] and McPAT [26], and measured on ARM Cortex-A7, Cortex-A12, and Cortex-A15, for applications from the MiBench benchmark suite [24].

and improving the system reliability due to intrinsic redundancy in multicore systems. High power consumption can lead to increasing chip temperature, which can aggravate the fault rate [5]. To achieve high reliability, most of the studies had used fault-tolerance techniques [5][9][11]. Task replication is a powerful way to meet reliability targets in multicore embedded systems [1][21][27]. By scheduling multiple copies of the same task on multiple cores, the likelihood of successful execution of at least one of them increases significantly. The advantages of task replication are achieving high reliability and having the potential of tolerating permanent faults in addition to improving reliability in terms of tolerance to transient faults [1][12][14][18][19][20]. The other reason that affects the system reliability is violating the chip TDP. Therefore, the occurrence of peak power must be managed in fault-tolerant embedded systems. Dynamic Power Management (DPM) and Dynamic Voltage Scaling (DVS) are two ways for reducing the instantaneous power consumption and average power consumption, however, the DVS technique brings a number of overhead penalties: increasing instrumentation costs, prolonging the execution time, charging and discharging the capacitances of the chip [5][28], and etc.

In this paper, we minimize the instantaneous power consumption while keeping the system reliability under a given reliability target and meeting tasks timing constraints in Heterogeneous Multicore Embedded Systems. Also, due to the disadvantages of DVS, in this paper, we employ a Dynamic Frequency Scaling (DFS) method that reduces peak power consumption and satisfies thermal constraints. Also, in our proposed method, each island has a separate supply voltage, but each core of the island can operate at a different frequency. Due to the power and performance heterogeneity of the islands, effective management of the islands is an objective for reducing the power consumption and improving the system reliability. This is because the power profile and worst-case execution time of the tasks change according to the task mapping policy. For example, Fig. 1 shows power consumption profile, energy consumption, worst-case execution time and system-level reliability for applications from the MiBench benchmark suite [24], where it can be observed that due to the heterogeneity an application has different characteristics when running on the different types of cores.

Objective: The goal of the paper is to minimize the instantaneous power consumption while keeping the system reliability under a given reliability target and meeting tasks timing constraints in heterogeneous multicore embedded systems. To achieve the mentioned goal, we rely on mapping, scheduling and replicating tasks such that the power constraints are met.

To address the above goal, we make the following contribution:

1) Proposing a peak-power-aware reliability management scheme that distributes the power consumption on the whole chip and determines the number of replicas for each task to keep the system reliability at an acceptable level. Also, the proposed method assigns applications to different islands and maps them to the cores of them based on balancing the utilization among all cores.

2) Employing a dynamic frequency scaling technique to reduce peak power consumption such that timing constraints are met.

Evaluation: To compare our method with state-of-the-art methods, we have used gem5 [25], McPAT [26], HotSpot [29], and TSP [3]. Our experiments show that our proposed method provides up to 54.3% (on average by 26.5%) peak power reduction compared to the other schemes in the worst-case scenario.

Organization: In order to evaluate the effectiveness of the proposed method, we compared our method with state-of-the-art techniques. The rest of this paper is formed as follows. Section 2 reviews related work and section 3 presents our system model. In section 4, we present the details of our solution. The experimental results are shown in section 5 and we conclude the paper in section 6.

2 Related Work

Homogenous multicore systems have been widely studied in both power and reliability aspects. An example of the reliability-aware peak power management for homogenous multicore embedded systems is the work presented in [2]. In this paper, Ansari *et al.* have proposed a method that manages peak power overlaps between concurrently executing tasks such that the system reliability is preserved at an acceptable level while guaranteeing to keep the total power consumption of cores below the chip TDP and the power consumption of each underlying core below the core TDP constraint. It should be noted that the application model in [2] is the task graph and hard real-time, while the model in this paper is the periodic and soft real-time. In the context of real-time systems, this difference in the application model is very fundamental [36][37]. Moreover, the system model in the mentioned paper is the homogenous multicore system, while the system model in this paper is based on an island architec-

ture of heterogeneous multicores. Indeed, this difference in application and system model results in the difference in almost all the other aspects including scheduling policies, energy management, experiments, and experimental setup, etc. As energy-aware reliability management for homogenous multicore systems, several works have been recently presented [9][22][27]. Salehi *et al*. [9] have proposed an N-modular redundancy (NMR) technique with low energy consumption for hard real-time multicore systems. Haque *et al*. [22] have proposed an energy-management technique for a standby-sparing system that executes preemptive fixed-priority real-time tasks. The reference [27] finds the level of task replication, voltage and frequency assignment, and task mapping at design time, in order to achieve a given reliability level while reducing the energy consumption. On the other hand, some related works have addressed the problem of power reduction and performance improvement tradeoff on heterogeneous multicore systems without considering the reliability requirements [3][4][6][7][10]. Pagani *et al*. [3] have presented a new power budget concept, called Thermal Safe Power (TSP), that provides safe power and power density constraints. Khdr *et al*. [4] have focused on improving the overall system performance under a critical power constraint for heterogeneous tiled multicore systems. Pagani *et al*. [6] have presented an efficient method to reduce the total energy consumption while satisfying the timing constraints of real-time tasks in heterogeneous multicore systems. This method consists of the conscious selection of the voltage and frequency levels for each island of the chip. Also, they have proposed a task partitioning strategy that considers the energy consumption of the task executing on different islands and at different frequencies. Pagani *et al*. [7] have presented a solution both for energy minimization and peak power reduction, called Single Voltage Approximation (SVA) scheme, for periodic real-time tasks on heterogeneous multicore systems with a shared supply voltage in a voltage island. Munawar *et al*. [10] have exploited the free cycles for each active core to reduce the peak power.

However, reliability and power management for heterogeneous multicore embedded systems have different issues related to the heterogeneity and hence have more complexity. Since the power, worst-case execution time, energy, and reliability characteristics of applications significantly differ from a core type to another one in heterogeneous multicore embedded systems, we should consider the peak power and reliability management simultaneously. The previous works in the context of multicore systems either consider power reduction on heterogeneous platforms without considering reliability or consider reliability improvement without considering peak power on homogenous platforms.

3 System Models and Assumptions

3.1 Task Model

We consider a set of periodic soft real-time tasks $\psi=\{T_1,\ldots, T_n\}$, which each task T_i has a period P_i, a worst-case execution time wc_i. The j^{th} job of a task T_i (J_{ij}) arrives at time $r_{ij}=(j-1)\times P_i$ and must complete by its deadline $j\times P_i$. Also, the relative deadline D_i of the job J_{ij} is equal to the period P_i. The utilization of the task T_i is defined as wc_i/P_i.

3.2 System and Power Model

The proposed system model is based on an island architecture of heterogeneous multicores consisted of two heterogeneous islands [41][42][43][44], i.e. (1) High Performance Island (HPI), (2) Low Power Island (LPI), where each island has a number of homogeneous processing cores, and the number of cores in HPI and LPI is denoted as N_{HPI} and N_{LPI}, respectively. Therefore, the total number of cores in the system is defined as $N=N_{HPI}+ N_{LPI}$. We consider that each core has an L_1 cache, that each island has an L_2 cache shared between their cores, and that there is an L_3 cache shared between two islands. In each island, all cores share a voltage level. However, due to supporting Dynamic Frequency Scaling, each core may have different frequency level.

The total power consumption of the system consists of static and dynamic power components [1][3][5][31]. The static power (P_{static}) is dominated by the leakage current. Dynamic power ($P_{dynamic}$) is mainly consumed due to system activity.

$$P_{total}(V_i, f_i) = P_{static} + P_{dynamic} = I_0 e^{\frac{-V_{th}}{\eta V_T}} V_i + C_{eff} V_i^2 f_i \quad (1)$$

where C_{eff} is the effective switched capacitance, η is a technology parameter, V_i and f_i are supply voltage and operational frequency, and (a) is the activity factor. Under DVS, the voltage V_i that is used for the execution of the tasks in an island may be less than the maximum voltage V_{max}. We denote the normalized voltage ρ_i as [13]:

$$\rho_i = \frac{V_i}{V_{\max}} \quad (2)$$

Let V_{max} be the maximum voltage corresponding to the maximum frequency f_{max}. Considering the almost linear relationship between voltage and frequency [5], [9], we can write: $\rho_i=V_i/V_{max}=f_i/f_{max}$. Therefore, Eq. 1 can be rewritten as:

$$P_{total}(V, f) = \rho_i (I_0 e^{\frac{-V_{th}}{\eta V_T}} V_{\max}) + \rho_i^3 (\alpha C_L V_{\max}^2 f_{\max}) \quad (3)$$

Under the Dynamic Frequency Scaling (DFS) technique, the frequency f_i that is used for the execution of the tasks may be less than the maximum frequency f_{max}. We denote the normalized frequency β_i as:

$$\beta_i = \frac{f_i}{f_{\max}} \quad (4)$$

Now, Eq. 1 can be rewritten as:

$$P_{total}(V, f) = I_0 e^{\frac{-V_{th}}{\eta V_T}} V_{\max} + \beta_i (\alpha C_L V_{\max}^2 f_{\max}) \quad (5)$$

The relationship between DVS and DFS are determined by the system overheads with the different operation

modes to achieve the power efficient operations, i.e. the difference between DVS and DFS is the tradeoff between the system efficiency and the power efficiency. It should be noted that per-core DVS, due to chip-area cost and power consumption overhead of on-chip controllable power supplies, may not be applicable for multicore embedded systems, e.g. for a single chip with 16 and 32 cores. Moreover, due to heterogeneity, all types of cores should have separate DVS controller, and hence, the overheads of DVS increase. Moreover, since DVFS reduces the level of voltage, the fault rate increases. But exploiting DFS does not have any negative effect on the fault rate.

3.3 Fault Model and Reliability Analysis

In this paper, we consider permanent and transient faults. We consider peak power consumption as a system-level power constraint to prevent system failure (permanent faults). Meanwhile, we consider transient faults that perturb the underlying core without causing permanent damage [5][2]. The average fault rate λ is dependent on the supply voltage whereby decreasing supply voltage V, λ increases exponentially. The average fault rate on the supply voltage V can be expressed as [5]:

$$\lambda(V) = \lambda_0 \times 10^{\frac{V_{\max}-V}{d}} \quad (6)$$

where λ_0 is the transient fault rate at $V_{\max}$ and d determines the sensitivity of the system to voltage scaling. The task failure rate can be expressed by $\lambda(V) \times FVI$, where the Function Vulnerability Index (FVI) is the software's susceptibility to failure due to hardware-level transient faults at instruction-level [5][11]. Therefore, the functional reliability of a task can be written as:

$$R(T_i) = e^{-\lambda(V) \times FVI \times wc_i} \quad (7)$$

where wc_i is the worst-case execution time of T_i. Due to the core heterogeneity of the heterogeneous multicore systems, the task reliability on each core is different from each other. Therefore, when k identical copies of a task i are executed on k different cores, the total reliability of the task is defined as the probability of having at least one successful execution and is calculated as [5]:

$$R_{total}(T_i) = 1 - \prod_{j=1}^{k} (1 - R_j) \quad (8)$$

Generally, the reliability of a system with n tasks running by our proposed method can be calculated as:

$$R_{system} = \prod_{i=1}^{n} R_{total}(T_i) \quad (9)$$

4 PROBLEM DEFINITION AND OUR PROPOSED METHOD

In this paper we aim at distributing the power consumption on heterogeneous multicores platforms such that the system reliability is preserved at an acceptable level [39][40]. From [4] we know that chip-level power budgets are not suitable for heterogeneous multicore platforms, and hence, we must adopt a new power constraint for

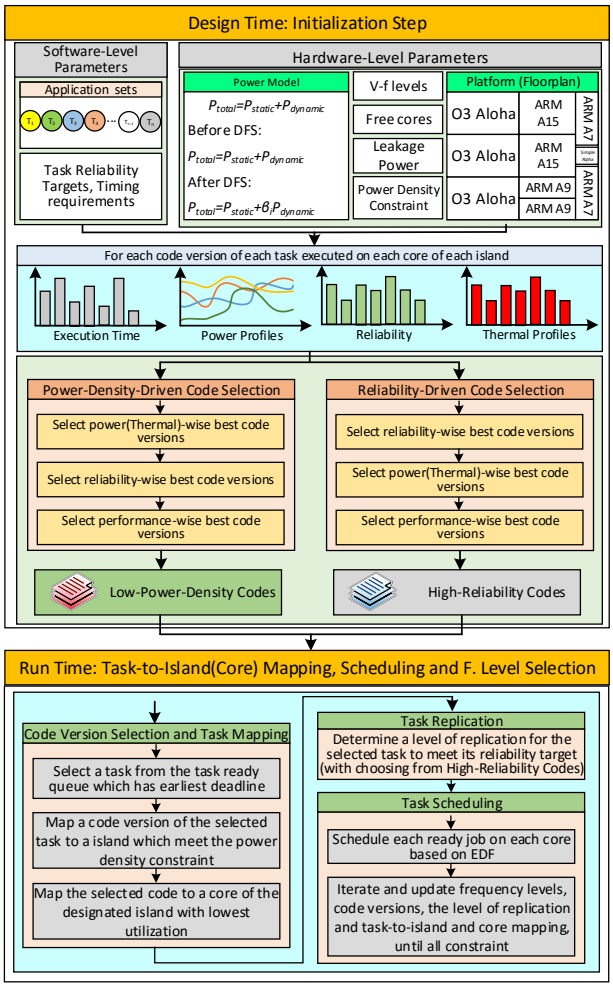


Fig. 2. Overview of our Simultaneous Management of Peak-Power and Reliability flow.

avoiding any thermal violation. Therefore, it is important to propose a method that meets the constraints and considers power constraints on the processing cores and the chip. As discussed in Section 1, executing tasks at a high-frequency level makes lower execution time and fault rate, however, the power consumption at a high-frequency level may exceed the power constraint of the chip. To balance the power consumption, we propose a scheme that performs tasks mapping, tasks scheduling, and frequency level assignment under power constraints on a heterogeneous multicore chip. The overview of the design-time step and run-time of our proposed system are shown in Fig. 2 (To explain the proposed method, see Section 4.2). It should be noted that in Fig. 2 two flows run simultaneously and finally result in the code versions with high-reliability and low-power consumption (i.e. these flows have no dependency on each other). Indeed, we determine and highlight the mentioned code versions, and then in the next step of design flows we exploit them. At design time, for each task, we find a set of code versions that provide high reliability and low power consumption. The high-reliability code version of a task is a code that has the highest functional reliability among all

code versions of the mentioned task (according to Eq. 7). Meanwhile, the low-power-density code version of a task is a code that has lowest peak power and average power consumption among all code versions of the mentioned. In order to calculate the resulting power density of executing different code versions of the task, we run all the code versions of tasks on all the types of cores and make a table with all information (peak power consumption, worst-case execution time, reliability, and energy) before assigning the tasks to the cores. Then, based on the mentioned table, we map and schedule the applications.

4.1 Problem Definition

We use the following notation to represent the system reliability and power consumption, frequency level, and task-to-island mapping, and task-to-core mapping. In this formulation, n is the number of tasks, m is the number of islands, c is the number of free cores in an island j, and v is the number of available frequency levels for each core:

- The task reliability is represented by the matrix $R \in \mathbb{R}^{n \times m \times c \times v}$, in which each element R_{ijkl} denotes the reliability of task i when the task is executed on the core k of the island j under the frequency level l.
- The power consumption is represented by the matrix $P \in \mathbb{R}^{n \times m \times c \times v}$, in which each element P_{ijkl} denotes the power consumption for the task i when the task is executed on the core k of the island j under the frequency level l.
- The task-to-island mapping, task-to-core mapping, and frequency level assignments are represented by the matrix $X \in \{0,1\}^{n \times m \times c \times v}$. The task i is mapped to the core k of the island j and is executed under the frequency level l if and only if $X_{ijkl} = 1$.

The goal of the proposed method is to minimize the instantaneous power consumption while keeping the system reliability under a given reliability target and meeting tasks timing constraints (deadlines). As power-reliability optimization knob, in our system tasks can potentially execute on different islands supporting different frequency levels, and can execute with frequency variations on each core of them, resulting in separate power profiles, reliability and worst-case execution time. We formulate the above problem as a convex problem.

Optimization Goal: Minimize the total average power consumption defined by the sum of the power consumption of all tasks executed on the system.

$$\text{Minimize } P_{system} \tag{10}$$

Chip Power Constraint: The instantaneous power consumption of the chip, i.e. the sum of the power of all underlying cores should be less than the chip TDP constraint in each time slot of the frame of the system.

$$\sum_{i,j,k,l} X_{i,j,k,l} P_{i,j,k,l} \le P_{TDP,chip} \tag{11}$$

Cores Power Constraint: The instantaneous power consumption of each underlying core should be less than the core TSP constraint [3] in each time slot.

$$X_{i,j,k,l} P_{i,j,k,l} \le P_{TSP,k} \tag{12}$$

Tasks Timing Constraint: The worst-case execution time $wc_i/f_{k,l}$ for a task i on the core k and at the frequency level l should not exceed the task timing constraint (defined by the D_i).

$$X_{i,j,k,l} \frac{wc_i}{f_{k,l}} \le D_i \tag{13}$$

Task Reliability Target: For each task, the reliability mechanism should satisfy its reliability requirement R_{req}.

$$\forall i : R_i \ge R_{req} \tag{14}$$

Core Assignment Constraint: Each task can be only mapped to a single core of a single island.

$$\forall i,l : \sum_{j} \sum_{k} X_{i,j,k,l} = 1 \tag{15}$$

Frequency Levels Assignment Constraint: Each task can be only executed under a single frequency level on a core (the frequency level does not change during the task execution).

$$\forall i,j,k : \sum_{l} X_{i,j,k,l} \le 1 \tag{16}$$

Since Eq. 14 is an exponential equation, the mentioned problem is convex [17][38]. The convex formulated problem can be solved by the available convex (CVX) solvers [30], and it is categorized as an NP-Complete problem [8][33][32]. On the other hand, the complexity of such problems may increase exponentially with the increase of problem size at run-time, e.g., with the number of ready tasks, islands, cores, and frequency levels. Accordingly, when the ready tasks and free cores are determined at run-time and a dynamic mapping is required, CVX solvers cannot be employed in online scenarios. Therefore, we propose a heuristic-based algorithm to provide an effective solution for the presented problem.

4.2 Simultaneous Management of Peak-Power and Reliability

In this section, we focus on the task mapping to islands and the task scheduling on the cores. To solve these problems, we propose a heuristic method which maps the tasks to the cores of the islands such that the power constraint, the task-level reliability target, and deadlines are satisfied. Fig. 2 shows the overview of our system flow, while its run-time pseudo-code is presented in Algorithm 1. At design time, we choose the proper code versions for each task between different compiled codes and use them at run-time by Algorithm 1. For this purpose, at first, for each task, the code version with the lowest power density is determined. Then, among the other code versions, we select those that have more reliability than the code version with the less execution time. The selected tasks are defined as Low-Power-Density (LPD) codes. Since the reliability of a single task execution may not satisfy the task reliability target, the use of a task-level redundancy is required. To do this, we select a code version with high reliability to replicate it, improving the task reliability. Then, we select power-density-wise best code versions and, finally, from them, we choose code versions with the lowest execution time. In the task replication step, the selected tasks are defined as High-Reliability (HR) codes.

Algorithm 1. Simultaneous Management of Peak-Power and Reliability

INPUT: ready tasks Ψ with the execution time and the deadline, set of free cores Φ={HPI:{C_1, … , C_{NHPI}}, LPI:{C_1, …, C_{NLPI}}}, code versions for each task, available frequency levels for each core, core power constraint $P_{TSP,core}$ and chip power constraint $P_{TDP,Chip}$.
OUTPUT: task Mapping and task scheduling
BEGIN

1: h=*LCM(the periods of all tasks)*; //Total # of time slots in the frame
2: *PDA* [1...h]={0}; //Initialize the total power consumption array
3: S_HPI_i={*Null*, 1≤i≤N_{HPI}}; //Initialize S with an empty schedule
4: S_LPI_i={*Null*, 1≤i≤ N_{LPI}}; //Initialize S with an empty schedule
5: **while** (Ψ *is not empty*) **do**
6: T_i= Ψ.*remove*(); /Select a task chosen from LPD Codes
7: φ= *find_island*(); //Find the best island
8: C= $\varphi.min_{utilization}$; //Find a core of the island with lowest utilization
9: $C.add(T_i)$;
10: **while** (R_T< R_{req}) **do**
11: $T_i.insert()$; //Insert a replica task chosen from HR Codes to Ψ
12: *Update_reliability*(R_T); // update the task reliability
13: φ= *find_island*(); //Find the island
14: C= $\varphi.min_{utilization}$; //Find a core of the island with lowest utilization
15: $C.add(T_i)$; //Insert a replica task chosen from HR Codes to Ψ
16: **end while**
17: **end while**
18: **while** (Ψ *is not empty*) **do**
19: J_{ij}= Ψ.*remove*(); //Select a job chosen which has highest priority
20: J_{ijl} = {J_{ijl}, 1≤l≤wc_i}; /Partition the selected job into parts
21: k=*release_time*(J_{ij});
22: **foreach** *part* J_{ijl} *starting from the first part* **do**
23: **foreach** *free slot* t=k→ j×P_i *in C.S* **do**
24: **if** *PDA*[t]+ *power* (J_{ijl}) ≤ $P_{TDP,Chip}$ **then**
25: **if** *power* (J_{ijl}) ≤ $P_{TSP,Core}$ **then**
26: $S\_C.add(t, J_{ijl})$;
27: *PDA*[t] = *PDA*[t]+ *power* (J_{ijl});
28: k=t+1;
29: **break**;
30: **end if**;
31: **end if**;
32: **end for**;
33: **end for**;
34: **end while**
35: **if** *not all the jobs are scheduled* **then**
36: **return** *infeasible*;
37: **end if**;
END

EVEN-DFS Function

1: **Function** DFS(J_{ij});
2: $slack$ ← *Extract_Slack*();
3: f_{ij}← max(f_{ee}, $\frac{wc_{ij}}{wc_{ij}+slack}$);
4: Execute J_{ij} at frequency f_{ij};
5: **End Function**

Algorithm Discussion: Algorithm 1 presents the pseudo-code of the run-time step of our proposed method. Algorithm 1 gets the ready tasks with their execution time and deadline, the set of free cores, different code versions for each task, and available frequency levels for each core, the core power constraint $P_{TSP,core}$, and the chip power constraint $P_{TDP,Chip}$. At first, we compute hyperperiod h which is defined as the least common multiple of all task periods (line 1). In line 2, we use a power density array including h slots that determines the power consumption of the system in each time slot. Then, the algorithm initializes two schedules S_HPI_i and S_LPI_i to Null for each core of Φ (Φ is the set of islands which each of islands have multiple cores). In line 5 to 17, the algorithm iterates until all the main and replica tasks are assigned onto a core of an island such that the power constraints are met. For this purpose, we choose a task from its Low-Power-Density code versions which has minimum peak and average power consumption and assign it to an island that its power constraint is not violated. Then, based on the lowest utilization first policy, we map the selected task to a core which has the lowest utilization. Also, in lines 10-16, the algorithm iterates until the reliability of the selected task becomes higher than its reliability target. To do this, the algorithm inserts a replica task into the selected task one after another to satisfy R_{req}. In lines 22 to 34, the algorithm iterates until all the jobs of each task are scheduled based on the PPA-EDF (Peak-Power-Aware Earliest Deadline First) policy [15]. In line 19, the algorithm selects a job which has the earliest deadline and partitions the selected job into parts with different power consumption. In line 21, the variable k is initialized to the release time of the selected job. The algorithm places the parts of the tasks, beginning from the first part, on time slots that come sooner in the schedule *C.S*. In lines 22-25, the algorithm checks free time slots of the core C one after another and places each part J_{ijl} on the first free time slot t such that the power consumption of J_{ijl} does not exceed the core TSP constraint and also does not increase the power consumption of the chip beyond the chip TDP. If these terms are met, T_{ij} is placed in the time slot t of *C.S* in line 26. The power density array *PDA* is updated in line 27 and the variable k is updated in line 28. Finally, if not all the copies are scheduled, the algorithm returns *infeasible* in line 36.

In Algorithm 1, suppose that m is the number of all tasks (the primary and replica tasks), N is the number of free cores, h is the total time slots. The main computation of Algorithm 1 is performed to map and schedule all ready tasks and then putting them into a max-heap. Therefore, for m ready tasks, N free cores and h time slots, building the max-heap is performed in $O(m\times N)$. The first **while** loop iterates for $O(m\times N)$ times. The main **while** loop iterates for $O(m\times N\times h)$ times. Therefore, the order of the algorithm is $max\{O(m\times N), O(m\times N\times h)\}$.

Task Dropping Mechanism: The second part of the problem is to exploit the opportunities created during the actual execution of the tasks at runtime to reduce further power consumption. This can be satisfied through the addition of an online monitor to the system. The responsibility of the online monitor is to control the execution accuracy of different instances of each task, handle the run-time state of the system, and apply DPM and DFS. The online monitor needs to find the first copy of each task that has finished successfully and cancels the execution of the remaining part of the other copies on the other cores. Based on the rare nature of the faults, most of the time, there is no need to execute all copies of a task completely. So, as soon as the first copy of a task finishes suc-

cessfully, the online monitor stops the execution of the remaining part of the other copies, and put the system into a low power state where the dynamic power consumption is avoided; therefore, further power is saved.

Applying our EVEN-DFS technique: As the amount of dynamic slacks is determined at run-time, our method uses DFS to distribute these slacks among the tasks. Therefore, at run-time we determine static and dynamic slacks in each core and apply our DFS technique on them such that the deadlines of the tasks are not violated. To do this, based on the EVEN-DVS technique [34], the frequency of all the tasks is set on each core. Our EVEN-DFS technique distributes slacks evenly among all tasks. The operation of this technique is shown in EVEN-DFS Function. An advantage of the EVEN-DFS technique is its linear time complexity when compared to the quadratic complexity of other DVFS techniques. Also, our DFS technique is software-based while other techniques are hardware-based. It should be noted that the EVEN-DFS function is done after mapping and scheduling all the tasks. Therefore, this function is independently executed after Algorithm 1.

5 RESULTS AND DISCUSSION

In order to evaluate the effectiveness of our proposed scheme, we use gem5 full-system simulator [25], McPAT [26], HotSpot [29], and TSP [3]. We ran our simulations with various task sets including real-life embedded applications of MiBench Benchmark [24] running on a target heterogeneous multicore platform. Since ARM processors are widely used in many embedded systems, we consider a platform with several types of ARM cores [23], e.g. ARM Cortex-A7, Cortex-A12, and Cortex-A15. Meanwhile, we considered that the system supports per core DFS and per island DVFS. The details of simulation configurations for the processing cores of our system are summarized in Table 1. Also, Fig. 3 shows our tool flow for scheduling simulation, and power, thermal, and performance evaluation. As shown in Fig. 3, we extract the peak power consumption, the worst-case execution time, reliability, and energy values based on simulations in gem5 [25] and McPAT [26] measured on ARM processors for applications from the MiBench benchmark suite [24]. Then, we employ a system-level simulator for simulating different execution scenarios. To determine the power constraints, we use the TSP simulator as the core-level power constraint calculator. We use the results of Fig. 1 to simulate our proposed system. To the best of our knowledge, this paper is the first attempt that addresses peak-power management and fault-tolerance in conjunction on heterogeneous multicore embedded systems. Therefore, we compare our peak-power-aware reliability management (PPARM) scheme with state-of-the-art power and reliability management techniques. The comparison schemes are:

- [1]-EM: This technique has proposed a task replication mechanism with low power/energy overhead for multicore systems [1]. Haque *et. al.* [1] have proposed the newest power-aware task replication mechanism for multicore systems. They have developed static solutions for managing power consumption and then have proposed dynamic adaptation schemes in order to reduce the concurrent execution of the replicas of a given task and to take advantage of early completions to further achieve the power reduction.
- TMR: In the Triple Modular Redundancy (TMR) technique, each task has three copies executed on three different cores. Three copies of each task perform majority voting on the results for error masking.
- DMR: In the Dual Modular Redundancy (DMR) technique, each task is executed in the DMR mode with a backup task. This technique makes the DMR mode for applications' tasks according to the tasks' vulnerability [35].

Table 1. The details of simulation configuration

Processor	4, 8, 16, and 32 cores, two islands, five different voltage and frequency levels between [0.85Volt, 1GHz] and [1.1Volt, 2GHz].	
Memory	Main Memory	4GB, 1 channel, 2 ranks, 8 banks per rank, Access time: 100 cycles, DRAM
	L1	32KB, 8KB block-width, 4-way, Access time: 2 cycles, SRAM
	L2	1MB, 16-way, 64B line, Write back, write: 20 cycles, STT-RAM

We compared PPARM with three selected schemes ([1]-EM, DMR, TMR) for: *i*) the worst-case execution scenario when the system consumes the maximum possible power (Section 5.1) and *ii*) the actual-case execution scenario when the system consumes real power (Section 5.2). In the worst-case scenario, all tasks and their replicas are executed, and therefore, the system consumes the maximum possible power that is very pessimistic. In the actual-case scenario, the system needs to find the first copy of each task that has finished successfully and cancels the execution of the remaining part of the other replicas on the other cores. Based on the rare nature of the faults, most of the time, there is no need to

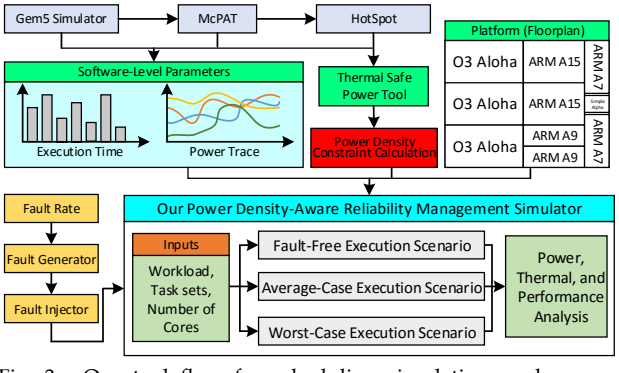
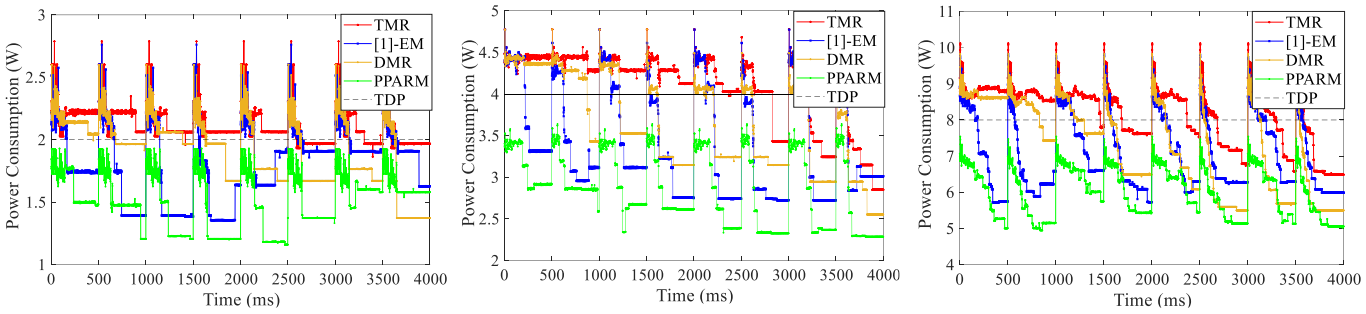


Fig. 3. Our tool flow for scheduling simulation, and power, thermal, and performance evaluation.



(a) $U_{primary\_tasks}$=0.8 , # of cores= 4 (b) $U_{primary\_tasks}$=1.6, # of cores= 8 (c) $U_{primary\_tasks}$=3.2, # of cores= 16

Fig. 4. Power consumption profile in the worst-case scenario on different systems.

execute all replicas of a task completely. So, as soon as the first copy of a task finishes successfully, the system stops the execution of the remaining part of the other replicas, and put the system into a low power state where the dynamic power consumption is avoided; therefore, further power is saved. In this scenario, the system consumes real power (actual power consumption).

5.1 Worst-Case Execution Scenario

The worst-case execution scenario shows the maximum power consumption by the system because all copies of each task are executed in this scenario. Therefore, it can be considered a suitable condition for comparing PPARM and other techniques. Fig. 4 shows the power consumption profile of PPARM and the three mentioned techniques. This figure shows that PPARM consumes less peak power than other schemes. In this figure, the reliability target is equal to 0.99999 and the dashed line is the TDP constraint. As Fig. 4 shows, other techniques miss this TDP constraint while PPARM always meets it. In Fig. 4, we have used only one random task set for each system configuration. It should be noted that PPARM meets the core-level power constraints (obtained from TSP simulator) while other techniques violate TSP. To provide a more detailed analysis, for each system configuration, we used more task sets and then the average results are shown in Fig. 5. Each case of this figure was simulated for 1000 times with different parameters of the applications and the average results are reported. Also, for each configuration of this figure the reliability target for each task is equal to 1-10^{-5}. This figure shows the normalized peak power consumption of the schemes with respect to TDP and the normalized average power consumption of the schemes with respect to our scheme. From Fig. 5 it can be concluded that our scheme completely outperforms the three schemes from the peak and average power consumption viewpoints. Our PPARM scheme provides on average 25.7% (up to 53.9%) and 19.7% (up to 26.5%) peak power and average power reduction as compared to three

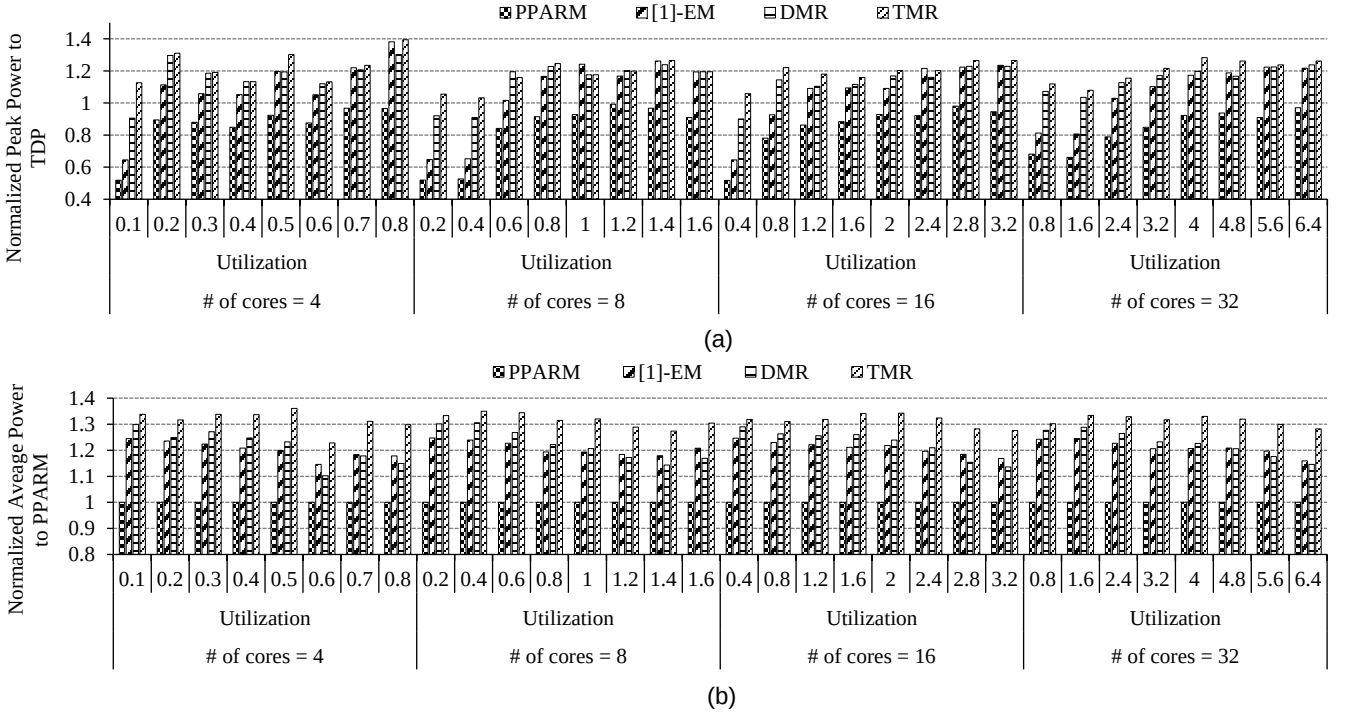
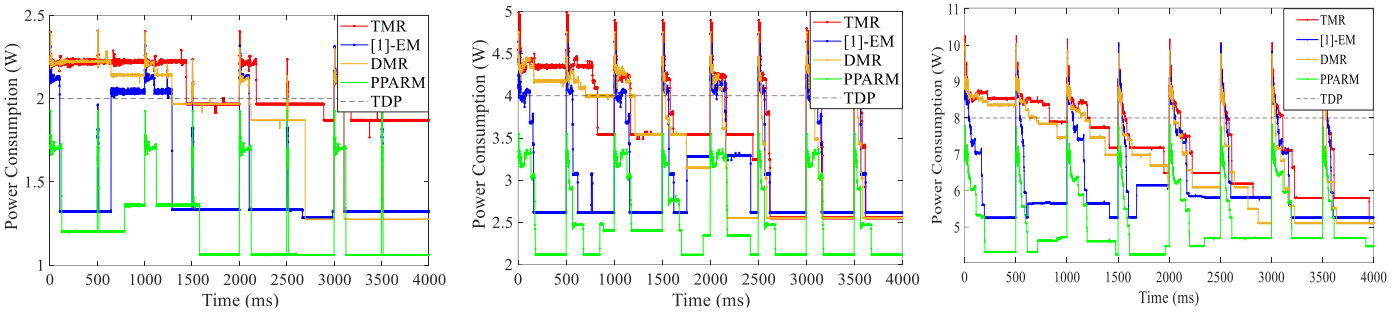


Fig. 5. a) Normalized peak power consumption to TDP, and b) Normalized average power consumption to our scheme in the worst-case scenario with different system configurations.



(a) $U_{primary\_tasks}$=0.8, # of cores= 4 (b) $U_{primary\_tasks}$=1.6, # of cores= 8 (c) $U_{primary\_tasks}$=3.2, # of cores= 16

Fig. 6. Power consumption profile in the actual-case execution scenario on different systems.

mentioned schemes, respectively. When the number of task increases (increasing the system utilization), the peak and average consumption of the system increase. This is because the amount of slack times decreases.

5.2 Actual-Case Execution Scenario

In this case, we evaluate the actual-case execution scenarios. In order to generate fault rate and pattern in our experiments, transient faults were generated using a Poisson process where the fault rate λ corresponding to different voltage levels was modeled using Eq. 6. Therefore, in order to inject faults, we generated a fault vector that determines at which times faults occur. Then, based on the fault vector, we decide which task becomes faulty during the execution of a task set. Since transient faults are rare in nature, PPARM achieves further power reduction at runtime beyond what is achieved through its task dropping mechanism. In the task dropping mechanism, when a copy of task T_i is executed successfully at runtime, we cancel the execution of the remaining parts of the other copies on the other cores. In this condition, the dynamic slack time is released that can be exploited by DFS to reduce the power consumption of the system. Fig. 6 illustrates the power consumption profile of executing task sets that were deployed in Fig. 4 where some tasks are faulty. Like the worst-case execution scenario, in this case, our scheme consumes less power than other schemes due to its different policies and better peak power management scheme.

Fig. 7 shows the peak and average power consumption of PPARM, [1]-EM, DMR, and TMR schemes where the peak and average power consumption have been normalized with respect to TDP and the average power consumption of PPARM, respectively. The experiments show that our PPARM scheme completely outperforms the three mentioned schemes from the peak and average power consumption viewpoints. This is because other schemes either consider power reduction without considering relia-

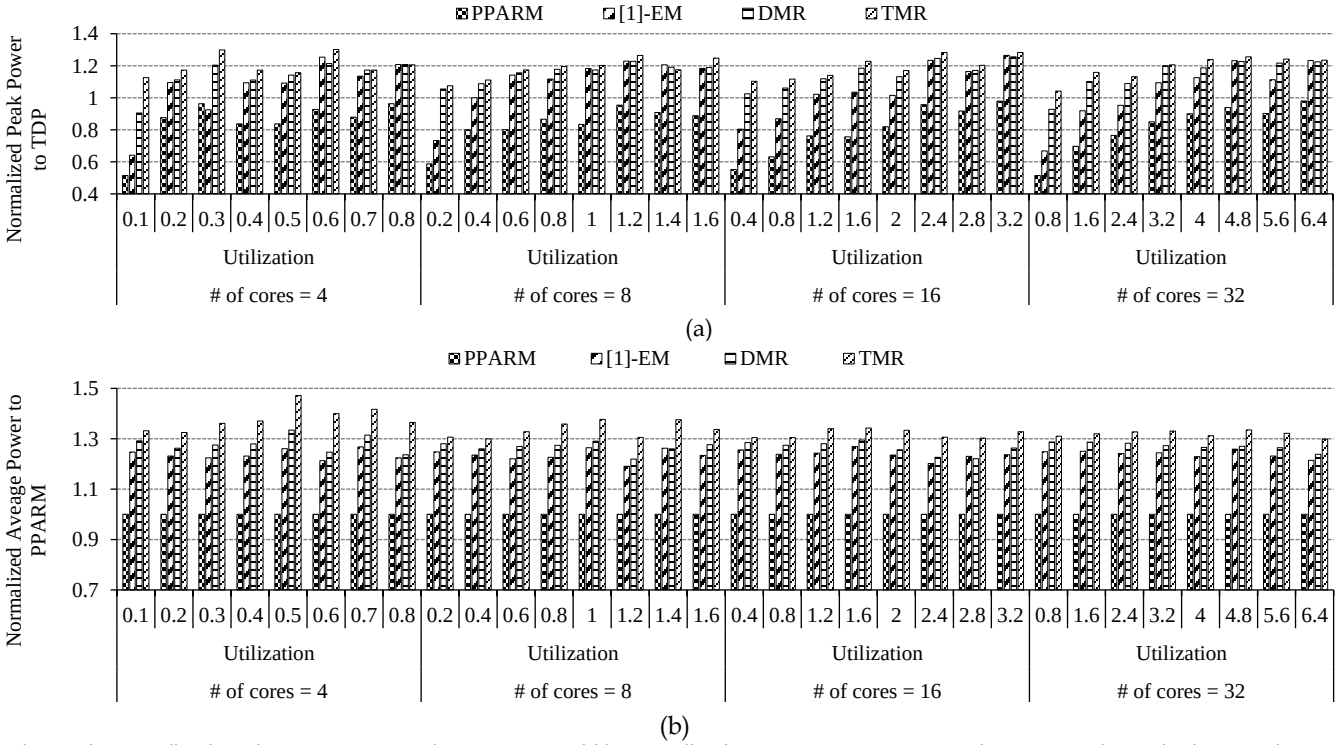


Fig. 7. a) Normalized peak power consumption to TDP, and b) Normalized average power consumption to our scheme in the actual-case execution scenario with different system configurations.

bility or consider reliability improvement without considering peak/average power consumption. PPARM provides on average 27.3% (up to 54.3%) and 21.8% (up to 32.03%) peak power and average power reduction as compared to three mentioned schemes, respectively.

6 Conclusion

In this paper, we have proposed a method to manage the power consumption on the heterogeneous multicore embedded systems. As power is a critical resource for multicore embedded systems, the usage of this resource should be optimized. That is because high power consumption can lead to increasing the chip temperature, which can aggravate the fault rate. In order to manage power consumption and achieve high reliability, we have presented a run-time scheme as our proposed scheme. Our PPARM scheme proposes a peak-power-aware reliability management scheme that distributes the power consumption on the whole chip and determines the number of replicas for each task to satisfy the system reliability target. Also, the proposed method assigns applications to different islands and maps them to the cores of them based on balancing the utilization among all cores. In order to further reduce power consumption, we apply the DFS technique on all the cores of our system. We also illustrated the benefits of PPARM by comparing it with state-of-the-art schemes, resulting in average in 26.5% less peak power consumption (up to 54.3%).

Acknowledgment

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

References

[1] M. A. Haque, H. Aydin and D. Zhu, "On Reliability Management of Energy-Aware Real-Time Systems Through Task Replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 813-825, 2017.

[2] M. Ansari, S. Safari, A. Y. Khaksar, M. Salehi, and A. Ejlali, "Peak Power Management to Meet Thermal Design Power in Fault-Tolerant Embedded Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 1, pp. 161-173, 2019.

[3] S. Pagani, H. Khdr, J. J. Chen, M. Shafique, M. Li and J. Henkel, "Thermal Safe Power (TSP): Efficient Power Budgeting for Heterogeneous Manycore Systems in Dark Silicon," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 147-162, 2017.

[4] H. Khdr, S. Pagani, E. Sousa, V. Lari, A. Pathania, F.Hanning, M. Shafique, J. Teich and J. Henkel "Power Density-Aware Resource Management for Heterogeneous Tiled Multicores," *IEEE Transactions on Computers*, vol. 66, no. 3, pp. 488-501, 2017.

[5] M. Salehi, A. Ejlali and M. Shafique, "Run-Time Adaptive Power-Aware Reliability Management for Many-Cores," *IEEE Design & Test*, no. 99, pp. 1-1, 2017.

[6] S. Pagani, A. Pathania, M. Shafique, J. J. Chen and J. Henkel, "Energy Efficiency for Clustered Heterogeneous Multicores," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 5, pp. 1315-1330, 2017.

[7] S. Pagani, J. J. Chen and J. Henkel, "Energy and Peak Power Efficiency Analysis for the Single Voltage Approximation (SVA) Scheme," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 9, pp. 1415-1428, 2015.

[8] J. Lee, B. Yun and K. G. Shin, "Reducing Peak Power Consumption in Multi-Core Systems without Violating Real-Time Constraints," *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, no. 4, pp. 1024-1033, April 2014.

[9] M. Salehi, A. Ejlali, and B.M. Al-Hashimi, "Two-Phase Low-Energy N-Modular Redundancy for Hard Real-Time Multi-Core Systems," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 25, no. 4, pp. 1024-1033, April 2015.

[10] W. Munawar, H. Khdr, S. Pagani, M. Shafique, J.-J. Chen, and J. Henkel, "Peak Power Management for Scheduling Real-time Tasks on Heterogeneous Many-Core Systems," *in 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS),* Hsinchu, Taiwan, December 2014.

[11] S. Rehman, M. Shafique, F. Kriebel and J. Henkel, "Reliable software for unreliable hardware: Embedded code generation aiming at reliability," *Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 237-246, 2011.

[12] D. Pradhan, *Fault-tolerant computer system design*. Upper Saddle River, N.J.: Prentice Hall PTR, 1996.

[13] P. Pillai and K. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5, p. 89, 2001.

[14] D. Zhu, R. Melhem, and Mosse, "The effects of energy management on reliability in real-time embedded systems," in *Proceedings of Int'l Conf. Computer Aided Design*, pp. 35–40, 2004.

[15] M. Ansari, A. Yeganeh-Khaksar, S. Safari, and A. Ejlali, "Peak-Power-Aware Energy Management for Periodic Real-Time Applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[16] M. Salehi, and A. Ejlali, "A Hardware Platform for Evaluating Low-Energy Multicore Embedded Systems Based on COTS Devices," *IEEE Trans. on Industrial Electronics,* vol. 62, no. 2, pp. 1262-1269, 2015.

[17] M. Ansari, M. Pasandideh, J. Saber-Latibari and A. Ejlali, "Meeting Thermal Safe Power in Fault-Tolerant Heterogeneous Embedded Systems," in *IEEE Embedded Systems Letters*, 2019.

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

[19] B. Zhao, H. Aydin, and D. Zhu, "Enhanced reliability-aware power management through shared recovery technique," in *Proc. int'l conf. on Computer Aided Design (ICCAD)*, 2009.

[20] 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, April 2009.

[21] M.A. Haque, H. Aydin, and D. Zhu, "Energy-Aware Standby-Sparing Technique for Periodic Real-Time Applications," *Proc. IEEE 29th Int'l Conf. Comput. Design (ICCD'11)*, pp. 190-197, Oct. 2011.

[22] M.A. Haque, H. Aydin, and D. Zhu, "Energy Management of Standby-Sparing Systems for Fixed-Priority Real-Time Workloads," *Green Computing Conf. (IGCC)*, Arlington, June 2013.

[23] P. Greenhalgh, "big.LITTLE processing with ARM Cortex-A15 & Cortex-A7," ARM Limited, White Paper, September 2011.

[24] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown, "MiBench: A Free, Commercially Representative Embedded Benchmark Suite," *Proc. Fourth IEEE Ann. Workshop on Workload Characterization*, pp. 3-14, 2001.

[25] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen,

K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*,vol. 39, no. 2, pp. 1–7, May 2011.

[26] S. Li, J.-H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, pp. 469–480, 2009.

[27] F. R. Poursafaei, S. Safari, M. Ansari, M. Salehi, and A. Ejlali, "Offline Replication and Online Energy Management for Hard Real-Time Multicore Systems," *Proc. of the 1th Int'l the CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, Tehran, Iran, 2015.

[28] M. Salehi, M. Khavari Tavana, S. Rehman, M. Shafique, A. Ejlali and J. Henkel, "Two-State Checkpointing for Energy-Efficient Fault Tolerance in Hard Real-Time Systems," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 7, pp. 2426-2437, July 2016.

[29] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," in *IEEE Transactions on VLSI Systems*, vol. 14, no. 5, pp. 501–513, May 2006.

[30] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," Optimization and Engineering 13.1, 1-27, 2012.

[31] M. Ansari, S. Safari, F. R. Poursafaei, M. Salehi, A. Ejlali "AdDQ: Low-Energy Hardware Replication for Real-Time Systems through Adaptive Dual Queue Scheduling," *The CSI Journal on Computer Science and Engineering (JCSE)*, vol. 15, no. 1, pp. 31-38, 2017.

[32] J. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems, *in Handbook of Applied Optimization*, 2002.

[33] M. Salehi *et al.*, "dsReliM: Power-constrained reliability management in Dark-Silicon many-core chips under process variations," in *Proc. Int. Conf. Hardware/Softw. Codesign Syst. Synthesis*, pp. 75-82, 2015.

[34] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, "*System-Level Design Techniques for Energy-Efficient Embedded Systems*," vol. 53, no. 9. Springer Science & Business Media, 2004.

[35] R. Vadlamani, J. Zhao, W. Burleson and R. Tessier, "Multicore soft error rate stabilization using adaptive dual modular redundancy," *Design, Automation & Test in Europe Conference & Exhibition* (*DATE*), pp. 27-32, 2010.

[36] Buttazzo, Giorgio, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications,* New York, NY: Springer, 2011.

[37] G. C. Buttazzo, G. Lipari, L. Abeni, and M. Caccamo "Soft Real-Time Systems: Predictability vs. Efficiency" Springer, 2005.

[38] S. Boyd and L. Vandenberghe, "Convex Optimization", 2004.

[39] L. A. Johnson, "DO-178B: Software considerations in airborne systems and equipment certification," in Radio Technical Commission for Aeronautics (RTCA), 1992.

[40] S. Safari, M. Ansari, G. Ershadi and S. Hessabi, "On the Scheduling of Energy-Aware Fault-Tolerant Mixed-Criticality Multicore Systems with Service Guarantee Exploration," in *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[41] F. Kong, W. Yi, and Q. Deng, "Energy-efficient scheduling of realtime tasks on cluster-based multicores," in *Proceedings of the 14th Design, Automation and Test in Europe (DATE)*, pp. 1–6, 2011.

[42] N. Nikitin and J. Cortadella, "Static task mapping for tiled chip multiprocessors with multiple voltage islands," in *Proceedings of the 25th International Conference on Architecture of Computing Systems (ARCS)*, pp. 50–62, 2012.

[43] S. Pagani, J.-J. Chen, and M. Li, "Energy efficiency on multicore architectures with multiple voltage islands," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 26, no. 6, pp. 1608–1621, June 2015.

[44] X. Wu, Y. Zeng, and J.-J. Han, "Energy-efficient task allocation for VFI-based real-time multi-core systems," in *Proceedings of the International Conference on Information Science and Cloud Computing Companion (ISCC-C)*, pp. 123–128, Dec 2013.

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 at Sharif University, Tehran, Iran, from 2016 until now. 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 embedded systems, and multi-/many-core systems with a focus on dependability/reliability.

Javad Saber-Latibari is currently a M.Sc. student in the Department of Computer Engineering at Sharif University of Technology, Tehran, Iran. He received the B.Sc. degree in computer engineering from Ferdowsi University of Mashhad. His research interest lies in computer architecture, especially in Low Power Design and Embedded Systems.

Mostafa Pasandideh is currently a M.Sc. student in the Department of Computer Engineering at Sharif University of Technology, Tehran, Iran. He received the B.Sc. degree in computer engineering from Shahed University, Tehran, Iran. His research interests include low-power design of embedded systems and Reliability management in multicore embedded systems.

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.