READY: Reliability- and Deadline-Aware Power-Budgeting for Heterogeneous Multi-Core Systems

Javad Saber-Latibari, Mohsen Ansari, Pourya Gohari-Nazari, Sina Yari-Karin, Amir Mahdi Hosseini Monazzah and Alireza Ejlali

***Abstract*— Tackling the dark silicon problem in a heterogeneous multi-core system, the temperature constraints across the system should be addressed carefully by assigning a proper set of tasks to a pool of the heterogeneous cores during the run-time. When such a system is utilized in a reliable/real-time application, the reliability/timing constraints of the application should also be augmented to the temperature constraints and make the tasks mapping problem more and more complex. To solve the mapping problem in such a situation, we propose READY; an online reliability- and deadline-aware mapping and scheduling algorithm for heterogeneous multi-core systems. READY utilizes an adaptive power constraint (as a metric for temperature measurement) that is updated according to the number and position of the active cores on the chip. READY, first, attempts to meet the reliability target of the system by improving the reliability of each task. Then, it performs the mapping and scheduling of the tasks on cores of different islands, so that the peak power and timing constraints are met. The simulation results illustrate that while READY guarantees the timing constraints and meets reliability targets, it improves the peak-power-aware system schedulability (chip performance) by 23.77% (up to 40.69%).**

***Index Terms*— Power constraint, Reliability, Timing constraint, Schedulability, Heterogeneous architectures.**

I. Inroduction

HETEROGENEOUS multi-core processors are a branch of multi-core systems where the architectural heterogeneity and diversity in features of different parts enable digital designers to have more ability in managing and balancing power consumption [1] and reliability of the system. Due to this heterogeneity, different tasks, by running on distinct parts with diverse frequencies and voltages, consume different average power. These variations in system characteristics can be used to achieve the application's desired goals including power, energy consumption, and reliability. Real-time applications are one of the important domains that benefit from heterogeneous architectures [4]. In real-time applications, any violation of constraints will cause critical conditions and systems failure [4][9][46]. For this reason, these applications should have high reliability, which is achieved through fault tolerance techniques [37].

Tasks replication is one of the effective approaches to achieve high reliability in multi-core systems [4][10][11][12]. In this approach, by scheduling the task replicas on different cores, the probability that at least one of them runs correctly will increase, and therefore the reliability of the system will improve. While this approach theoretically increases the reliability of the task, it can impose another reliability challenge to the design, which is thermal violation due to activating more and more cores across the multi-core chips to run the replica tasks [6][7][11][12]. As a result, it is critical to managing the temperature and peak power of the fault-tolerant real-time systems [6][7][19].

Generally, in today's multi-core systems, the size of the processing cores relative to the operating voltage is disproportionately shrinking [13][24][31][32][35]. This process increases the power density of the chips, which causes excessive temperature rise on the chip. As a result, a part of the chip must be inactive so that the system operates within a safe temperature range, which is called dark silicon [14][33]. For real-time systems, designers consider an upper limit of power consumption to ensure that the temperature on the chip is within the safe range [6][7][12]. Thermal Design Power (TDP) is an estimate of the upper limit of power consumption that is considered at system design time [6][7][12]. This power constraint has been widely used in designing systems [6][7][15]. However, TDP is a conservative estimate, with the assumption that the chip works at the worst level of voltage and frequency, and workload [2][16]. Recently, a more efficient core-level power constraint than TDP has been introduced, which is called Thermal Safe Power (TSP). TSP is dynamically calculated and is a function of the number of active cores and their locations [8]. In order to map an application on a core, this power constraint is determined by considering the interactive temperature effect of the active and inactive cores around the target core [8]. The work [6] is the closest research to our work. The authors in [6] have considered similar constraints and targets in their work, but by disregarding to the heterogeneity of the system and exploiting a pessimistic power constraint, reduced system schedulability.

Motivational Example: Here we will see how READY outperforms state-of-the-art approaches in dealing with reliability and power consumption constraints. To this end, we consider a graph-based application which is received as input. For this application, the execution of its tasks has data dependency with each other. Since we consider heterogeneous multi-core processor with two different types of islands (each of them consists of 3 homogeneous cores), each task has a different worst-case execution time when running on different islands. Fig. 1 depicts the task graph (equipped with the deadline and the duration of each task) of

Manuscript received Jan 31, 2020; revised May 4, 2020 and June 9, 2020; accepted June 12, 2020. Date of publication M D, Y; date of current version M D, Y. This paper was recommended by Associate Editor Z. Shao. (*Corresponding author: Alireza Ejlali*)

- J. Saber-Latibari, M. Ansari, P. Gohari-Nazari, S. Yari-Karin, and A. Ejlali are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. (E-mails: jsaber@ce.sharif.edu, mansari@ce.sharif.edu, gohary@ce.sharif.edu, sinayari@ce.sharif.edu, and ejlali@sharif.edu)
- A. M. H. Monazzah is with the School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran and with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. (Email: monazzah@iust.ac.ir)

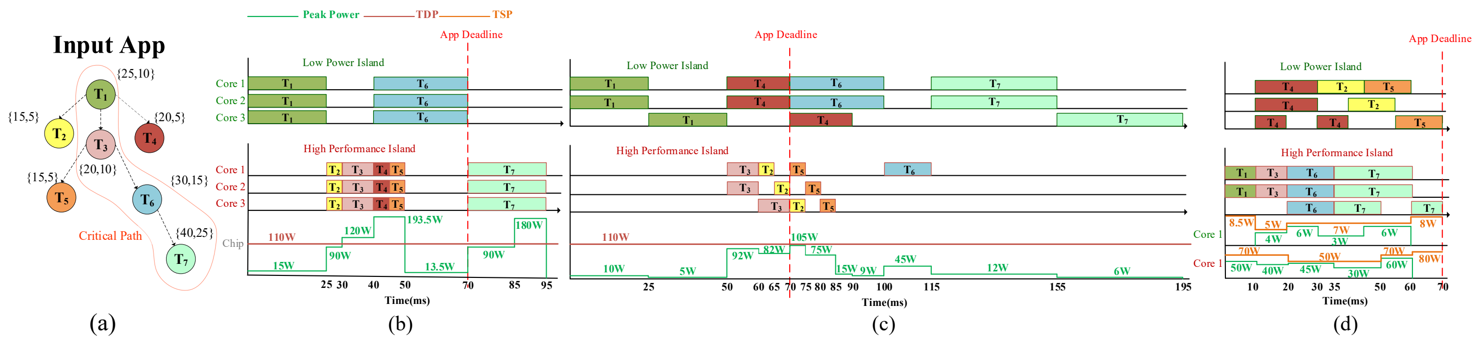


Fig. 1. Motivational example. a) Task graph of a graph-based applications (task graph model), b) TMR mapping and scheduling [30], c) TP3M mapping and scheduling [6], and d) READY mapping and scheduling.

the considered application (Fig. 1a) and the three different mapping and scheduling scenarios for the tasks in a heterogeneous multi-core processor. For the sake of comparison, we considered two other mapping and scheduling approaches besides READY in this example, one is the traditional triple modular redundancy (TMR) which is depicted in Fig. 1b [30] and the other is a state-of-the-art approach i.e., two-phase peak-power management (TP3M) [6] which is illustrated in Fig. 1c. Furthermore, the mapping and scheduling Gantt chart of the READY approach can be seen in Fig. 1d. According to Fig. 1b, the traditional TMR scheduling approach first attempts to increase the reliability by considering three replicas for each task, i.e. the parallel execution of the tasks with TMR technique. It then maps and schedules the tasks by considering the load of each core regardless of the type of cores and the data dependencies between the tasks. Because of the lack of attention to the priority of tasks, core heterogeneity, and peak power consumption, this approach violates application deadline at time 70ms and also TDP constraint on the chip at the time slot [30ms to 50ms] and [85ms to 95ms]. Considering Fig. 1.c, TP3M method provides three replicas for each task to improve the reliability of the system. The overall goal of TP3M is peak power-aware reliability management. This approach assigns tasks to less-loaded cores, regardless of system heterogeneity. Its scheduling goal is to reduce the overlaps between the tasks with the highest peak power consumption while keeping the maximum power consumption below the chip TDP constraint. Also, at first it executes two replicas of each task in parallel and then schedules the third replica because when no fault occurs, the third replica of the tasks is not required. This policy drastically reduces system schedulability due to the disregard of system heterogeneity and imposing the pessimistic TDP constraint and violates the deadline of the application, e.g. at the time 70ms. Now let's see how READY resolves the previous challenges in mapping and scheduling the soft real-time tasks on a heterogeneous multi-core processor. Fig. 1d illustrates that READY, by considering system reliability target, provides a different number of replicas for each task to prevent extra execution of replicas for the tasks. In order to improve the schedulability and performance, READY employs TSP constraint as an adaptive power constraint on each core. TSP is dynamically calculated and is a function of the number of active cores and their locations [8]. In order to meet the application deadline, READY determines the tasks that are on the critical path and then they are assigned to the higher performance cores along with their replicas. Then, the remaining application's tasks and their replicas are assigned to cores that are capable of meeting deadline and peak power constraint with the lowest TSP. In order to schedule the tasks, READY first checks the TSP constraint of the designated core. If the selected task meets the TSP constraint, the task can be scheduled, otherwise, the execution of task shifts to the next time slots. This shifting increases the probability of meeting the deadline because if we shift the whole of the mentioned task to the next time slots, we should re-schedule its other data dependent tasks. Therefore, in order to increase the probability of deadline meeting and utilize the core efficiently, we use this shift. Indeed, READY decides about the scheduling each part of a task individually. Accordingly, this example demonstrates that READY has been able to increase tasks schedulability (system performance) by adhering to deadlines, power constraints, and system reliability target. In the following subsection, we will introduce how we can apply READY to its target system.

In this paper, we present READY; a mapping and scheduling algorithm by exploiting the features of heterogeneous multi-core systems and considering the power constraints that are calculated according to the state of the chip. Our proposed method improves the reliability of applications by exploiting fault tolerance techniques. The mentioned reliability improvement imposes more computational workload to the system, which increases the temperature of the chip. As a result, READY employs adaptive power constraints at the core-level to manage the temperature of the system. Furthermore, power constraints are updated during run-time to keep the performance of the system at an acceptable level.

In summary, READY's contributions in this work are as follows:

- Proposing a reliability improvement method for graph-based applications that inserts the different required number of replicas for each task such that a specific system reliability target is met.
- Presenting a mapping and scheduling algorithm for task graph model with respect to power constraints in heterogeneous multi-core systems.
- Providing a dynamic frequency scaling method based on graph-based applications (Sub-Frame-Based DFS) for heterogeneous multi-core systems.

We evaluated READY by exploiting Gem5 [26], McPAT [27], HotSpot [28], and TSP [8] simulators. We used a set of applications from PARSEC Benchmark [29]. Our simulation results show while READY guarantees the timing constraints and preserves reliability target it improves the peak-power-aware system schedulability (chip performance) by 23.77 percent (up to 40.69 percent).

The rest of this paper is organized as follow. In section II we will explore the related work. Section III explains the preliminaries on the system model. In section IV we will introduce READY in detail. Section V presents our evaluation system setup and results. Finally, in section VI we conclude the paper.

II. Related Work

Generally, researches related to the power-budgeting and energy management in heterogeneous multi-cores real-time systems which take into account reliability are divided into two categories: (*i*) Power/energy management in heterogeneous multi-core systems, regardless of the reliability, (*ii*) Reliability-aware power management in homogeneous multi-core systems.

Several works have been carried out on heterogeneous multi-core processors that focus on the voltage/frequency scaling method applied to each core [17]. However, very few works have been done to reduce energy in heterogeneous multi-chip processors with the VFIS model (the cores are homogeneous on the island, but the islands can differ in terms of the number and type of cores with each other) [1][2][3][18][36][38][44]. Muthukaruppan et al. in [1] seek to observe performance constraints along with reduced power consumption. It should be noted that this study did not consider real-time constraints. In addition, diminishing power does not always mean that we have reduced energy consumption [2][6][7]. One of the works done in the field of energy reduction in heterogeneous processors is the work [18]. This work attempts to balance overall utilization on each island by exploiting Equally-Worst-Fit-Decreasing (EWFD) algorithm. Furthermore, [2] provides a mapping algorithm by using task partitioning and considering the energy factor for each task, which aims to reduce the energy consumption of the system. Finally, the authors in [5] have provided a method for managing the processor's resources with the goal of increasing the system's performance and respect to the power density constraints.

A lot of studies has been done to maximize performance by taking into account various parameters such as process variation, reliability, and temperature considerations [20][21][22][23][39]. Kanduri.et. al. in [25] have presented a temperature management scheme to meet the power constraints on the chip in a way that the performance of the processor is not degraded. This paper presents a sparse mapping algorithm that is superior to dense mapping algorithms in terms of cumulative heat on the chip. Also, several works have been done to study both power and reliability in the homogeneous multi-core systems [6][7][15]. Salehi et al. in [15] have developed a reliability management system for dark silicon processors by taking into account TDP, soft errors, and process variations. Ansari et al. [6] have proposed a peak-power-aware reliability management method that manages peak power overlays among tasks running concurrently such that the system reliability is preserved at an acceptable level. Also, in the work [7], the authors have tried to meet TDP in the standby-sparing systems by proposing two distinct scheduling policies for primary and backup tasks.

However, unlike the presented READY approach, the two concepts of reliability and power-budgeting are not simultaneously considered for graph-based applications in heterogeneous multi-core systems, which are widely used today. Previous works either managed these two concepts in the homogeneous multi-core systems or only managed power and energy in the heterogeneous multi-core systems for this type of application.

III. Preliminaries on System Model

To introduce and evaluate the READY approach, first, we need to explain our system model. In this section we declare application model, platform model, power model, and fault model.

A. Application Model

READY is considered to utilize soft real-time workloads that benefits from graph-based applications. Each application consists of n tasks $\Phi = \{T_1, T_2, \ldots, T_n\}$ that has data dependency with each other and can be modeled as a graph which nodes and edges respectively represent tasks of application and their data dependency [6]. Fig. 1a depicts an abstract example of this graph. The deadline for all tasks of an application is defined commonly and depicted with *D*. Also, the worst-case execution time of the task T_i at the maximum frequency is shown with $WCET_i$.

B. Platform Model

We considered READY to be implemented on top of an island-based architecture of heterogeneous multi-core processors like [2][3]. In island-based architecture, each island benefits from multiple processing cores. We consider that each processor has two distinct types of islands: (1) Low Power Island (LPI) and (2) High-Performance Island (HPI). Each of the islands has several cores, the cores type and their voltages are the same in each island. However, due to applying the Dynamic Frequency Scaling algorithm (DFS), each of the cores of an island can have different frequencies. We declare the number of cores in islands LPI and HPI, respectively, with N_{LPI} and N_{HPI}, and the total number of processor cores with $N = N_{HPI} + N_{LPI}$.

C. Power Model and Analysis

Power consumption in real-time systems includes dynamic power and static power, which is mostly consumed by leakage currents and system activity, respectively [8][15].

$$P_{total}(V_i, f_i) = P_{static} + P_{dynamic} = I_{sub}V_i + \alpha C_L V_i^2 f_i \tag{1}$$

where I_{sub} is a subthreshold leakage current, α is the system activity factor, V_i and f_i are supply voltage and operational frequency, and C_L is the average switched capacitance.

In order to reduce the power consumption, Dynamic Voltage Scaling (DVS) and Dynamic Frequency Scaling (DFS) methods are used such that tasks' timing constraints are not violated. However, exploiting any of these methods in the systems has its own advantages and disadvantages. DVS enables the systems to reduce the power consumption as much as possible, but this method is hardware-dependent, and for its implementation, each core requires additional circuits (i.e. additional hardware overhead) [40][41]. Due to the increasing number of cores on the processors and their heterogeneity, implementation of these circuits imposes unacceptable overhead [6][40][41]. On the other hand, DFS is the software-dependent method and, by decreasing the frequency, is able to reduce the power consumption to meet the power constraint of the system. Also, it should be noted that exploiting the DVFS technique has a very negative impact on the system reliability in two aspects [30]. First, by scaling the supply voltage, the fault rate of the system increases exponentially. Second, reducing the frequency increases the execution time of tasks and has a negative effect on the reliability of the task, as a result degrading the system reliability. However, by exploiting the DFS technique, the fault rate of the system did not change and just the execution time of tasks increases [30]. Hence, in this work, we use the dynamic frequency scaling method which is developed for heterogeneous multi-core architecture.

D. *Fault Model and Reliability Analysis*

In this paper, we consider two common types of faults, (*i*) permanent, and (*ii*) transient [4][6][7]. The incidence of any occurrence that violates the peak power constraints causes system failure and is considered as a permanent fault [6][7]. Also, any perturbed occurrence on the underlying core without permanent damage is considered as a transient fault [4][6][34]. The average fault rate can be represented as follows [6][7]:

$$\lambda(V)=\lambda_0\times 10^{\frac{V_{max}-V}{d}} \tag{2}$$

Where V is the supply voltage, λ_0 is the transient fault rate at the maximum value of V and d determines the system sensitivity to voltage scaling. It should be noted that in this work since we exploit the DFS method in order to manage power consumption, the value of supply voltage is constant. Hence, the value of d doesn't have any effect on the fault rate and the fault rate is equal to λ. Meanwhile, the reliability function of a task can be expressed as follows [6][7]:

$$R(T_i)=e^{-\lambda(V)\times FVI\times wc_i} \tag{3}$$

Where λ(V)×FVI is task failure rate and Function Vulnerability Index (FVI) is the susceptibility of software to failure owing to transient fault occurrence in hardware-level and $\boldsymbol{wc_i}$ is worst-case execution time of $\boldsymbol{T_i}$ [6][7]. Given that several copies may be considered for each task and these copies may have different reliability due to the heterogeneity of the cores; the reliability of each task can be written as [6]:

$$R_{task_i}(R_{replica_1},R_{replica_2},\dots,R_{replica_n})=1-\prod_{k=1}^{n}(1-R_{replica_k}) \tag{4}$$

Since each application consists of several tasks then it's reliability can be calculated as [6]:

$$R_{app_i}(R_{task_1},R_{task_2},\dots,R_{task_n})=\prod_{k=1}^{n}R_{task_k} \tag{5}$$

Finally, the system workload consists of several applications, then the reliability of the system can be written as [6]:

$$R_{system}(R_{app_1},R_{app_2},\dots,R_{app_n})=\prod_{k=1}^{n}R_{app_k} \tag{6}$$

IV. How Does READY Work?

In this section, we will introduce the READY approach in details and explain how READY meets timing, power constraints and reliability target and simultaneously improve schedulability of the heterogeneous multi-core processors for graph-based applications.

A. *READY in Details*

In this paper, we seek to provide a timing constraint- and reliability-aware power-budgeting method for applications mapping and scheduling in heterogeneous multi-core systems. Previous studies have shown that existing power budgeting on the chip surface are not appropriate for heterogeneous multi-core processors. These power budgets for preventing temperature violations are often considered very pessimistic which can violate the applications deadline. The goal of our proposed method is to meet the core-level

Algorithm 1. READY mapping and scheduling algorithm

```
INPUT: readyApps: graph-based applications (each of them consist of n
tasks with the global deadline) with worst-case execution time,
resources: Two set of available cores with different types: $\Phi_{HPI}=\{C_1, \dots, C_{NHPI}\}$, $\Phi_{LPI}=\{C_1, \dots, C_{NLPI}\}$, power-budget: initial core power constraint
$P_{initialTSP,core}$.
OUTPUT: applications Mapping to cores of islands and applications
scheduling.
Body:
 1:  $Slots_{size}$= max (the deadlines of all readyApps);   #Size of time slots in
                                                              the frame
-----------------------------------Offline Part-----------------------------------
.2:  while ($R_{system} < R_{system\_target}$) do    #Meeting the system reliability target
 3:      MinR_App= index.min (the reliability of all readyApps);
 4:      MinR_task= index.min (the reliability of $App_{MinR\_App}$ 's all tasks);
 5:      Insert.replica (MinR_task);          #Insert a replica of $task_{MinR\_task}$
                                                 to $App_{MinR\_App}$
 6:      $R_{system}$ =Update_reliability ();                      #Based on Eq. 6
 7:  end
 8:  while (readyApps is not empty) do          #Mapping ready applications
 9:      MinD_App= Index.min (the deadlines of all readyApps);
10:      Parallelization_factor= max_size(levels of $App_{MinD\_App}$);
11:      $Cores\_Number_{MinD\_App}$=Cores.number(Parallelization_factor);
12:      Critical_Path_Tasks= find_Critical_Path($App_{MinD-App}$);
13:      $C_{MinTSP\_HPI}$ =find_Core(TSP of cores - Critical_Path_Tasks
                        estimated p-power, $Cores\_Number_{MinD\_App}$);
14:      $C_{MinTSP\_HPI}$.add (Critical_Path_Tasks);
                       #Finding min value TSP of HPI cores according to
                  $Cores\_Number_{MinD\_App}$ and mapping critical path tasks
15:      $App_{MinD\_App}$.remove(Critical_Path_Tasks);
16:      Non_Critical_Path_Tasks=$App_{MinD-App}$ - Critical_Path_Tasks;
17:      while (Non_Critical_Path_Tasks have a task) do
18:          Longest_Task_First = find_task(Non_Critical_Path_Tasks);
                              #Finding longest task in terms of WCET
19:          $C_{MinTSP}$ =find_Core (TSP of cores - Longest_Task_First
                        estimated p-power, $Cores\_Number_{MinD-App}$);
       #Finding nearest larger TSP among cores that able to meet the deadline
20:          $C_{MinTSP}$.add (Longest_Task_First);
21:          $App_{MinD\_App}$.remove (Longest_Task_First);
22:      end
23:      readyApps.remove (MinD_App);
24:  end
-----------------------------------Online Part-----------------------------------
25:  for (all cores)                                          #Scheduling part
26:      while (tasks are able to scheduling) do   #Scheduling tasks in each
                                                     core
27:          first_task = find_first_task(each core tasks set);
28:          if all cores TSP constraints will be satisfied
29:              schedule (MinD_task);
30:              update_TSP (all cores);
31:          else
32:              Shift_forward (remained partitions of first_task);
33:      end
34:  end
35:  if all the tasks are not scheduled then
36:      return infeasible;
```

power constraints, preserving the system reliability at an acceptable level and meeting the deadlines of applications. Hence, two algorithms are presented for the applications mapping and scheduling with the desired characteristics and the frequency scaling to reduce the power consumption. The first step of READY is determining the required number of replicas for each task according to the system reliability target. It should be noted that READY never eliminates any tasks such as low-reliability tasks, however, it improves the reliability of them. Indeed, a task with the minimum reliability gets more attention from READY. Because the reliability improvement algorithm in each iteration finds the task with the minimum reliability and introduces the selected task as the system bottleneck in the application, and then inserts a replica for the selected task. This process continues until the reliability of the application satisfies the reliability target. Moreover, it should be mention that READY in any situation is able to meet the reliability target because we

Algorithm 2. READY Sub-Frame-Based DFS algorithm

```
Body:
----------------------------------Online Part--------------------------------------
 1:  (Subframes , NumberofSets) = Find_Subframes(Time-Frame);
 2:  for i=1: NumberofSets
 3:      DFS(Subframes_i);
 4:  end
Function DFS(Sf)
 5:  Slack_Time ← Extract_Slack(Sf);
 6:  Task_ij = Find_Scalable(tasks in Sf);
 7:  $freq_{ij} = max\ (f_{ee}, \frac{wc_{ij}}{wc_{ij} + Slack\_Time})$;
 8:  Perform Task_ij at freq_ij;
```

consider the replica insertion in the offline phase (design time) but we don't claim that READY is able to meet a tight deadline in any condition. If there is an application with high utilization and the deadline of the application be very tight, READY may miss the deadline of the application. In this paper, we will illustrate that in the comparison with the state-of-the-art methods, READY is able to improve the PPA_Sceduablity (i.e. READY executes more tasks by considering deadlines, peak power constraint, and reliability target). Considering a feasible solution, it should be mentioned that the proposed problem is an NP-hard problem [15][32]. Therefore, we cannot find an optimal solution for tasks mapping and scheduling (by considering several constraints) in the polynomial-time in runtime. As a result, we have proposed READY as a heuristic method that meets the power constraint, the reliability target, and the deadlines.

Here, we focus on applications mapping to the cores of the islands and tasks scheduling on the cores. In order to solve the mentioned problem, we have presented a heuristic algorithm in Algorithm 1. First, it should be noted that the value of a time slot depends on the simulation tool and the accuracy of it[1]. Algorithm 1 has two parts: (*i*) the offline part, and (*ii*) the online part. In the offline part, at first, in order to reach the system's reliability target, we find the application with the lowest reliability which is the bottleneck of the system. Then, we find the task of this application with the minimum reliability, and then, a replication of the selected task is inserted into the application to improve the system's reliability. This process is repeated until the system's reliability target is met (lines 2-7). In order to map the applications, the mapping operations start with an application that has the closest deadline to meet the timing constraints (line 9). For mapping the selected application, the algorithm considers a parallelization factor (line 10). In order to allocate appropriate numbers of cores to each application, we have determined this factor for each application. The way to determine this factor is that the algorithm for each application finds the maximum number of tasks that can be executed concurrently (based on the application's task graph). Based on the mentioned factor, the number of cores considered for one application is calculated in line 11. The application mapping operation is followed in two steps. In the first step, tasks that are in the critical path of the application are identified and then mapped to the cores on the high-performance island to meet the application deadlines (line 12). In order to achieve system power efficiency, the cores on the high-performance island are selected that have the nearest larger TSP than the estimated peak power consumption of the application (lines 13-14). It should be noted that in the offline phase there are the peak power consumption estimations table of tasks when running on the available processors. After mapping the tasks on the critical path, the second step of the mapping operation continues with the remaining tasks of the application, where the priority of mapping is with the longest tasks in terms of WCET (lines 16-22). The philosophy of using this policy is that, when we select the priority of the tasks with the longest worst-case execution time, the probability of executing the tasks in the critical path increase. Indeed, by applying this policy, we increase the probability of meeting the application deadline. After completing the mapping of an application, we are looking for the next application that has the closest deadline. The mapping section of Algorithm 1 (lines 8-24) ends with the mapping of all applications. In the online part (lines 25-36), the priority of the tasks scheduling on each core is with the longest unscheduled task whose predecessors have all been scheduled. If the cores do not violate their TSP, we schedule the tasks and then update all TSP values. Otherwise, Algorithm 1 calls *Shift_forward* function and the execution of task shifts to the next time slots (line 32). Indeed, for meeting the TSP constraint at each time slot, we check the power consumption of the mentioned task on the designated core at each time slot. If shifting is required, the current time slot of the selected task is moved to the next time slot that TSP is met. These shifting increases meeting the deadline because if we shift the whole of the mentioned task to the next time slots, we should re-schedule its other dependent tasks. Therefore, in order to increase the deadline meeting and utilize the core efficiently, we use this shift. Finally, after the end of the online part, if not all the tasks are scheduled, the algorithm returns infeasible (lines 35-36).

In order to further reduce power consumption, READY utilizes the sub-frame based DFS algorithm which is illustrated in Algorithm 2. If there are time slots on a core that are not used for any tasks, we consider these time slots as slack time. The algorithm can use these slack times to further reduce power consumption (scaling the frequency). In the graph-based applications, the tasks of an application have data dependency with each other. It should be noted that according to this data dependency the finish time of a task may be the start time of another task. Hence, we cannot apply EVEN DFS algorithm in all slack times. As we mentioned in the description of Algorithm 1, we consider the numbers of cores for each application. In the assigned cores to one application, we find sub-frames that tasks of the application do not have data dependency with each other. Algorithm 2 divides the whole of the system's frame (execution time-bar) into a number of sub-frames (line 1). In each sub-frame, the execution of all the tasks is independent of each other. Then, Algorithm 2 executes the EVEN DFS function for all sub-frames (lines 2-4). The EVEN-DFS technique [42] distributes slacks evenly among all tasks that are able to exploit these slack time (according to the application deadline and other constraints). In this function (lines 5-8), Algorithm 2 finds the slack time of each sub-frame and then assigns them to the tasks that their deadlines and TSP constraints are not violated by allocating these slack times. Also, it should be noted that the algorithm after scaling the frequency checks the reliability of the system with the reliability target. If system reliability

[1] In this work, we have exploited the gem5 architecture simulator and in our simulation, we have captured the system profile every 1ms.

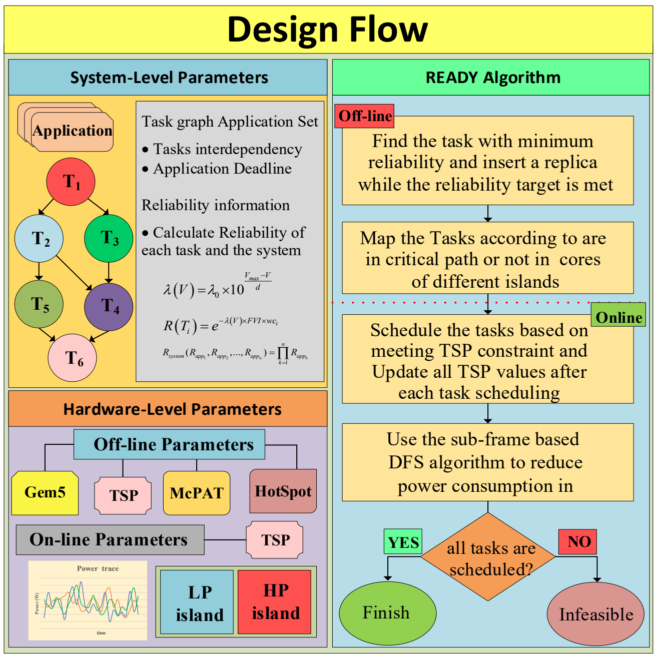

Design Flow
System-Level Parameters
READY Algorithm
Application
Task graph Application Set
• Tasks interdependency
• Application Deadline
Reliability information
• Calculate Reliability of each task and the system
T_1
T_2
T_3
T_4
T_5
T_6
Off-line
Find the task with minimum reliability and insert a replica while the reliability target is met
Map the Tasks according to are in critical path or not in cores of different islands
Online
Schedule the tasks based on meeting TSP constraint and Update all TSP values after each task scheduling
Use the sub-frame based DFS algorithm to reduce power consumption in
all tasks are scheduled?
YES
NO
Finish
Infeasible
Hardware-Level Parameters
Off-line Parameters
Gem5
TSP
McPAT
HotSpot
On-line Parameters
TSP
LP island
HP island

Fig. 2. The design flow of READY

does not meet the target our algorithm increases the frequency in order to meet the target. Finally, we repeat the mentioned steps for the cores of another application until all cores are covered. By assigning these slack times, Algorithm 2 can decrease the execution frequency of the tasks, and hence, the power consumption of the system is further reduced.

B. *Complexity Analysis*

To explore the complexity of the READY's Algorithm, first we define a set of notations. We showed the number of applications with A, the tasks of each application with Ti (i denotes the identification number of the task), the most time consuming task with Tmax which is defined as max(T1, T2, ..., TA), the number of core in HP island with CHPI, the number of core in LP island with CLPI, the total number of cores with C and the slot size with T. In order to find slot size, the maximum values of application deadlines can be found in O(A). Also, in the reliability improvement while loop, the minimum value of applications' reliability founded in O(A) and also the tasks' reliability minimum value of selected application can be found in O(Ti). It's should be noted that in the worst case our algorithm considers 3 replicas for each task of each application (according to reliability target and initial reliability of tasks). Therefore, in the worst case the reliability *while* loop executed in O(A*Tmax) * max(O(A),O(Tmax)). In the next step (offline part) in Algorithm 1, in order to map the tasks of applications to the cores of islands, the minimum values of applications deadlines found in O(A), also the worst case the maximum values of parallelization factor can be found in O(Tmax), in the worst case the critical path of one application can be found in O(Tmax), finding core for tasks in the critical path can be done in O(CHPI), similarly the algorithm can find non-critical tasks and core respectively in O(Tmax) and O(C). Mapping the tasks of one application can be done in O (Tmax * C) and also the mapping of all tasks can be done in O(A*Tmax*C). In the next step (online part) in Algorithm 1, in order to schedule each task, the scheduling operation in each core can be done in O(T) and the scheduling of all tasks can be done in O(T*C). Finally, in order to further reduce power consumption, applying the sub-frame-based DFS can be finished for each time in O(C) and overall, in O(T*C). Therefore, the time complexity of READY is the max(O(A*Tmax)* max(O(A), O(Tmax)), O(A*Tmax*C) , O(T*C),O(T*C)) that is equal to the max(O(A*Tmax)* max(O(A),O(Tmax)),O(A*Tmax*C),O(T*C)). In comparison with the two mentioned methods, the time complexity of TMR and TP3M is max(O(A*Tmax),O(A*Tmax*C), O(T*C)).

Considering READY's space overhead during the runtime, it should be noted that in order to manage peak power consumption, we have employed TPS [8] in READY. The TSP computes the number of simultaneously active cores and determines their positions in the chip with each other. We have calculated all the possible combinations that cores may be active (2^CHPI + 2^CLPI) in the design time and then used the results in runtime. As we mentioned, we have 2^CHPI + 2^CLPI possible combinations and each value of the power constraint needs a byte. Therefore, we need 2^CHPI + 2^CLPI Byte in order to work with peak power consumption in runtime. Moreover, we have a power profile for each task. In our simulation we have exploited 7 types of benchmark, with average execution time 4420ms, therefore we need 30KB (7*4420) space to work with Benchmarks. Also, we had CHPI and CLPI cores in our architecture. Hence, based on system slot size, C*T Bytes are needed for having a power trace on all cores.

V. Experimental Setups and Results

In this section, we evaluate the efficiency of READY in comparison with state-of-the-art approaches. Fig. 2 illustrates the overview of READY by exploiting several simulation tools. To this end, we implemented READY in gem5 simulator [26]. As the workloads of our experiments, we utilized the PARSEC benchmark suite [29]. The power, area, and timing information which is used in this study are retrieved from McPAT [27]. To model the temperature in this study we utilized HotSpot [28], and TSP [8] simulators. Other useful tools such as [43] can also be considered for our

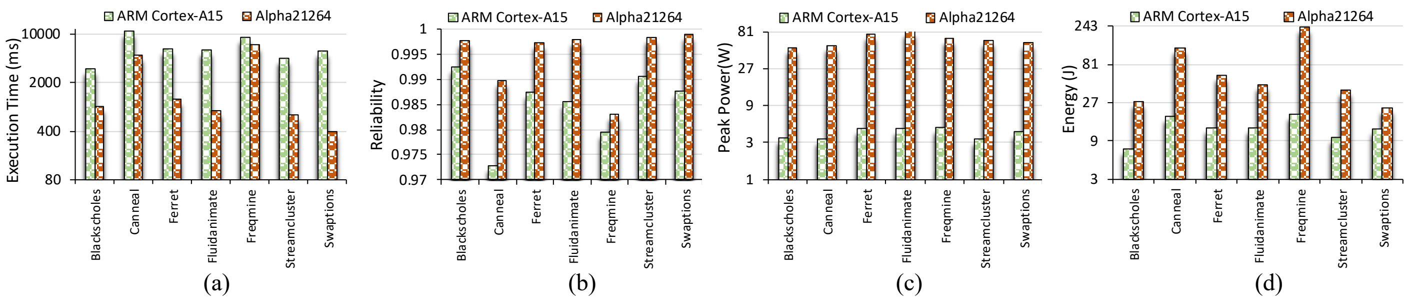
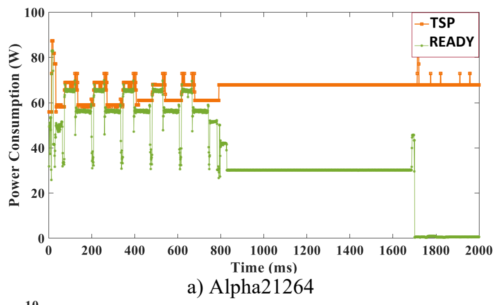
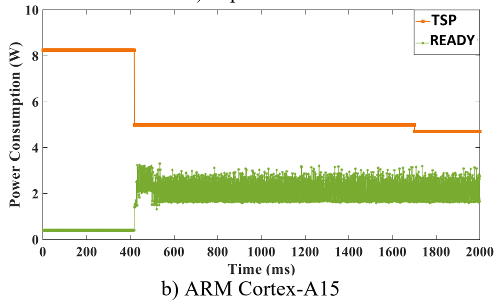


Fig. 3. PARSEC benchmark characterization running on ARM Cortex-A15 and Alpha21264 processors. a) Execution time, b) Reliability, c) Peak power, and d) Energy.



a) Alpha21264



b) ARM Cortex-A15

Fig. 4. Power consumption profile and TSP power constraint adaptation on the cores of Alpha21264 and ARM Cortex-A15 when READY is applied to them.

approach in our future works. The hardware configuration of the simulation is a processor with two heterogeneous islands (each of them including 9 homogeneous cores) which executes different number of tasks (between 36 to 90 tasks). The cores on High-performance Island are considered to be Alpha21264 type, in which execution of tasks have less execution time and more reliability. On the other hand, the cores on Low Power Island, are considered ARM Cortex-A15 which execute tasks with more execution time and less power consumption. In the following, the reliability target and power constraint are introduced, and then the execution scenario is explained to highlight the aspects that are evaluated in the experiments. The results are included in the analysis of the schedulability and feasibility of the system. Finally, we investigate and discuss the insights behind the experimental results.

A. *Reliability Target and Power Constraint*

Regarding the system reliability target, our target is defined according to the avionics DO-178B standard that defines five reliability levels from A with highest to E with lowest reliability levels. Safety requirements of each reliability level are shown in [24] and [45]. In this paper based on the mentioned references, we consider 0.99999 as our reliability target. About considering the deadlines of applications, in the graph-based applications, we consider 20-40 percent more than the execution times of tasks in the critical path as the application deadline [6][31][46]. Finally, about peak power constrains, based on our simulation result from TSP, the peak power constraint in the low power and high-performance island varies between, 4-8.2 watt and 55.9-87.3 watt, respectively.

To clarify our motivation for exploiting heterogeneous platform, we have measured the worst-case execution time, reliability, peak power, and energy consumption values by using the mentioned simulation tools on ARM Cortex-A15 and Alpha21264 processor. The mentioned information is shown in Fig. 3. The measured values represent how heterogeneity can help designers to achieve the desired goals in multi-core real-time systems. Therefore, it is important to provide an efficient power budgeting algorithm in these systems.

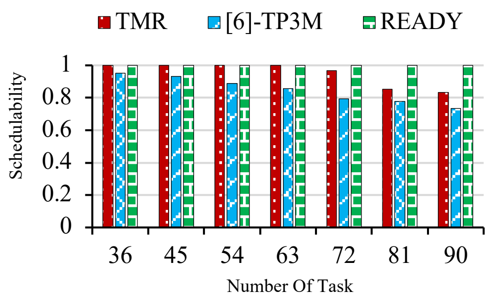


Fig. 5. Schedulability factor analysis.

B. *Execution Scenario*

We evaluated READY in the realistic-case execution scenario on a processor consisting of two islands, low power, and high-performance islands. Each of the islands includes 9 cores. In this scenario, based on the rare nature of the fault occurrence (fault rate $\lambda=10^{-6}$) during a task execution, if each of the tasks' copies is performed correctly, the execution of the remaining copies will be canceled. Fig. 4 shows the power consumption profile and TSP power constraint adaptation on the cores of Alpha21264 and ARM Cortex-A15 when READY is applied to them. The simulation results in Fig. 4 show that our method performs task scheduling by considering a fair peak-power constraint on each core so that neither peak power violation occurs nor reduce system performance loss.

We compared the simulation results of the proposed algorithm in realistic-case execution scenario with the following approaches:

- TP3M: This work presents a peak-power-aware reliability management approach, which removes the overlaps of the peak power of concurrently executing tasks to keep the maximum power consumption below the chip TDP.
- TMR: The conventional Triple Modular Redundancy (TMR) approach attempts to increase the system reliability by considering three replicas for each task executed in parallel.

C. *Analysis of Schedulability*

TP3M approach meets the chip-level constraint but the simulation results in Fig. 5 show that it reduces tasks schedulability by considering a pessimistic power constraint. On the other hand, although the TMR approach has better schedulability relative to TP3M, this approach violates the chip-level and core-level power consumption constraints. READY, by exploiting adaptive power constraint and considering the priority of tasks and system heterogeneity, has been able to have a much better system schedulability. Since READY considers and exploits the features of system heterogeneity, it executes the tasks as soon as possible to meet their deadlines. Hence, the schedulability of READY is higher than other schemes, i.e. TP3M and TMR.

D. *Analysis of Feasibility*

In order to evaluate READY, from meeting power constraints and improving the task schedulability perspectives simultaneously we have defined peak-power-aware schedulability factor. PPA_Schedulability factor is the ratio of the number of time slots, which the peak power constraint is met, to the total execution time slots of the frame

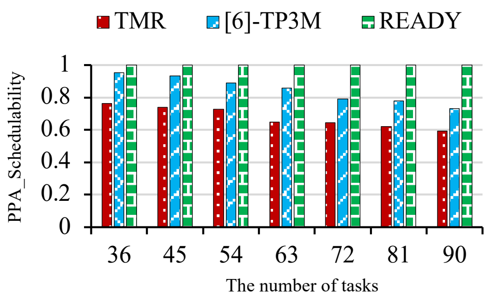


Fig. 6. *PPA_Schedulability* factor analysis.

multiplied by the ratio of the number of tasks which meet their deadline to the total number of tasks.

$$PPA\_Scheduability = \frac{MetPowerConstr\_Times}{Time\_Frame\_Size} \times \frac{MetDeadline\_Tasks}{NumberOfTasks} \quad (7)$$

PPA_Schedulability can decline 0, in the worst-case power-aware schedulability, and can achieve 1 in the best case.

In Fig. 6, a comparison of the simulation results of READY with the two mentioned approaches in PPA_Schedulability factor is shown. Fig. 6 shows that READY performs better in PPA_Schedulability factor. READY has been able to improve PPA_Schedulability by 23.77 percent (up to 40.69 percent). Note that READY and TP3M consider the power constraint for the system, but READY considers the core-level power constraint instead of the chip-level power constraint. Since considering the chip-level power constraint is pessimistic and reduces the performance of the system, READY considers the core-level power constraint, and hence, it performs better in PPA_Schedulability factor. i.e. READY is higher than other schemes in terms of feasibility.

VI. Conclusion

In this paper, we presented READY "reliability- and deadline-aware power budgeting" method. Additionally, we have also proposed a Sub-Frame-Based DFS mechanism to further reduce the power and energy consumption in heterogeneous multi-core systems. READY employs a heuristic algorithm for mapping and scheduling the graph-based applications. Our experimental results show that, in comparison with state-of-the-art approaches, READY has succeeded to improve the peak-power-aware system schedulability by 23.77 percent (up to 40.69 percent) such that the applications' timing constraints, the core-level power constraint, and the system's reliability target are met.

References

[1] T. S. Muthukaruppan, A. Pathania, and T. Mitra, "Price theory-based power management for heterogeneous multi-cores," in *ASPLOS*, ACM, New York, NY, USA, pp. 161-176, 2014.

[2] S. Pagani, et al., "Energy Efficiency for Clustered Heterogeneous Multicores," in *IEEE Trans.on Parallel and Dist. Systems*, vol. 28, no. 5, pp. 1315-1330, 1 May 2017.

[3] H. Khdr et al., "Power Density-Aware Resource Management for Heterogeneous Tiled Multicores," in *IEEE Transactions on Computers*, vol. 66, no. 3, pp. 488-501, 1 March 2017.

[4] F. R. Poursafaei, S. Safari, M. Ansari, M. Salehi and A. Ejlali, "Offline replication and online energy management for hard real-time multicore systems," *2015 CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, pp. 1-7, 2015.

[5] H. Kopetz, "Real-Time Systems: Design Principles for Distributed Embedded Applications," in *Springer US*, pp I-378, 2011.

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

[7] M. Ansari, A. Y. Khaksar, S. Safari and A. Ejlali, "Peak-Power-Aware Energy Management for Periodic Real-Time Applications," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 4, pp. 779-788, April 2020.

[8] S. Pagani, et al., "Thermal Safe Power (TSP): Efficient Power Budgeting for Heterogeneous Manycore Systems in Dark Silicon," in *IEEE Trans. on Computers*, vol. 66, no. 1, pp. 147-162, 1 Jan. 2017.

[9] 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," *CSI J. Comput. Sci. Eng.*, vol. 15, no. 1, pp. 31–38, 2017.

[10] S. Yari-Karin, A. Sahraee, J. Saber-Latibari, M. Ansari, N. Rohbani, and A. Ejlali, "A Comparative Study of Joint Power and Reliability Management Techniques in Multicore Embedded Systems," *2020 CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, pp. 1-7, 2020.

[11] 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,* vol. 12, no. 1, pp. 29-32, 2020.

[12] M. Ansari, J. Saberlatibari, M. Pasandideh, and A. Ejlali, "Simultaneous Management of Peak-Power and Reliability in Heterogeneous Multicore Embedded Systems," in *IEEE Transactions on Parallel and Distributed Systems,* vol. 31, no. 3, pp. 623-633, 1 March 2020.

[13] A. Rahmani, et al., "The Dark Side of Silicon," in *Springer, Cham,* 2017.

[14] H. Esmaeilzadeh, et al., "Dark silicon and the end of multicore scaling," *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pp. 365-376, San Jose, CA, 2011.

[15] M. Salehi et al., "dsReliM: Power-constrained reliability management in Dark-Silicon many-core chips under process variations," *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 75-82, Amsterdam, 2015.

[16] "Intel Xeon Processor - Measuring Processor Power, revision 1.1, "in *White paper, Intel Corporation*, April 2011.

[17] C. Yang, J. Chen, T. Kuo and L. Thiele, "An approximation scheme for energy-efficient scheduling of real-time tasks in heterogeneous multiprocessor systems," *Design, Automation & Test in Europe Conference & Exhibition*, pp. 694-699, Nice, 2009.

[18] A. Elewi, M. Shalan, M. Awadalla, and E. M. Saad, "Energy-efficient task allocation techniques for asymmetric multiprocessor embedded systems," in *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 2s, pp. 71:1–71:27, Jan. 2014.

[19] M. Salehi et al., "DRVS: Power-efficient reliability management through Dynamic Redundancy and Voltage Scaling under variations," *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 225-230, Rome, 2015.

[20] J. Zhan, Y. Xie and G. Sun, "NoC-sprinting: Interconnect for fine-grained sprinting in the dark silicon era," *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1-6, San Francisco, 2014.

[21] B. Raghunathan, Y. Turakhia, S. Garg and D. Marculescu, "Cherry-picking: Exploiting process variations in dark-silicon homogeneous chip multi-processors," *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 39-44, Grenoble, France, 2013.

[22] N. Kapadia and S. Pasricha, "VARSHA: Variation and reliability-aware application scheduling with adaptive parallelism in the dark-silicon era," *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1060-1065, Grenoble, 2015.

[23] Y. Liu, Y. Ruan, Z. Lai and W. Jing, "Energy and thermal aware mapping for mesh-based NoC architectures using multi-objective ant colony algorithm," *3rd International Conference on Computer Research and Development*, pp. 407-411, Shanghai, 2011.

[24] 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 Trans. on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2338-2354, 1 Oct. 2019.

[25] A. kanduri, et al., "adBoost: Thermal Aware Performance Boosting Through Dark Silicon Patterning," in *IEEE Transactions on Computers*, vol. 67, no. 8, pp. 1062-1077, 1 Aug. 2018.

[26] N. Binkert, et al., "The gem5 simulator, " in *2011 ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[27] S. Li, et al., "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 469-480, New York, NY, 2009.

[28] W. Huang, et al., "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," in *IEEE Trans. on Very Large-Scale Integ. (VLSI) Sys.*, vol. 14, no. 5, pp. 501-513, May 2006.

[29] C. Bienia, S. Kumar, J. P. Singh and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," *2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 72-81, Toronto, ON, Canada, 2008.

[30] D. Zhu, R. Melhem, D. Mosse and E. Elnozahy, "Analysis of an energy efficient optimistic TMR scheme," *Proceedings. Tenth International Conference on Parallel and Distributed Systems, ICPADS 2004*, pp. 559-568, Newport Beach, CA, USA, 2004.

[31] M. Salehi, A. Ejlali and B. M. Al-Hashimi, "Two-Phase Low-Energy N-Modular Redundancy for Hard Real-Time Multi-Core Systems," in *IEEE Trans. on Par. and Distr. Sys.*, vol. 27, no. 5, pp. 1497-1510, 1 May 2016.

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

[33] S.-H. Hung, et al., "A real-time, energy-efficient system software suite for heterogeneous multicore platforms," in *Proceedings of the 8th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '12)*, ACM, pp. 23-32, New York, NY, USA 2012.

[34] Z. Shirmohammadi, M. Ansari, S. K. Abharian, S. Safari and S. G. Miremadi, "PAM: A Packet Manipulation Mechanism for Mitigating Crosstalk Faults in NoCs," *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Liverpool, 2015.

[35] S. Safari, M. Ansari, M. Salehi, and A. Ejlali, "Energy- Budget-Aware Reliability Management in Multi-Core Embedded Systems with Hybrid Energy Source," The *CSI Journal on Computer Science and Engineering (JCSE)*, vol. 15, no. 2, pp. 31-43, 2018.

[36] M. Rapp, et al., "Power-and Cache-Aware Task Mapping with Dynamic Power Budgeting for Many-Cores," *in IEEE Trans. on Comp.*, 2020.

[37] P. Mercati, et al., "WARM: Workload-Aware Reliability Management in Linux/Android," *in IEEE Trans. on Computer-Aided Design of Integrated Circuits and Sys.*, vol. 36, no. 9, pp. 1557-1570, Sept. 2017.

[38] J. Zhou, et al., "Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms," *in Journal of Systems and Software*, vol. 133, pp. 1-6, 2017.

[39] J. Zhou, et al., "A Framework to Solve the Energy, Makespan and Lifetime Problems in Reliability-Driven Task Scheduling," *International Conference on iThings and IEEE (GreenCom and IEEE (CPSCom and IEEE Smart Data*, pp. 608-614, Atlanta, GA, USA, 2019.

[40] H. S. Jung, A. J. Gil, and J. T. Kim. "A Case Study of Limited Dynamic Voltage Frequency Scaling in Low-Power Processors," *International Journal of Electrical and Computer Engineering*, no. 12, pp 1523-1526, 2017.

[41] Q. Wang, et al., "Impact of DVFS on n-tier application performance," *Proceedings of the First ACM SIGOPS Conference on Timely Results in Operating Systems* - TRIOS '13, 2013.

[42] 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.

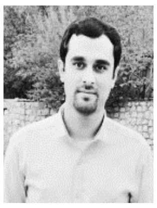
[43] S. Pagani, J. Chen, M. Shafique and J. Henkel, "MatEx: Efficient transient and peak temperature computation for compact thermal models," *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE),* Grenoble, 2015, pp. 1515-1520.

[44] L. Yang, et al., "Dark silicon-aware hardware-software collaborated design for heterogeneous many-core systems," *ASP-DAC, Chiba,* 2017, pp. 494-499.

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

[46] S. Safari, S. Hessabi, and G. Ershadi, "LESS-MICS: A Low Energy Standby-Sparing Scheme for Mixed-Criticality Systems," in *IEEE Trans. on Comp.-Aided Design of Integrated Circuits and Sys.*, 2020.

Javad Saber-Latibari received his M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran in 2020. Also, he is a member of Embedded Systems Research Laboratory (ESR-LAB) at the department of computer engineering, Sharif University of Technology. He is currently planning to start his Ph.D. degree at the University of California, Riverside. His research interest lies in computer architecture, especially in Embedded Systems and reconfigurable systems.

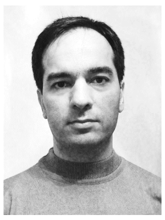


Mohsen Ansari received the M.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2016. He is currently working toward the PhD degree in computer engineering at Sharif University, Tehran, Iran, from 2016 until now. He is now a visiting researcher in the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany. Also, he is 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 and multi-/many-core systems with a focus on dependability/reliability.

Pourya Gohari-Nazari received the B.Sc. degree in computer engineering from the University of Isfahan. He is currently working toward the M.Sc. degree in the Department of Computer Engineering at the Sharif University of Technology, Tehran, Iran. His research interests are thermal management in many-core systems and design embedded systems with a focus on low-power and reliability.

Sina Yari-Karin received his B.Sc. degree in computer engineering from the Ferdowsi University of Mashhad in 2017. He is currently an M.Sc. student in computer engineering at the Sharif University of Technology, Tehran, Iran. Also, he is a member of Embedded Systems Research Laboratory (ESR-LAB) at the department of computer engineering at Sharif University of Technology. His research interests are embedded system design, low power system design, fault-tolerant system design, and computer architecture.

Amir Mahdi Hosseini Monazzah received his Ph.D degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2017. He was a member of the Dependable Systems Laboratory from 2010 to 2017. As a Visiting Researcher, he was with the Embedded Systems Laboratory, University of California, Irvine, CA, USA from 2016 to 2017. As a postdoc fellow he was with the school of computer science, institute for research in fundamental sciences (IPM), Tehran, Iran from 2017 to 2019. He is currently a faculty member of the School of Computer Engineering, Iran University of Science and Technology (IUST), Tehran, Iran. His research interests include investigating the challenges of emerging nonvolatile memories, hybrid memory hierarchy design, and IoT applications.



Alireza Ejlali received the PhD degree in computer engineering from Sharif University of Technology in, Tehran, Iran, in 2006. He is cur-rently an associate professor of computer engineer-ing at Sharif University of Technology. From 2005 to 2006, he was a visiting re-searcher 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 depart-ment of computer engineering and from 2011 to 2015 he was the director of Computer Archi-tecture Group in this department. His research interests in-clude low power design, real-time embedded systems, and fault-tolerant embedded systems.