International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I4P217

Volume 70 Issue 4, 203-214, April 2022

Original Article

Criticality-cognizant Energy-efficient Task Scheduling on Heterogeneous Multicore Processor

N. Gomathi[1], K. Nagalakshmi[2]

[1]*Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, Tamilnadu, India*
[2]*Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, Tamilnadu, India*

[1]nagulaxmi@gmail.com

Received: 02 February 2022 Revised: 27 March 2022 Accepted: 28 March 2022 Published: 26 April 2022

Abstract *- Recently, scheduling mixed-criticality tasks on a common computational system has become an imperative study in academia and engineering proposals. Since multicore processors are the main paradigm in mixed-criticality systems (MCS), reliability and energy consumption are vital concerns. In modern MCS, increased peak power dissipation, particularly in critical scenarios, may cause temperature issues, disturbing the system's consistency and timeliness. This work proposes a criticality-cognizant energy-efficient scheduling approach (CESA) that concurrently provides reliability, power management, and failsafe service level in MCS. The proposed approach decreases the system power dissipation as far as achievable at runtime through the dynamic voltage and frequency scaling (DVFS) approach with laxity allocation. CESA simultaneously accepts a number of tasks (i.e., workloads) and creates clusters with one high-criticality workload and a set of low-criticality workloads. It calculates the available laxities effectively and finds the most suitable task cluster to utilize that available laxity by considering its effect on the instantaneous power consumption and thermal issues. At the same time, varying the core speed, assigning an appropriate cluster for remaining laxity, and selecting a suitable core for task migration at runtime are arduous endeavors and lead to deadline defilement which is not acceptable for high-level workloads. Hence, we propose an online scheduling approach with DVFS and task migration during runtime whenever there is laxity. A cost function is defined as finding out the most suitable cluster to allot the laxities to reduce its V/F level or transfer the task to a new processing element. We assess the performance of our approach in an asymmetric multicore platform (i.e., ARM big. LITTLE processor) with several benchmark task sets. Empirical results demonstrate that the proposed algorithm realizes up to a 6.76% drop in maximum power and a 26.17% drop in core temperature related to the state-of-the-art method.*

Keywords *- ARM big.LITTLE, DVFS, Energy efficiency, Task scheduling, Mixed-criticality system, Multicore processors, Laxity utilization.*

1. Introduction

Following the unprecedented trend in the semiconductor industry, the domain of embedded electronics has faced an irreversible shift towards assimilating multiple tasks on a common processing platform [1]. Integrating more than one application on a shared hardware platform brings innumerable reimbursements to the safety-critical domains, allowing us to execute more tasks simultaneously to increase the reliability and resource utilization while minimizing size, weight, and power dissipation [2]. A system is called safety-critical, whose failure might cause a risk to human life or severe environmental harm (e.g., automotive, nuclear reactors, chemical plants, medical, avionics, etc.) [3]. Tasks in such systems are generally pigeonholed by the degree of criticality, called safety integrity levels (SIL) [5] or design assurance levels (DAL) [4], which are defined as the level of assurance needed against malfunctions.

The workload with a higher degree of criticality reflects that a greater level of assurance is mandatory for reliable system functionality. For example, in the control system of a flight executing surveillance operation, it is required to provide more priority for the accuracy of flight-critical tasks (e.g., tasks related to power system control, actuation control, trajectory, computation, and flight control) so that the aircraft does not crash, than for mission-oriented tasks (e.g., tasks related to reconnaissance purposes including navigation services, tracking potential targets, video surveillance, power steering, navigation services, weapons management, night vision, and parking assistance) or non-critical tasks (e.g., tasks related to vehicular entertainment like music streaming). Indeed, the flight-critical workloads must be finished before the deadline, and significant mission-critical workloads could be discarded in overloaded conditions.

In an MCS, the worst-case execution time (WCET) is an important parameter employed to ensure the correctness of all workloads, particularly high-critical tasks. Every

high-criticality task is defined by two or multiple WCETs [6], including a less conservative, low-level WCET (C_i^1) and a more conservative, high-level WCET(C_i^2). The larger C_i^2 is exploited to provide real-time assurance, and the likelihood that the actual completion time of the workload will be as large as C_i^2 is very low. As a result, mostly, the computational capacity of the system is unexploited since the completion time of workloads is less than the C_i^2, which is estimated through various approaches and techniques [7]. In the present work, we consider a dual-criticality system in which Ci1and Ci2 define each high-level workload $C_i^1$and C_i^2. Each low-critical workload is defined by C_i^1 only.

High-level workloads (τ_i^2) must be schedulable in both low-criticality mode (M^1) and high-criticality mode (M^2); however, the schedulability of low-level workloads (τ_i^1) in M^2 mode depends on the selected scheduling algorithm. Some schedulers impose to drop all τ_i^1 workloads, whereas the others provide the least possible service level. Indeed, the system begins its operation with M^1 mode, and it enters into M^2 mode whenever an τ_i^2 overruns its C_i^1. Now, the scheduler assumes C_i^2 for all the residual workloads to ensure the system correctness, and it continues in this mode until all the high-level workloads are completed; now, the operation of all τ_i^1 and τ_i^2 tasks demands heavy computation, which may surpass the system's capacity, and the processor becomes overloaded [8].

In overrun mode, all the cores perform workloads concurrently to ensure the timeliness of τ_i^2 tasks, which raises the processor's power consumption beyond its TDP limit (thermal design power) [9]. TDP is the peak power that a device can draw without any harm. The overall power dissipation of the processor is the summation of power dissipation of all processing elements irrespective of the task criticality level. Therefore, even undervaluing the power dissipation of a τ_i^1 workload may disrupt the system TDP, which produces a lot of heat beyond the cooling capacity of the core. Hence, we need to restart or halt the system to prevent permanent damage [10]. Consequently, the timeliness constraints of τ_i^2 workloads are violated, which causes jeopardizing effects in safety-critical applications. Thus, it is essential to reduce the power dissipation of workloads at any criticality mode.

Earlier studies have developed algorithms to execute mixed-critical workloads in both M^1 and M^2 modes, but most of them have focused on the average power consumption of the system [11, 12]. These studies exploit the DVFS method and discard low-level workloads in M^2 to tackle average power dissipation, but no one has attempted to control maximum instantaneous power dissipation. In the chorus, some methods cannot be simply implemented in M^2 mode exclusively in the overloaded scenarios since varying the V/F settings of cores enforces increased timing complexity that may lead to timeliness defilement of τ_i^2 and therefore reduce the dependability of the system. However, reducing only the average power is not acceptable. Even though it reduces the system's total power consumption, there is no assurance that the TDP is not disrupted [9]. In this regard, we attempt to minimize system peak power consumption and related thermal issues. One increasing difficulty in performing MCSs is providing fail-safe service levels for low-critical jobs in overloaded scenarios.

This work proposes an energy-aware online scheduling method, CESA, to control the peak power dissipation of an MCS. To reach our goal, we calculate the available laxity (i.e., the deviation between the WCETs and their actual execution time of the workloads) in consort with DVFS. We consider two modes of operation: (i) Offline phase in which the workloads are clustered using the approach given in our previous work [3] and scheduled across cores using the earliest deadline first (EDF) approach for both M^1 and M^2. Then, the resultant scheduling plan is recorded as a static table. The low-level τ_i^1 workloads that have to be discarded in M^2 are reduced; consequently, the system performance is increased; and (ii) online mode in which CESA analyzes the prevailing clusters to find out the best one for execution. For selecting the appropriate cluster, we consider the effect of the workloads in the cluster on the maximum power consumption and thermal profile of the system. Hence, the core speed that executes the workload can be reduced accordingly through scaling.

Moreover, along with the laxity utilization technique, we develop a task migration method to enhance the thermal profile of the system further. This work develops a criticality-aware energy-efficient real-time scheduling approach that reduces peak power dissipation of the mixed-criticality tasks and ensures sufficient response time for low-criticality workloads. The key contributions of our work are five-fold.

- We propose a criticality-cognizant energy-efficient scheduling approach that provides reliability, reduces peak power consumption, and simultaneously enables failsafe service levels for low-critical tasks.
- We generate an offline static scheduling table for both M^1 and M^2 modes.
- We develop a task migration method that utilizes the laxity to migrate the workloads to other cores within a core cluster to reduce the thermal issues in run time.
- We evaluate the performance of our scheduling algorithm with DVFS controlling unit to deliver the timeliness assurance of MCS.
- We also aim to provide an adequate service level for low critical workloads without jeopardizing the timeliness guarantee of the high-level tasks.

The remaining sections are structured as follows: We review the most relevant studies in this domain in Section II. We illustrate the processor, task, and power models employed in Section III. The proposed criticality-cognizant energy-efficient scheduling approach is given in Section IV. We describe the empirical setup and results in Section V. Finally, and Section VI concludes this work.

2. Related works

This section reviews some prior works which are related to our present study. Various earlier studies considered mixed-criticality task scheduling in both design and runtime modes. Since we consider runtime mixed-criticality scheduling to tackle power consumption and related thermal issues, we only focus on the methods developed for MCS with related scope.

References	Techniques /Methods used	Objective(s)	Result	Limitation
Singh et al. [13]	A survey of dynamic energy and thermal management approaches for multicore mobile platforms.	To provide useful insights about dynamic energy and thermal management approaches for multicore mobile platforms.	Upcoming trends and open challenges are identified.	-
Guasque et al. [14]	Energy-efficient partition to CPU allocation	To provide a set of pre-calculated allocations (i.e., profiles) so at run time, the system can switch to various modes based on the existing energy level.	Achieves energy saving of up to a 5%.	It does not provide an optimum solution; instead, it delivers a faster feasible solution.
Salami et al. [15]	A heterogeneous fairness-aware energy-efficient model	To meet fairness constraints and provide energy-efficient scheduling.	Energy consumption is about 33% and 41% less than Linux and Min-Fair, respectively.	Scheduling overhead is high
Li et al. [16]	Thermal and energy-aware mixed-criticality Fluid Scheduling	To minimize the energy consumption and temperature while providing a better schedulability ratio.	Achieve a considerable amount of energy saving	Reduce the energy consumption and temperature with loss of schedulability
Moulik [17]	A three-phase hierarchical resource allocation strategy	To schedule periodic tasks with a bounded number of migrations and context switches.	Achieves 77% success ratio	Resource contention leads to a substantial variation in memory access latencies.
Bao et al. [18]	Online temperature aware DVFS technique to utilize both dynamic and static slack	To reduce energy consumption and temperature-related issues	Achieves s an energy reduction of up to 39%	Not focused on runtime complexities
Kannaian and Palanisamy [19]	Fixed Window dynamic reclamation algorithm	To dynamically adjust the slow-down factor of the processor based on the online task requirements.	The energy consumption of the task set is reduced by 50%	The algorithm is particularly designed for non-preemptive tasks.
Kang et al. [20]	Dynamic scheduling algorithms for reallocating the slack for future tasks	To reduce energy and/or satisfy deadline constraints	Reduces energy up to 14% and reduces deadline miss ratio up to 80%	Not focused on computational complexities

Zhang et al. [21]	Joint optimization schemes of energy efficiency and system reliability for directed acyclic graph (DAG) by adopting the shared recovery technique	To achieve high system reliability and noticeable energy preservation	Reduces energy consumption up to 21.3%	High scheduling overhead
Zhu et al. [22]	Power-aware scheduling algorithms to reclaim the time unused by a task	To reduce the execution speed of future tasks and reduce the total energy consumption of the system	Save up to 44%	overhead on energy-saving ranges from 6% to 12%

Several researchers have assessed their proposed approaches using simulators, and just considered energy or temperature reduction, and are not focused on online performance. They have not considered peak power or related thermal issues, and also, their approach is not appropriate for MCS in which workloads have diverse levels of criticality. In this context, we propose an online scheduling approach for dual-criticality workloads, which are performed on a heterogeneous multicore processor to reduce peak power dissipation and core temperature.

3. Description of System Modeling

The key objective of this work is to reduce peak power dissipation and related thermal problems in an MCS. Although several researchers attempt to tackle or reduce the power dissipation of MCS, they do not focus on peak power in both M^1 and M^2 modes. We developed an online task scheduler to tackle instantaneous power and related thermal issues. We use the DVFS technique with laxity utilization at runtime to reach the goal. In this method, the V/F setting of cores can be varied according to existing laxity time to decrease the peak power dissipation. However, the decisive research questions are (i) how to find out a suitable workload to allocate the available laxity; (ii) if it is feasible to migrate the workloads to other cores for improved temperature control, when and where the workloads are to be migrated; (iii) the computational overhead should be measurable and be as simple as feasible to circumvent intervention with the system workloads; and (iv) the cost function employed to find out suitable workload should not only be easy for estimation but also better to consider other measures such as energy and thermal profiles and calculate the potential effects of the impending workload. This study attempts to explore possible solutions to these issues.

3.1. Processor Model

The multicore system ψ with heterogeneous processing elements λ is represented as $\psi = \{\psi_0, \psi_1, \ldots\ldots \psi_{\lambda-1}\}$. We use the DVFS-enabled ODROID XU3 board, in which the processing elements can operate with various V/F levels. Furthermore, we assume that the resource distribution among the cores does not experience any discrepancy among the workloads performing over the cores. The ODROID XU3 contains two constellations with four A7 (LITTLE) cores and four A15 (big) cores. Each constellation can operate with various V/F levels and process elements within a constellation run at the same V/F level. The allowed voltage and frequency range of cores is given in Table 1.

Table 1. Voltage-frequency range of cores in ODROID XU3

	ARM cortex-A7	ARM cortex-A15
Voltage (V)	[0.9, 1.3]	[0.9, 1.3625]
Frequency (GHz)	[0.2, 1.4]	[0.2, 2.0]

3.2. Task model

Since our ultimate interest is in executing periodic mixed-criticality workloads, we consider a finite set of sporadic workloads scheduling across a heterogeneous multicore system. In this work, we limit our focus to a dual-criticality system where every workload is designated as $\tau_i^x = (\mathcal{P}_i, \mathcal{D}_i, \quad xi, C_i^1, C_i^2)$ with the following semantics: $i \in N^+$ is a task index (i.e., $1 \leq i \leq n$). $\mathcal{P}_i$ denotes the period, and $\mathcal{D}_i$ represents the deadline of the task τ_i^x. Since we focused on implicit-deadline sporadic workloads, every workload τ_i^x has a deadline equivalent to inter-arrival time (i.e., for each $\tau_i^x, \mathcal{P}_i = \mathcal{D}_i$). $x_i = \{1,2\}$ is the criticality level of τ_i^x. Correspondingly, workloads with $x_i \leftarrow 1$ and $x_i \leftarrow 2$ represent low- and high-criticality. $C_i^1 \in N^+$ is the WCET of τ_i^x at a maximum frequency in M^1 mode. $C_i^2 \in N^+$ is the WCET of τ_i^x at a maximum frequency in M^2 mode. Each τ_i^2 task is defined by two WCET values (i.e., C_i^2 and C_i^1). Also, for each $\tau_i^2, C_i^2 \geq C_i^1$. Each τ_i^1 is described by C_i^1. The high-criticality workloads must be completed before their subsequent arrival. Our approach unwinds the service level restraint of τ_i^1 workloads in which some degree of deadline failures are acceptable.

3.3 Power dissipation model

The total power dissipation of the chip can be divided into three measures [2] as given in Equation (1).

$$P_{Total} = P_{Leak} + P_{Dynamic} + P_{innate} \quad (1)$$

where P_{Total} is total power consumption, P_{Leak} is leakage or static power due to bias currents leakage. $P_{Dynamic}$ is dynamic power due to transistor switching. P_{innate} is an innate power of memory and I/O devices. Typically, static power contributes 20–40% of [23]. The key source of power dissipation is the subthreshold leakage current (I_{sub}), which can considerably upsurge with adaptive body biasing.

$$P_{Leak} = V_S \times I_{sub} \tag{2}$$

where V_S is the supply voltage. Dynamic power is a dominant element related to other power constituents in Equation (1). It is a function of core frequency and supply voltage and contributes to the greater part of $the\ total$ in runtime. It can be calculated using Equation (3).

$$P_{Dynamic} = C_{Load} \times V_S^2 \times f \tag{3}$$

where C_{Load} represents the effective load capacitance, and f is the frequency (speed) of the core. As stated earlier, the V/F setting of a whole constellation in ODROID XU3 can be varied. This indicates that the V/F setting of all processing elements in a constellation is identical. This work aims to decrease $P_{Dynamic}$ by utilizing the scaling factor of voltage and frequency as given in Equation (4).

$$\begin{cases} f_{min} \le \alpha_1 \times f_{max} \le f_{max} \\ V_{s\_min} \le \alpha_2 \times V_{s\_max} \le V_{s\_max} \end{cases} \tag{4}$$

Where α_1 and α_2 are the scaling factors of speed and voltage, the total power is calculated using Equation (5).

$$P_{Total} = V_S I_{sub} + C_{Load}(\alpha_2 V_{s\_max})^2 \alpha_1 f_{max} + P_{innate} \tag{5}$$

In ARM big.LITTLE processor, some speed levels set with the same voltage as given in Figure 1. Thus, α_1 and α_2 are not identical. Based on speed levels of A15 and A7 cores, α_1 can be fixed in the range of [0.1, 1] for big cores and [0.143, 1] for the LITTLE cores. Also, α_2 is in [0.6606, 1] and [0.692, 1] for big and LITTLE cores. Even though the board is fabricated with power sensors, they only calculate the power of the whole constellation, not of each processing element. The energy consumption of cores in ODROID XU3 for different frequency levels is given in Figure 2.

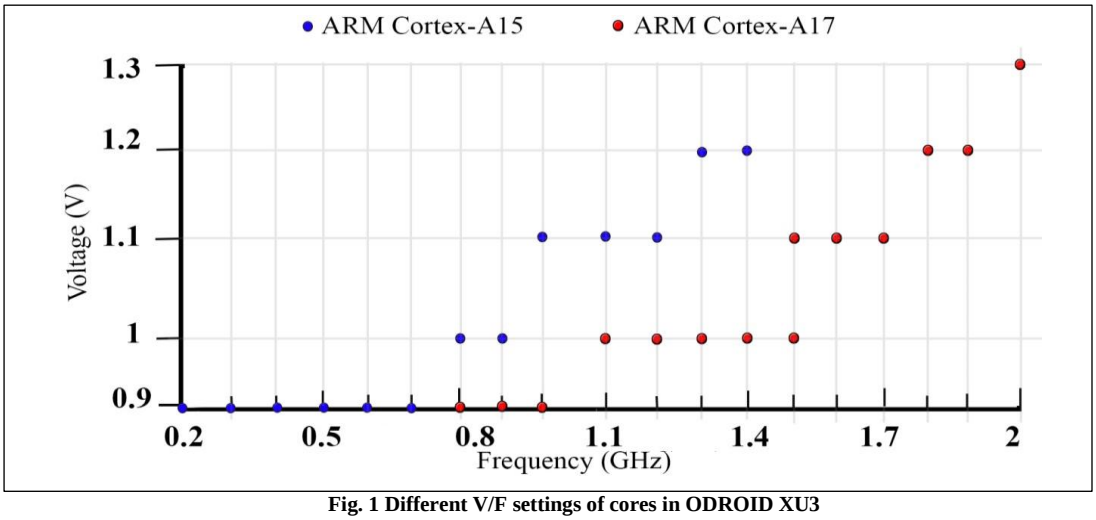

Fig. 1 Different V/F settings of cores in ODROID XU3

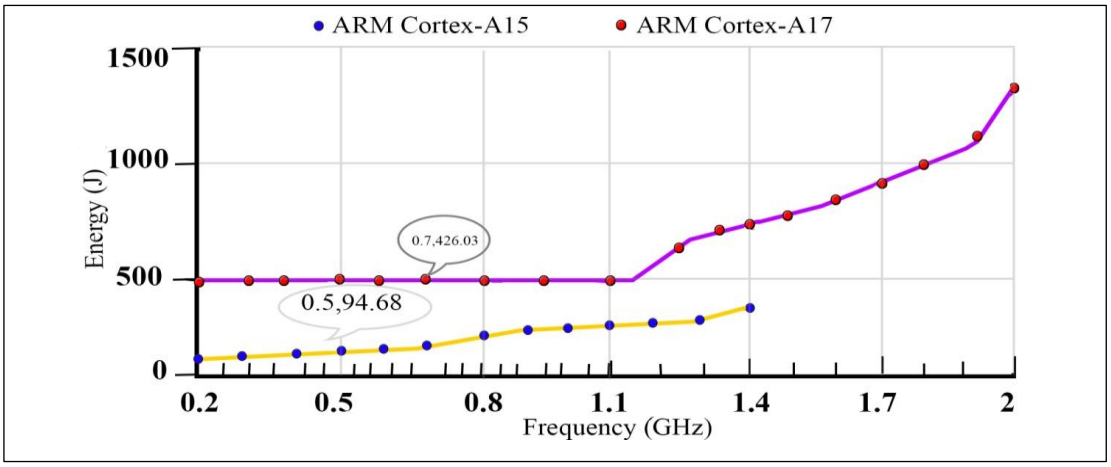

Fig. 2 Energy consumption of cores in ODROID XU3 for different frequency levels

4. Proposed Scheduling Method

The key objective of CESA is to reduce the peak power consumption and the related thermal issues of the processing elements. We exploit the DVFS approach to handle these measures. The objective function of CESA is defined in Equation (6).

$$minimize\ \left(\sum_{j\in cores} P_{\psi_j}, T_{max}\right), \forall\ time\ slot \tag{6}$$

Decreasing the V/F level of a particular core during task execution extends the task completion time and may lead to deadline defilement. Additionally, the overhead of varying V/F levels in the online phase leads to deadline defilement. Equation (7) denotes that the summation of the computation time of τ_i^x at V/F level ℓ on the core ψ_j and timing overhead of scheduling ($\mathcal{O}_s$) and varying V/F level ($\mathcal{O}_v$) should not be surpassed the deadline (d_i) of the task in different criticality modes.

$$\frac{C_i}{f_{\psi_j \ell}} + \mathcal{O}_s + \mathcal{O}_v \leq d_i \begin{cases} C_i = {C_i}^1\ in\ M^1 mode \\ C_i = {C_i}^2\ in\ M^2\ mode \end{cases} \tag{7}$$

Our CESA contains two operating phases such as offline and online. It exploits the online mode for handling instantaneous power and thermal issues; therefore, it is impractical to apply optimization methods like the MINLP (Mixed Integer Non-linear Programming) model due to its increased time complexity. Therefore, we introduce a heuristic-based approach. Figure 3 illustrates the overall architecture of CESA. We employ the ODROID XU3 board for profiling the power consumption of the tasks (in the offline phase) and for performing workload on cores (in the online phase).

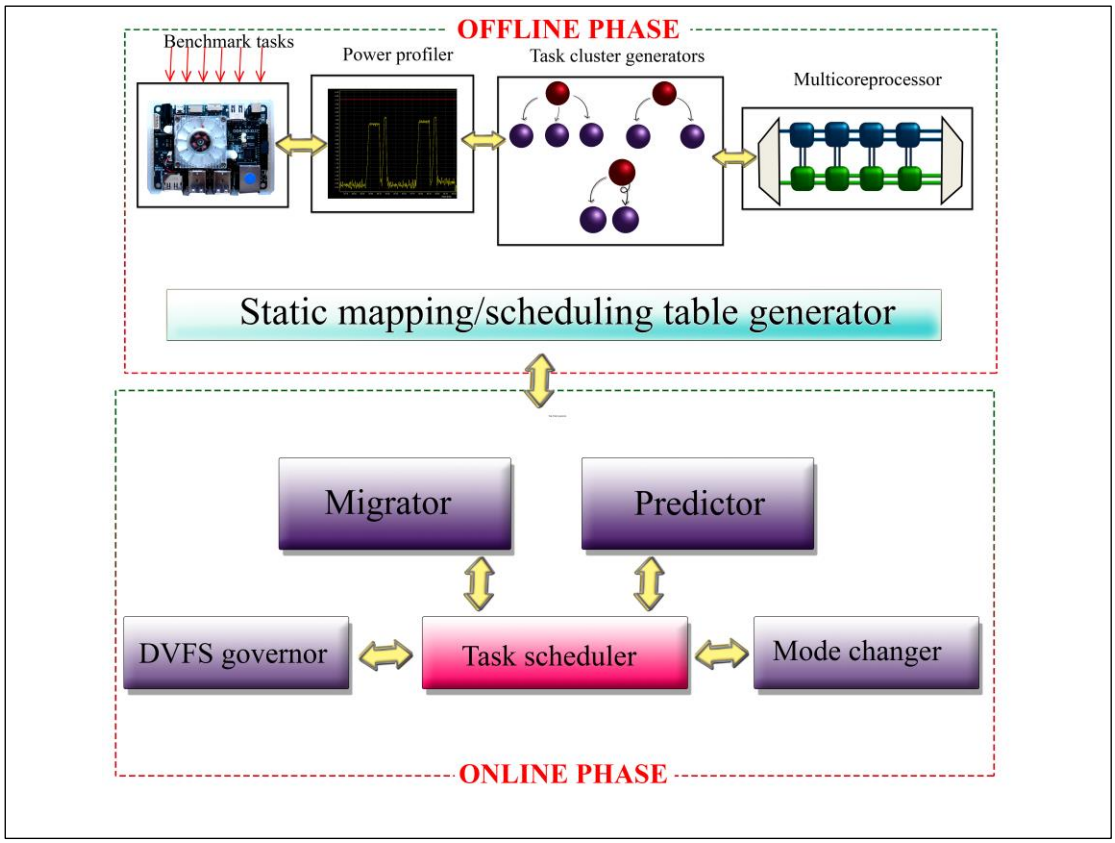

Fig. 3 Overview of the proposed approach

4.1 Offline Phase

In the offline phase, CESA accepts more tasks simultaneously and generates task clusters using the method employed in our previous work [3]. The power consumption of each task can be calculated by executing the benchmark tasks on the ODROID XU3 board. It is noteworthy that managing an unspecified task is out of the scope of our research. As we consider the safety-critical domain, generally, the system architect recognizes the workloads and their features at design time. Hence, two scheduling and mapping tables are generated using the task parameters for M^1 and M^2 modes. The EDF scheduling strategy is applied to find out the schedule in M^1 and M^2 modes statically according to the WCETs of tasks, using the method given in [24]. In mode M^1, all the workloads are performed with equal precedence; in M^2 mode, τ_i^2 workloads are executed with more priority. These predefined tables are then employed to execute workloads during runtime. This imposes a stringent ordering in performing the workloads and assures that all the timeliness constraints are satisfied as per design-time exploration. As ${C_i}^2$ of τ_i^2 is higher, not all τ_i^1 workloads

may be executed in M^2. CESA attempts to discard as few τ_i^1 workloads as possible in creating a table in M^2 mode to increase the service quality. Our approach utilizes these tables and task parameters in the online phase to handle the system.

4.2 Task Clustering

We defined the cluster as a crew of workloads assembled. Each cluster contains a single τ_i^2 and a set of τ_i^1 workloads in our method. The cluster is defined as $S_i = \{\tau_i^2, \tau_1^1, \tau_2^1 \dots \dots \tau_n^1\}$ where $\tau_i^2 (1 \le j \le m)$ is the single high-level workload and the tasks $\tau_i^1 (1 \le i \le n)$ are low-level. The base period (P_{S_i}) of S_i is calculated as the greatest common factor (gcf) of the period of all tasks in a specific cluster, as given in Equation (8).

$$P_{S_i} = gcf\{P(\tau_i^2), P(\tau_1^1), P(\tau_2^1) \dots \dots P(\tau_n^1)\} \tag{8}$$

Where $P(\tau_i^2)$ is the period of τ_i^2 and $P(\tau_i^1)$ is the period of τ_i^1. The number of cluster budget replenishments in $P(\tau_i^1)$ is defined by Equation (9).

$$LO_i^j = \frac{P(\tau_i^1)}{P_{S_i}} \tag{9}$$

Also, the number of budget replenishments in $P(\tau_j^2)$ is defined by Equation (10).

$$HI_j = \frac{P(\tau_j^1)}{P_{S_i}} \tag{10}$$

The cluster S_i needs to accumulate enough execution time budget E_i to ensure all of its tasks meet the schedulability constraint. Our approach executes every cluster S_i as a regular workload with inter-arrival time P_{S_i} and budget E_i. Now, we can calculate the utilization of a cluster by E_i / P_{S_i}.

4.3 Online Phase

The online phase of CESA contains some function controlling modules. The task scheduler is the most important component collaborating with the other modules. It is responsible for scheduling and mapping the workloads. When there is any laxity in a core or a cluster completes its execution earlier, the predictor finds the more suitable task cluster. Suppose a suitable task cluster is assigned for a processing element according to the thermal profile of the current core and the thermal profile of other processing elements. In that case, the task migrator decides whether to transfer the cluster to another core or not to minimize the core temperature. Then, the designated V/F setting for the processing element is tabulated. The DVFS controlling module exploits this designated speed and voltage to execute the workload. The DVFS controller is used to find an ideal V/F level for a particular core constellation. Owing to mixed-criticality performance, the system enters into M^2 if the processing time of anyone τ_i^2 exceeds its ${C_i}^1$. The mode changer should verify it. The processor changes its scheduling strategy based on the scheduling table in this scenario.

4.4 Selecting the Appropriate Task to Assign Laxity

Our predictor module selects a cluster after calculating laxity and maps that cluster on the core where the laxity $(\mathcal{L})$ is observed. A cost function (∂_i) is defined for every cluster, as given in Equation (11).

$$\partial_i = \rho E_i + \sigma P_i \tag{11}$$

where E_i and P_i are the cluster's maximum energy and instantaneous power, correspondingly. The factors ρ and σ are in [0, 1]. Indeed, a decrease in energy consumption causes a drop in core temperature. It is noteworthy that if we assume $\langle \rho, \sigma \rangle = \langle 1, 0 \rangle$, then ∂_i only focus on the power consumption and not its energy. Therefore, the cluster with the higher instantaneous power is selected to be performed at a lower frequency to decrease the system's peak power. If we select $\langle \rho, \sigma \rangle = \langle 0, 1 \rangle$, ∂_i only consider energy. Therefore, the cluster with the highest energy consumption is selected to be performed at minimum frequency, thus decreasing the maximum energy consumption. After choosing the cluster, the maximum power dissipation and its WCET (${C_i}^1$ or ${C_i}^2$) are varied according to the size of observed laxity and the V/F setting. Additionally, Equation (11) is used to a task cluster that can initiate their process early. A workload τ_i^x can execute earlier if it has arrived before $a_i - \mathcal{L}$; here, a_i is the commencement of τ_i^x. A new workload can be issued when all its ancestors complete their process. Hence, we form the following constraint.

$$T_{ri} < a_i - \mathcal{L}_{i-1} \tag{12}$$

In Equation (12), T_{ri} is the time of task release. Assume the designated workload τ_i^x with deadline d_i and the start time a_i that $a_i + C_i \le d_i$. Let us assume that we have the laxity time, $\mathcal{L}_{i-1}$ produced by τ_{i-1}^x during execution. To use this laxity for the suitable workload τ_i^x, generally, the task scheduler calculates the minimum suitable core speed using Equation (13).

$$f_i = max\left(f_{min}, \frac{C_i}{C_i + \mathcal{L}_{i-1}}, f_{max}\right) \tag{13}$$

This guarantees that only the commencement of the workload is earlier by $\mathcal{L}_{i-1}$, and the deadline is kept constant. Therefore,

$$a_i - \mathcal{L}_{i-1} + \frac{C_i}{\left(\frac{f_i}{f_{max}}\right)} \le a_i + C_i \le d_i \tag{14}$$

On the other hand, selecting an appropriate workload and the processing element and varying the V/F setting create overheads. If we neglect to select the optimal speed, it may lead to deadline defilement. Consequently, $\mathcal{L}_{i-1}$ is decreased by $\mathcal{O}_s$ and $\mathcal{O}_v$. After finding the optimal speed,

the start time of the suitable cluster is updated for the static schedule.

4.5 Task Migration Technique

CESA shifts the designated workload to the other cores without altering its deadline for controlling the core temperature. We define a cost function as given in Equation (15) to decide about the migration and identify the suitable processing element to migrate.

$$\partial_c = \gamma \sum_{t=1}^{t_c} E_c\,(t) \tag{15}$$

Here, the core temperature is calculated according to the aggregated energy. We observed that a core is likely to hoard a lesser temperature when its cumulative energy consumption is lower than the others. Conversely, the variation between the cumulative energy consumption of the designated processing element to migrate and the current processing element ought to be large enough. Consequently, we introduce a parameter (γ) (in our experiments, $\gamma = 0.9$). In Equation (15), t_c is the completion time of a specific workload. Also, do not disturb the tasks' deadline scheduled on other processing elements; they are studied for migrations that have laxity to perform the apt workload. Since we employ an asymmetric multicore system for our experimentation, each task cluster's execution time and power dissipation will be different when executing on different constellations. Even though migration from an A7 to A15 core decreases the task's execution time, it causes increased instantaneous power dissipation, which is not suitable for safety-critical applications. Hence, we apply the migration method within the constellation to retain the peak power dissipation. Since the migration method is used to a cluster that is not processed yet, the method is implemented concurrently with different speeds, and the relocation overhead does not disturb the timeliness restraints. The reason is that the overhead of migration is very low compared to the overhead related to varying the core speed.

4.6 Update V/F Levels

After completing a workload, there might be a laxity or a workload in the queue that is ready to start its execution. All processing elements within a constellation run at the identical V/F setting in a heterogeneous multicore system. Since the V/F settings of both constellations are not the same, it is checked on which constellation the recently accomplished workload was executing. Subsequently, we check the allocated V/F level of executing or ready tasks on all cores of the constellation. As cores within a constellation work with the same frequency, we select the optimum speed for the constellation. Choosing the greatest minimum speed is to guarantee that all workloads are finished without violating their deadline. Finally, if the selected speed is diverse from constellation speed, we vary the speed of the constellation by allocating the new frequency. It should be noted that by varying the speed of a constellation, its voltage will inevitably vary.

4.6.1 Evaluation

We carried out experiments on the DVFS-enabled ARM big to assess our proposed approach.LITTLE multicore system (ODROID XU3 board). Since it supports different V/F levels, we analyze the impact of different V/F settings by scaling the core speed within the given range. We generate random tasks using the tool presented in [24] to carry out experimentations. To achieve a feasible range of power consumption, we execute numerous benchmark tasks from the MiBench suite [25] on the ODROID XU3 processor with higher speed and measure the power consumption from sensors fabricated on the panel. As the V/F scaling is used for the entire system, the power dissipation at other lower speeds can be calculated by scaling the system frequency [26].

Additionally, we studied the impact of the number of cores by executing benchmarks on 1 to 8 cores. We execute every task 1000 times with diverse parameters (i.e., WCETs, actual execution times, and timeliness constraints) and report the average results. We observed the maximum power consumption of workloads in [0.492, 0.923] W in LITTLE cores and [2.986, 6.856] W in big cores.

We assess the enactment of our method under three different cases, as illustrated in Figures 4 - 6, wherein the results are normalized to [24]. Generally, as the applications become multifaceted (e.g., having several workloads or maximum system utilization), it is very difficult to realize the significant drop in maximum power, core temperature, and energy. We can achieve reduced maximum power consumption when the number of processing elements increases. The proposed migration method is used to redistribute the workloads more uniformly to the cores at runtime according to their cumulative energy.

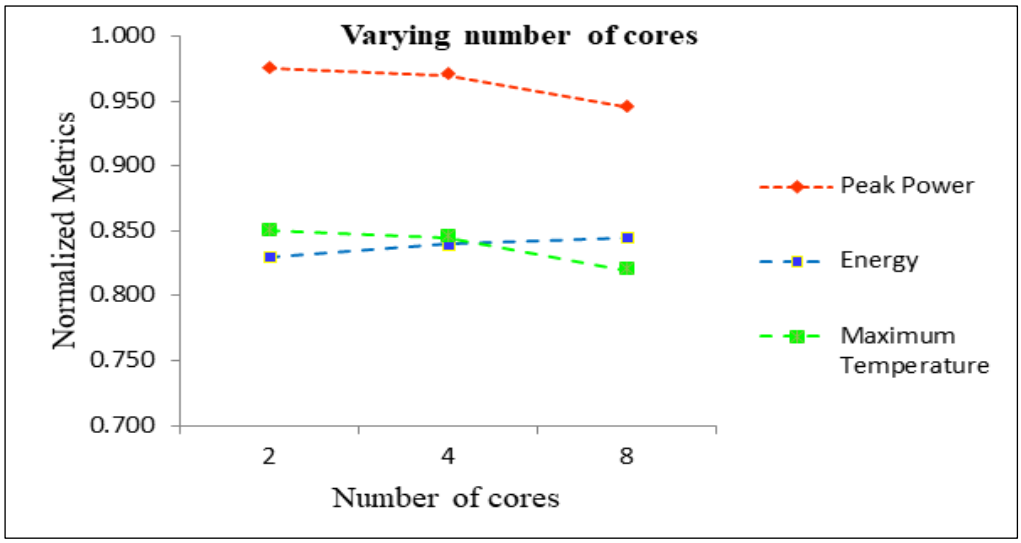

Fig. 4 The varying number of cores

Since CESA only focuses on reducing peak power consumption for each core independently, it is very hard to achieve a similar reduction in power when fewer cores are used. However, as shown in Figure 4, the variation in maximum power is substantial by growing the number of processing elements. Besides, as the adjacent elements exaggerate the temperature of the individual element, the maximum temperature drop is low by adding more processing elements. By utilizing our proposed method, the maximum power, energy, and temperature are decreased by 6.315%, 16.271%, and 21.12%, respectively.

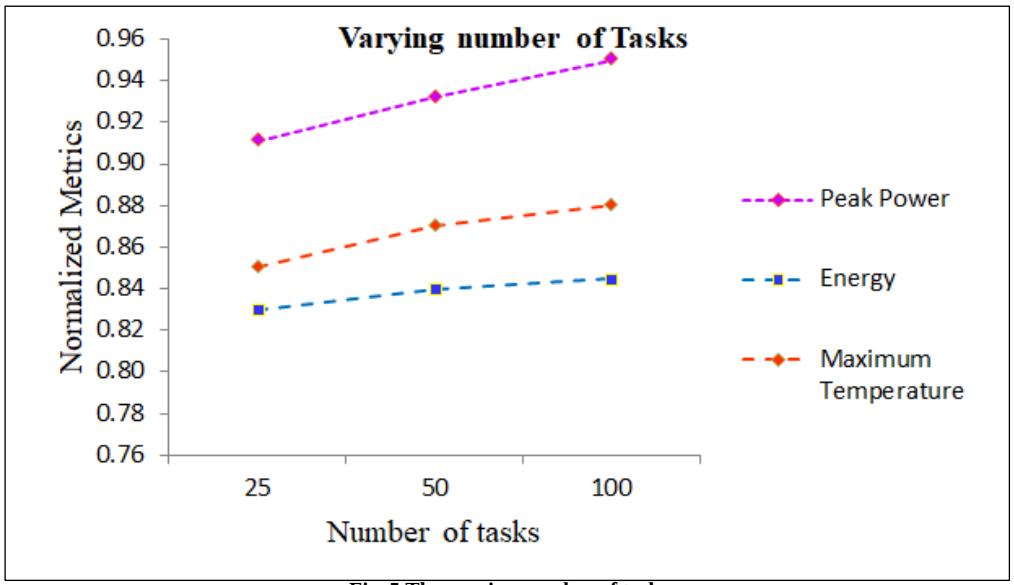

Fig. 5 The varying number of tasks

The efficiency of CESA hinges on the existing laxity during runtime and the opportunity of allocating them to the workloads. Hence, if there is small laxity observed due to the application type concerning the number of workloads and system utilization, maximum power, core temperature, and energy reduction are less. In Figure 5, when the utilization increases, the idle time of the processing element between two successive workloads are reduced. The workloads also tend to perform longer. Consequently, the size of laxities that can be used at runtime is restricted. In this scenario, our method achieves a minimum of 4.215% and a maximum 8.719% drop in maximum power dissipation.

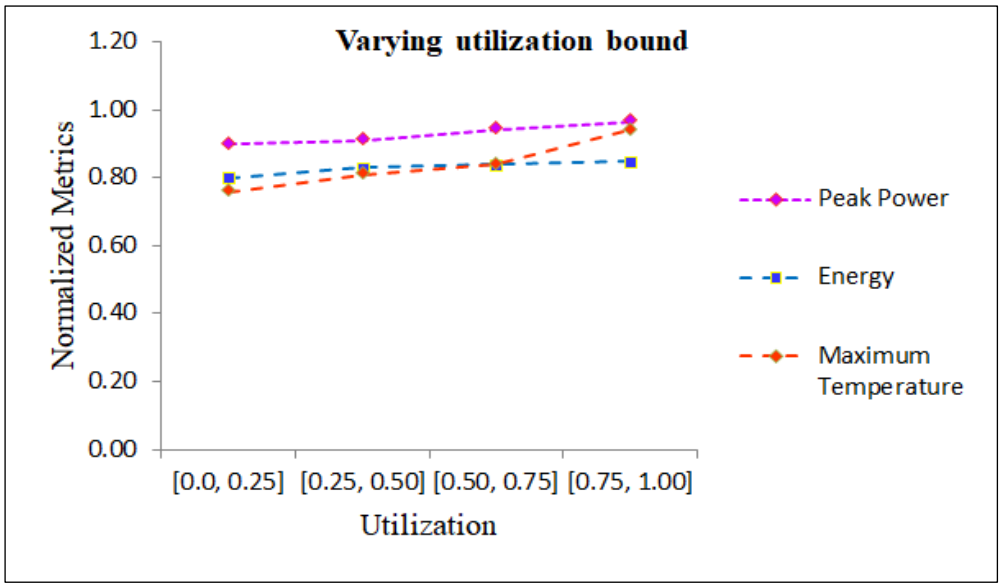

Fig. 6 Varying utilization bound

To study the system's temperature, we execute the workloads on Core 2 and 3 that generally realize maximum temperature owing to their closeness to the memory and other elements. The board consists of sensing devices to measure the temperature of every A15 core and measure the power consumption of each constellation. Hence, the power and temperature values are measured from these sensing devices. Figure 7 illustrates the power trace of the constellation with A15 cores during runtime using CESA and a state-of-the-art method [24].

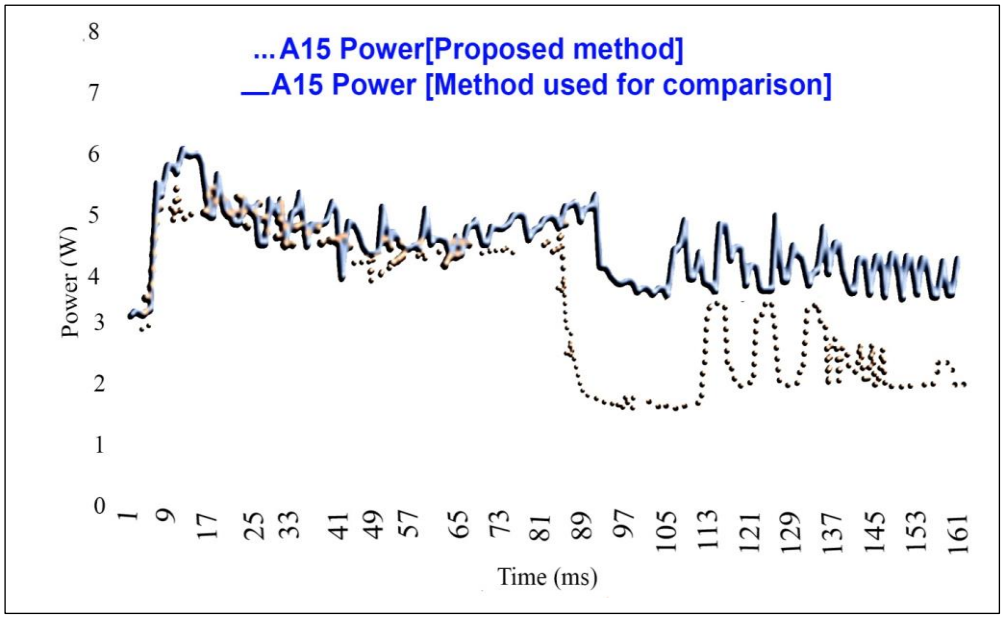

Fig. 7 Power trace of the constellation with A15 cores

The temperature traces of Core 2 and Core 3 are illustrated in Figures 8 and 9, respectively. The core temperature has been reduced by CESA significantly. After implementing our approach and decreasing the V/F values, the temperatures of the cores are decreased. Thus, CESA will be more efficient and provide an important performance enhancement whenever more workloads are executed with more cores.

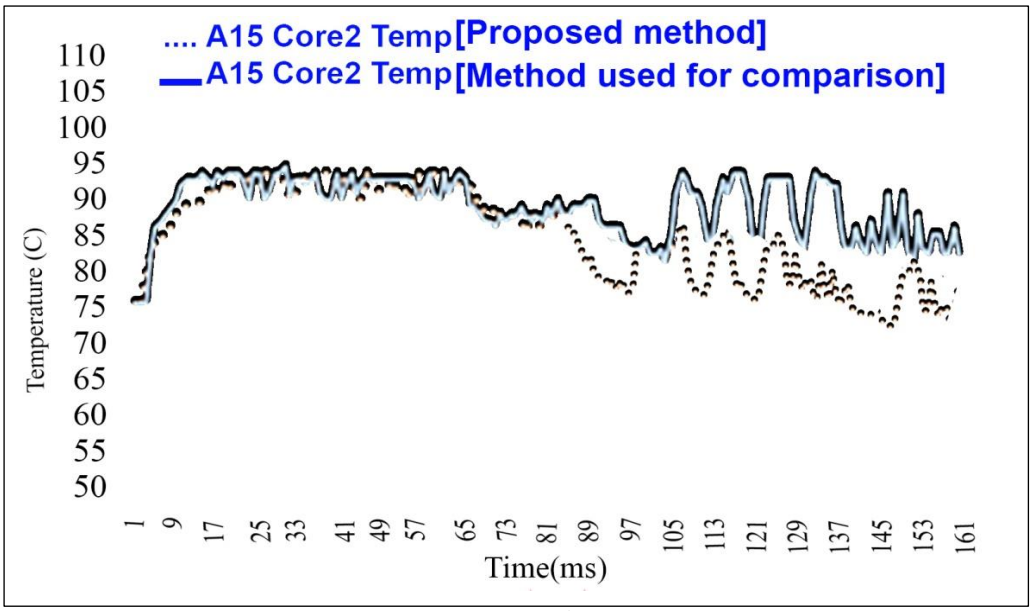

Fig. 8 Temperature trace of A15-core2

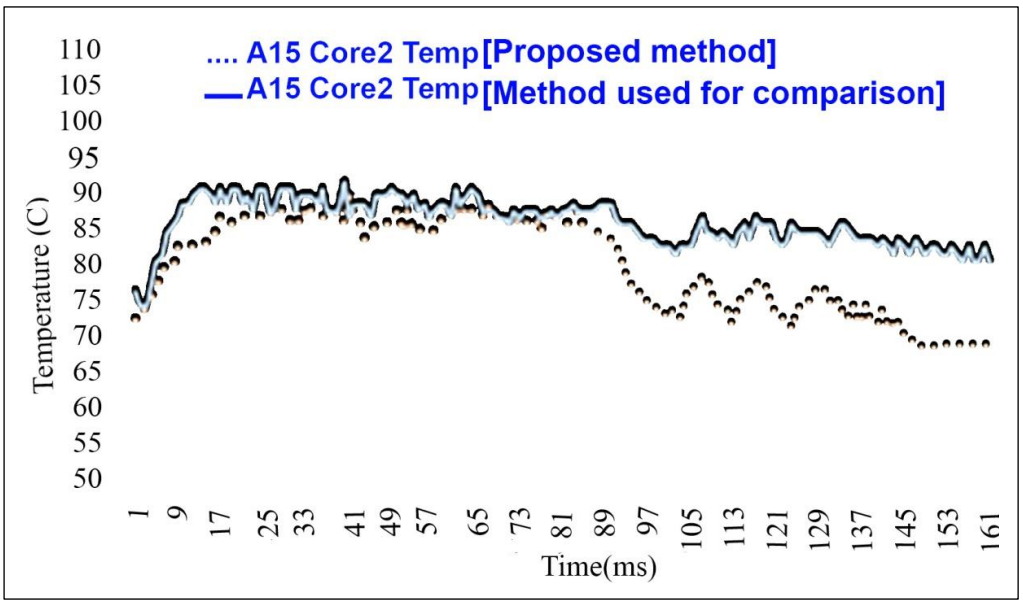

Fig. 9 Temperature trace of A15-core3

5. Conclusion

This paper has developed a criticality-cognizant energy-efficient scheduling approach for a dual-criticality system on a heterogeneous multicore processor. The proposed approach decreases the system power dissipation as far as achievable during runtime using DVFS with laxity allocation technique. In this method, each high-criticality task is combined in a cluster with a set of low-level workloads (to increase schedulability while maintaining the timeliness guarantee). It calculates the available laxity effectively and finds out the most suitable task cluster to utilize the laxity by considering its effect on the maximum power and thermal profile. At the same time, varying the core speed, assigning an appropriate task cluster for remaining laxity, and selecting a suitable core for task migration at runtime are arduous endeavors and lead to deadline defilement which is not acceptable for high-level workloads. CESA exploits task migration and DVFS during online mode whenever there is laxity. To find an apt cluster to allocate the laxities to reduce its V/F level or transfer it to another processing element, we define two cost functions. We assess the performance of our proposed approach in an asymmetric multicore platform with several benchmark task sets. Empirical results demonstrate that the proposed method realizes up to a 6.76% drop in maximum instantaneous power and a 26.17% decrease in core temperature related to the state-of-the-art method.

References

[1] K. Nagalakshmi and N. Gomathi, An Irreversible Transition towards Multicore Platform in Safety-Critical Domain for the Aviation Industries, International Journal of Scientific Research in Science Engineering and Technology. 2(5) (2016) 345-359.

[2] K. Nagalakshmi and N. Gomathi, Analysis of Power Management Techniques in Multicore Processors, In proceeding of International Conference on Artificial Intelligence and Evolutionary Computations in Engineering Systems, Advances in Intelligent Systems and Computing, Springer. 517 (2017) 397-418.

[3] Nagalakshmi K, Gomathi, Criticality-Cognizant Clustering-Based Task Scheduling on Multicore Processors in the Avionics Domain, International Journal of Computational Intelligence Systems. 11 (2018) 219–237

[4] ISO 26262, Road Vehicles - Functional Safety. (2011).

[5] RTCA. DO-178C/ED-12C Software Considerations in Airborne Systems and Equipment Certification. (2012).

[6] N. Srivastava, V. Pandey, R. Pathak, Abhishek Pandey, Multicore: Move to the Future, International Journal of Engineering Trends and Technology. 4(2) (2013) 204-206.

[7] A. Kritikakou and S. Skalistis, Progress-aware Dynamic Slack Exploitation in Mixed-critical Systems: Work-in-Progress, 2020 International Conference on Embedded Software (EMSOFT). (2020) 10-12.

[8] J. Simó, P. Balbastre, JF. Blanes, J-L.Poza-Luján and A. Guasque, The Role of Mixed-Criticality Technology in Industry 4.0 Electronics. 10(3) (2021) 226.

[9] 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, IEEE Transactions on Parallel and Distributed Systems. 30(1) (2019) 161-173.

[10] S. Hosseinimotlagh, A. Ghahremannezhad, and H. Kim, On Dynamic Thermal Conditions in Mixed-Criticality Systems, 2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). (2020) 336-349.

[11] H. Sobhani, S. Safari, J. Saber-Latibari, and S. Hessabi, REALISM: Reliability-Aware Energy Management in Multi-Level Mixed-Criticality Systems with Service Level Degradation, Journal of Systems Architecture. 117(2) (2021) 102090.

[12] I. Ali, Y-I. Jo, S. Lee, and KH. Kim, Reducing Dynamic Power Consumption in Mixed-Critical Real-Time Systems, Applied Sciences. 10(20) (2020) 7256.

[13] AK. Singh, S. Dey, KR. Basireddy, K. McDonald-Maier, GV. Merrett, and BM. Al-Hashimi, Dynamic Energy and Thermal Management of Multi-Core Mobile Platforms: A Survey, IEEE Design & Test. 10(1) (2020) 335-367.

[14] A. Guasque, P. Balbastre, A. Crespo and S. Peiró, Energy-Efficient Partition Allocation in Mixed-Criticality Systems, PLOS ONE. 14(3) (2019) e0213333.

[15] B. Salami, H. Noori and M. Naghibzadeh, Fairness-Aware Energy Efficient Scheduling on Heterogeneous Multi-Core Processors, in IEEE Transactions on Computers. 70(1) (2021) 72-82.

[16] T. Li, T. Zhang, G. Yu, Y. Zhan and J. Song, Ta-mcf: Thermal Aware Fluid Scheduling for the Mixed-Criticality System, Journal of Circuits, Systems and Computers. 28(2) (2019) 1950029.

[17] S. Moulik, RESET A Real-Time Scheduler for Energy and Temperature Aware Heterogeneous Multi-Core Systems, Integration. 77 (2021) 59-69.

[18] M. Bao, A. Andrei, P. Eles, and Z. Peng, On-Line Thermal Aware Dynamic Voltage Scaling for Energy Optimization with Frequency/Temperature Dependency Consideration, In Proc. ACM/IEEE DAC-2009. (2009) 490–495.

[19] V. Kannaian and V. Palanisamy, Energy-Efficient Scheduling for Real-Time Tasks using Dynamic Slack Reclamation, Turkish Journal of Electrical Engineering and Computer Sciences. 27(4) (2019) 2746-2754.

[20] J. Kang and S. Ranka, Dynamic Slack Allocation Algorithms for Energy Minimization on Parallel Machines, Journal of Parallel and Distributed Computing. 70(5) (2010) 417–430.

[21] L. Zhang, K. Li, K. Li, Y. Xu, Joint Optimization of Energy Efficiency and System Reliability for Precedence Constrained Tasks in Heterogeneous Systems, International Journal of Electrical Power & Energy Systems. 78 (2016) 499–512.

[22] D. Zhu, R. Melhem and BR. Childers, Scheduling with Dynamic Voltage/Speed Adjustment using Slack Reclamation in Multiprocessor Real-Time Systems, IEEE Transactions on Parallel and Distributed Systems (TPDS). 14(7) (2003) 686–700.

[23] D. Chinnery and K. Keutzer, Overview of the Factors Affecting the Power Consumption, In Proc. TTLPD-2007, Springer. (2007) 11–53.

[24] R. Medina, E. Borde, and L. Pautet, Availability Enhancement, and Analysis for Mixed-Criticality Systems on Multi-Core, In Proc. DATE-2018. (2018) 1271–1276.

[25] MR. Gasthaus, JS. Ringenberg, D. Ernst, TM. Austin, T. Mudge, and RB. Brown, Mibench: A Free, Commercially Representative Embedded Benchmark Suite, In Proc. WWC-4. (2001) 3–14.

[26] S.P Rahul Santosh, K. Somasekhara Rao, Design of Real-Time Interactive Data Acquisition and Control SystemUsing ARM, International Journal of Engineering Trends and Technology (IJETT). 4(10) (2013) 4418- 4421

[27] M. Salehi and A. Ejlali, A Hardware Platform for Evaluating Low-Energy Multiprocessor Embedded Systems Based on Cots Devices, IEEE Transactions on Industrial Electronics. 62(2) (2015) 1262– 1269.