Journal of Theoretical and Applied Information Technology
31st May 2023. Vol.101. No 10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

LAXITY-AWARE MIXED-CRITICALITYTASK SCHEDULING FOR ENERGY-EFFICIENT HETEROGENEOUS MULTICORE PROCESSORS

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. , gomathin9746@gmail.com

[2]Research scholar, Vel Tech Rangarajan Dr. Sagunthala R & D Institute of Science and Technology, Chennai, Tamilnadu, India.

ABSTRACT

Mixed-criticality systems (MCS) have developed as an efficient solution in several industries, where numerous tasks with different criticality levels (safety requirements) are assimilated onto a shared computational platform. Today, increased energy consumption in MCS, especially in critical situations, leads to temperature hotspots, which may disrupt the reliability and correctness of the system. As processors with multiple processing elements are becoming the vital paradigm in MCS, an integrated timeliness and power management is an important issue. This paper proposes a laxity-aware mixed-critical task scheduling (LMTS) algorithm that provides correctness, timeliness, power management, and guaranteed service level in MCS simutaneously. This method minimizes energy consumption of the system considerably through dynamic voltage and frequency scaling (DVFS) method. It collects several workloads concurrently and form clusters with one high-critical workload and a set of low-critical workloads. It determines the laxities and selects the most suitable cluster to exploit the available laxity based on its impact on the energy consumption and hotspot problems of the system. However, changing the core frequency, allocating more suitable cluster for available laxity, and finding out an appropriate core for mapping at runtime are difficult processes and cause deadline desecration which is not suitable for safety-critical tasks. Therefore, we develop an effective scheduling method using DVFS schemes and task migration techniques in online mode to utilize available laxity. We also defined cost functions to select the most apposite cluster to right core by scaling its voltage/frequency (v/f) value or to migrate it to another processing element. We assess the effectiveness of our scheduling algorithm in a heterogeneous multicore processor with real-time tasks.

Keywords— *Arm Big. LITTLE; DVFS; Energy Efficiency; Task Scheduling; Mixed-Criticality System; Multi-Core Processors; Laxity Utilization*.

1. AIM AND BACKGROUND

The inexorable improvements in embedded technology allow system designer to devise more processing elements (cores) on a single chip to achieve high performance computing with improved reliability at low cost. Therefore, the field of embedded microelectronic has encountered an irretrievable shift towards integrating numerous workloads on a shared hardware platform [1]. Assimilating multiple applications on a common computational hardware fetches numerous benefits to the safety-critical systems such as improved dependability as well as resource utilization while reducing energy consumption, size, and weight [2]. A system is known as safety-critical whose failure might lead to a severe environmental risk to human life [3]. The task with higher criticality level denotes that a maximum guarantee is obligatory for correctness of the system. In a safety-critical system, the worst-case execution time (WCET) is a vital factor employed to provide real-time guarantee for all the tasks, especially high-level tasks. Each high-critical task (τ_i^2) is defined by two or more WCETs [4] with a more conservative, high-critical WCET (C_i^2) and less conservative, low-critical WCET (C_i^1). Here, i represents task index. The greater C_i^2 is used to provide maximum timeliness guarantee. However, the probability that the real execution time of the task will be equal to C_i^2 is very less. Consequently, in most of the cases, the processor is underutilized as the actual

execution time of tasks is less than C_i^2 [5]. In this work, we focus on a dual-criticality system where each τ_i^2 is defined by C_i^1 and C_i^2 and a low-level task (τ_i^1) is defined by C_i^1 only.

Several studies have proposed methods to implement mixed-critical tasks in both M^1 and M^2 modes; however, most of them have only considered average energy dissipation of the system [6]. These works use the DVFS scheme and reject τ_i^1 in M^2 mode to manage average energy consumption; but, none of them has endeavored to manage instantaneous energy consumption. Moreover, few algorithms cannot be just employed in M^2 especially in the critical conditions; since changing the v/f levels of processing elements imposes amplified timing overhead that leads to deadline defilement of τ_i^2 and also decreases the system dependability. It is worth mentioning that decreasing only the average power consumption is not adequate. Although it may decrease the instantaneous energy consumption, there is no guarantee that the TDP is not desecrated [7]. Hence, we aim to reduce instantaneous energy dissipation and associated thermal problems in a multiprocessor system. Another challenge in scheming MCSs is enabling guaranteed service level for τ_i^1 in critical conditions. This work proposes an online energy-efficient scheduling algorithm, LMTS, to manage instantaneous power consumption of a MCS using DVFS. Also, we determine the existing laxity (i.e., the difference between the WCET of the tasks and their actual execution time).

In this work, we develop a laxity-aware mixed-critical task scheduling method to achieve timeliness, peak power management and guaranteed service level for τ_i^1 simultaneously. We create a static scheduling table for both M^1 and M^2 and implement a task migration approach that calculate and exploit the available laxity to remap the tasks to other processing elements within a constellation to decrease the thermal profile of the heterogeneous system in run time. We assess the effectiveness of LMTS approach to provide the timeliness guarantee for safety-critical applications[8]. Also, we attempt to deliver reasonable service level for τ_i^1 without violating the real-time constraints of τ_i^2.

2.EXPERIMENTAL

In this work, we aim to decrease peak power consumption and associated temperature issues in a mixed-critical application. To attain our target, we employ a laxity-aware DVFS scheme. Here, the v/f value of each processing element can be changed based on available laxity to minimize the peak power consumption.

3.SCHEDULING METHOD:

The major goal of LMTS is to decrease the peak power dissipation and the associated temperature problems of the processing elements. We utilize DVFS method to manage these issues. Equation (5) is used to define the objective function of LMTS.

$$minimize\left(\sum_{j\in cores} P_{\psi_j}, T_{max}\right), \forall\ time\ slot \quad (5)$$

Reducing v/f value of a specified processing element in task execution extends the task finishing time and it may cause deadline defilement. Also, the overhead of changing v/f value during runtime also cause deadline defilement. Equation (6) represents that the summation of the processing time of τ_i^x at v/f value ℓ on the processing element ψ_j and timing complexity of scheduling ($\mathcal{O}_s$) and changing v/f value ($\mathcal{O}_v$) should not be exceeded the deadline (d_i) of the workload in diverse levels of criticality.

$$\frac{C_i}{f_{\psi_j \ell}} + \mathcal{O}_s + \mathcal{O}_v \le d_i \begin{cases} C_i = {C_i}^1\ in\ M^1 \\ C_i = {C_i}^2\ in\ M^2 \end{cases} \quad (6)$$

LMTS comprises of two phases including offline and online. It uses the online phase to manage the instantaneous power dissipation and temperature problems; hence it is impossible to implement optimization approaches due to its augmented timing overhead. As a result, we develop a heuristic-based method. We use ODROID XU3 processor for defining the power dissipation of the workloads in offline and for executing tasks on processing elements in online phase. During offline execution, LMTS takes multiple tasks at the same time and creates clusters using the technique used in our earlier study [9]. The power dissipation of a workload can be computed by performing tasks obtained from a real-time application on ODROID XU3 processor. It is notable that executing an unknown workload during runtime is beyond the scope of this study. Mostly, the system engineer identifies the tasks and their characteristics during design time.

In this work, we generate two scheduling and mapping tables based on the workload parameters for both normal and critical modes. The

EDF algorithm is used to determine the schedule in both modes statically based on the WCETs as given in [10]. In normal mode, all the tasks are treated with the equal significance; in critical mode, all τ_i^2 are performed with higher precedence. These predefined tables are then utilized to perform workloads at runtime. This imposes a strict ordering in executing the workloads and guarantees that all deadlines are satisfied in accordance with the design time analysis. As ${C_i}^2$ of τ_i^2 are higher, not all τ_i^1 workloads may be schedulable in critical mode. To increase the quality of service, LMTS is intended to drop some τ_i^1 for creating table in critical mode.

Our method employs task parameters and these tables in online mode to manage the peak power and thermal issues in the system.

The task cluster is a group of tasks gathered together before scheduling each task. In our approach, a cluster comprises of one τ_i^2 and a group of τ_i^1 tasks. The cluster is described as $S_i = \{\tau_i^2, \tau_1^1, \tau_2^1 \dots \dots \tau_n^1\}$, where $\tau_i^2 (1 \le j \le m)$ represents the single critical task and $\tau_i^1 (1 \le i \le n)$ are low-critical task in the cluster. The base period (P_{S_i}) of the cluster S_i is computed as the greatest common factor (gcf) of period of all tasks in a particular cluster as given in Equation (7).

$$P_{S_i} = gcf\{P(\tau_i^2), P(\tau_1^1), P(\tau_2^1) \dots \dots P(\tau_n^1)\} \quad (7)$$

where $P(\tau_i^x)$ is the period of workload. The number of cluster budget replenishments in $P(\tau_i^1)$ is defined by Equation (8).

$$LO_i^j = \frac{P(\tau_i^1)}{P_{S_i}} \quad (8)$$

Similarly, the number of budget replenishments in $P(\tau_j^2)$ is calculated by Equation (9).

$$HI_j = \frac{P(\tau_j^1)}{P_{S_j}} \quad (9)$$

Now, we can compute the utilization of each cluster by E_j / P_{S_j}. The term E_j is the execution time budget that a cluster must obtain to guarantee each workload satisfy the schedulability condition. The proposed approach performs each cluster S_j with other clusters as a normal task with budget E_j and period P_{S_j}. Our online phase comprises of some function controlling units as follows:

1. **Scheduling unit**: It is the vital element that is cooperating with the other units for mapping and scheduling of the tasks.
2. **Predictor:** If there is any laxity in the system, or a cluster accomplishes earlier, the predictor determines the most apposite cluster for execution.
3. **Migrator:** If an appropriate cluster is allocated for a processing element, according to the thermal profile of the current processing element related to other processing elements, the task migrator is used to reduce the core temperature and decide whether to migrate the cluster to other processing element or not. Then, the selected v/f level for the processing element is listed. This speed is used by the DVFS controller to perform the task.
4. **DVFS controller**: This unit is used to determine an optimum v/f value for a designated constellation. Owing to mixed-criticality behaviour, the system enters into critical mode if the WCET of any one τ_i^2 surpasses its ${C_i}^1$. It should be tested by the mode changer. In this condition, the system changes its scheduling policy according to the scheduling table.

4. LAXITY CALCULATION

After computing laxity, the predictor unit selects a most appropriate cluster for mapping on the processing element where the laxity $(\mathcal{L})$ is observed. Equation (10) defines the cost function (∂_i) for each cluster.

$$\partial_i = \rho E_i + \sigma P_i \quad (10)$$

where P_i and E_i are the maximum instantaneous power and energy of the cluster, respectively. The terms ρ and σ are in the range of [0, 1]. Indeed, reduced power dissipation leads to a reduction in chip temperature. It is important that if we assume $\langle \rho, \sigma \rangle = \langle 1, 0 \rangle$, and then ∂_i only depends on power dissipation of a cluster, and not its energy. Thus, the cluster with the maximum power is designated to be executed at lower frequency to reduce the peak power. If we consider $\langle \rho, \sigma \rangle = \langle 0, 1 \rangle$, cost function only relate to energy. Hence,

the cluster with the maximum energy dissipation is selected to be executed at lower level frequency, accordingly decreasing the power dissipation. After choosing the cluster, the maximum power consumption, and it's WCETs are updated based on the available laxity and the v/f values. Furthermore, Equation (10) is exploited by a cluster that can initiate their executions earlier. The task τ_i^x can start early if it is arrived before $a_i - \mathcal{L}$, where a_i is the start time of τ_i^x. A workload can be released when all its ancestors complete their performance. Therefore, we define a condition in Equation (11).

$$T_{ri} < a_i - \mathcal{L}_{i-1} \quad (11)$$

where T_{ri} is the task release time. Let us assume the selected task τ_i^x with deadline d_i and the start time a_i that $a_i + C_i \leq d_i$. Let us assume that we have the laxity time, $\mathcal{L}_{i-1}$ created by task τ_{i-1}^x in execution. To utilize this slack for the apt task τ_i^x, generally, the task scheduler estimates the minimum appropriate core frequency using Equation (12).

$$f_i = max\left(f_{min}, \frac{C_i}{C_i + \mathcal{L}_{i-1}}, f_{max}\right) \quad (12)$$

This guarantees that only the start time of the task is earlier by $\mathcal{L}_{i-1}$ and the deadline is kept constant. Hence,

$$a_i - \mathcal{L}_{i-1} + \frac{C_i}{\left(\frac{f_i}{f_{max}}\right)} \leq a_i + C_i \leq d_i \quad (13)$$

Conversely, changing the values of v/f and selecting a suitable task and the processing element, generate timing overheads. If we neglect them for selecting the optimum frequency, it may cause timeliness desecration. Hence, $\mathcal{L}_{i-1}$ is extended by $\mathcal{O}_s$ and $\mathcal{O}_v$. By calculating the optimum frequency the start time of the appropriate cluster is updated for the static schedule.

4.1

4.2 Energy efficiency

Laxity-aware task scheduling algorithm shifts the selected task to the other processing elements without changing its deadline for reducing the chip temperature. Therefore, to decide about the task migration and finding the right processing element to transfer, we define the cost function in Equation (14).

$$\partial_c = \gamma \sum_{t=1}^{t_c} E_c(t) \quad (14)$$

We calculate the temperature of each processing element from total energy dissipation. A processing element is likely to have a lower thermal profile when its energy dissipation is lower than the others. Conversely, the difference between the total energy dissipation of the base processing element and the selected processing element should be large enough. Thus, we develop a parameter (γ) (in our experiments $\gamma = 0.9$). In Equation (14), t_c is the completion time of a task. Since we employ an asymmetric multicore system for our experimentation, each execution time and power dissipation of the clusters will be different when executing on different constellations. Albeit migration from an A7 to A15 core reduces the execution time of the task, it leads to augmented power consumption, which is inappropriate for safety-critical domain. Therefore, to reduce the instantaneous power consumption, we implement migration technique within the constellation. Since this technique is applied to a cluster that is not started yet, the migration overhead does not affect the deadline limitations. Indeed, it is negligible related to the overhead due to changing the frequency.

After executing a task, there might be a laxity or a task in the processing element that is ready to execute. All processing elements in the constellation operate at the same v/f value in an asymmetric multicore processor. As the values of v/f for both constellations are not same, it is checked on which constellation the recently completed task was executing. Next, we verify the selected v/f value of running or ready to run tasks on all processing elements of the constellation. As processing elements within a constellation operate with the identical speed, we select the optimal value of frequency to fix to the constellation. The reason for choosing the greatest minimum

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

frequency is to ensure that all tasks are completed without violating their deadline. Finally, if the designated frequency is different from constellation frequency, we allocate the new frequency for the constellation. Then, voltage will be changed automatically.

4.3 LMTS algorithm

The pseudo code of the LMTS approach is given in Algorithm 1. Workloads are assigned and scheduled up to time T according to the current schedule. In critical scenario or mode switches at time T, LMTS assigns and executes the remaining workloads according to the new schedule from time T to the end of the application period. The time is equally splitted into multiple time slots (S_T), and the scheduling algorithm will assign workloads into cores only at the commencement of every time slot.

Algorithm 1: Laxity-aware mixed-critical task scheduling

Input: Set of processing element $\psi = \{\psi_0, \psi_1, \ldots \psi_{\lambda-1}\}$, clusters, time ($T$), schedule up to the time T (I_{sch}), task ready queue (Q_{ready}), count(χ)=0.
Output: Final schedule (F_{sch})

```
1   procedure Scheduling
2     for S_T = T to PERIOD do
3       A_ready ← ∅
4       Pop a task from Q_ready and push it into A_ready
        when key = S_T and Q_ready ≠ ∅;
5       if Q_ready = ∅ and A_ready = ∅ then
6         return F_sch
7       end if
8       if A_ready = ∅ then
9         continue
10      end if
11      TaskForExecution←Sort (A_ready, decreasing
        order);
12      CoresToExecute←Sort (ψ, increasing order);
13      for task in TaskForExecution do
14      for core in CoresToExecute do
15        TempTime ← WCET of the task
16        χ ← 0
17        TempSch ← F_sch
18        TempPower ← A_p_max
19        while TempTime > 0 do
20        if TempSch (S_T+ χ, ψ) ≠ ∅) and
          TempPower (S_T+ χ) + TaskPower ≤ TDP
          then
21          TempSch (S_T + χ, ψ) = task
22          TempPower (S_T + χ) +=TaskPower
23          TempTime = TempTime -1
24        end if
25        χ = χ + 1;
26        end while
27        if S_T+ χ ≤ TaskDLine then
28        I_sch ← TempSch
29        A_p_max ← TempPower
30        CoresToExecute ← Sort (ψ, increasing
          order);
31        TaskSch ← true
```

```
32                  break
33                  end if
34              end for
35              if TaskSch == false then
36                  return unschedulable clusters
37              end if
38              end for
39          end for
40      end procedure
```

At every time slot, LMTS algorithm creates an array for ready tasks (A_{ready}) and then it pop ups all elements from Q_{ready}, where their key is equal to the current time slot. This indicates all previous workloads in the ready queue have completed their performance. If Q_{ready} and A_{ready} are both empty, this approach delivers the final schedule (F_{sch}) since it effectively performs all workloads. If there is no ready task to be executed currently ($i.e., A_{ready} = \emptyset$) but, $Q_{ready} \neq \emptyset$ then the procedure moves to the subsequent time slot. Our approach sorts the ready tasks in decreasing order based on their energy dissipation. The energy dissipation of each workload ($E_{\tau_i^x}$) is measured using Equation (15).

$$E_{\tau_i^x} = P_{\tau_i^x} \times C_i^x \qquad (15)$$

where $P_{\tau_i^x}$ and C_i^x are the peak power dissipation, and the WCET of workload τ_i^x, respectively. The peak power of a workload can be calculated by executing them on a test bed. The power dissipation of the system must never surpass the TDP limit to evade the overheating issues. In this work, we assume fixed power dissipation for workloads at offline mode, which is equal to its peak power dissipation, to satisfy the TDP limit in the critical situation. Furthermore, increasing in energy causes a rise in core temperature. Accordingly, we assign a workload with more energy dissipation to a processing element with lower temperature. Next, the LMTS sorts the processing elements in the increasing order based on their energy consumption. A processing element has more priority for workload allotment if it has lower energy consumption (i.e., tends to have a lower thermal profile). Then, LMTS maps workloads to the processing elements sequentially. Hence, for every workload, the algorithm designates a processing element from the sorted list and executes the workload on the free slots of the processing element. The peak power dissipation must be lower than the TDP limit of the system; so, we create an array known as A_{p_max}, which stores the peak power dissipation in each time slot. LMTS verifies A_{p_max} and TDP limit before mapping a workload on a processing element. If the workload is finished before its deadline, LMTS updates the schedule I_{sch}, A_{p_max}, and scheduling condition of the workload (TaskSch). It also sorts the processing elements again since the energy of one processing element has varied, and starts to execute the subsequent workload. If there is a deadline violation on the designated processing element, the LMTS picks another processing element and executes tasks on that processing element. Conversely, if the deadline of one workload is violated in all processing elements, it returns an error message such as "un-schedulable tasks".

5.RESULT AND DISCUSSION

To evaluate the performance of LMTS approach, we conduct several experiments on ODROID XU3 processor (ARM big. LITTLE multiprocessor) system as given in Figure 1. Since it supports various v/f settings, we consider the effect of different v/f levels. To perform experiments, we engender random tasks employing the technique given in and execute these tasks on the processor with maximum frequency and calculate the energy dissipation from sensing elements used on the kit. As the v/f scaling is employed to the whole system, the energy consumption at other lower level core speeds can be measured by varying the frequency of the system. Moreover, we analyzed the effect of number of processing elements by performing tasks on 1 to 8 processing elements. We run each trial 1000 times with different parameters (i.e., deadline, actual execution times, WCETs, etc.) and calculate the average results. We found that the higher energy dissipation of tasks in the range of [2.986, 6.856] W in big cores and [0.492, 0.923] W in LITTLE cores.

Journal of Theoretical and Applied Information Technology
31st May 2023. Vol.101. No 10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

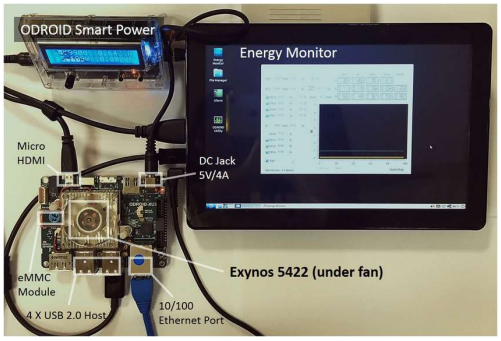

Figure 1: Experimental Set-Up In ODROID XU3 Board

To assess the effectiveness of our proposed algorithm, we employed random task generation proposed by Medina et al. [7]. The applications are created with 20, 40, 60, 80, and 100 workloads (n), where 10% to 70% of them are high-level workloads. Also, we consider 5% to 25% edge percentage (k) in this work. Edge percentage is defined as the possibility of having edges from one workload to other workloads. We consider the normalized system utilization $\frac{U}{\lambda}$, where U is the system utilization in critical mode, and λ is the number of processing elements in the system. The normalized system utilization is anticipated in the range of [0, 1].

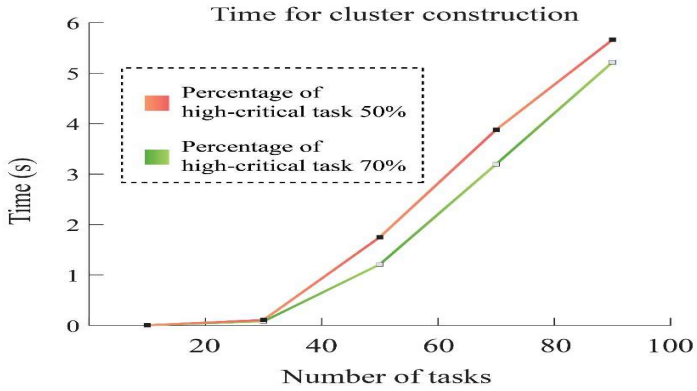

Figure 2: Time For Cluster Generation By Varying Number Of Tasks

We also assess LMTS by comparing its performance with other state-of-the-art approaches in the literature using a real-time application, vehicle cruise controller (VCC) . VCC contains 32 workloads, where 34% of them are high-level workloads. Besides, the value of k for this application is 7%. Initially, we assess offline cluster generation time by changing the parameters n and k. The time of cluster generation is observed on a system with an Intel core-i5 1.3 GHz speed processor. The time taken for cluster generation hinges on the number of tasks and faults. Figure 2 illustrates the impact of the number of workloads and the percentage of high-level workloads in the input dataset on a system with number of fault is 3. Similarly, Figure 3 illustrates the impacts of the number of fault occurred on a system with number of tasks is 40. These figures illustrate that by varying the number of workloads or faults, the time of cluster generation is growing exponentially. Though the offline cluster generation time is comparatively high for large applications, the online overhead is negligible and constant for all applications. It is obvious that our approach can create each node of a cluster simultaneously to minimize the time complexity. For instance, if we

have a system with 4 processing elements, the construction time is about four times faster than a system with single processing element.

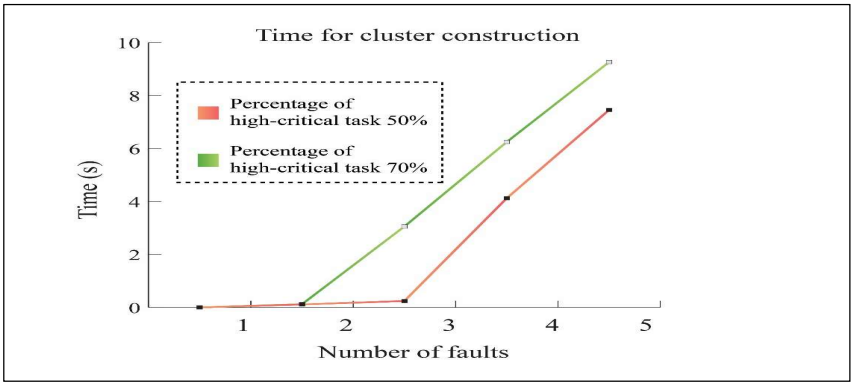

Figure 3: Time For Cluster Generation By Varying Number Of Faults

To measure the system temperature, we perform the tasks on Core 2 and 3 that usually realize maximum temperature due to their closeness to the memory and other components. The board contains sensors to monitor the temperature of every A15 core and to calculate the power dissipation of each constellation. Therefore, the power and temperature values are measured from these sensors. Figure 4 demonstrates the power trace of the constellation with A15 cores during runtime using LMTS and a state-of-the-art method .

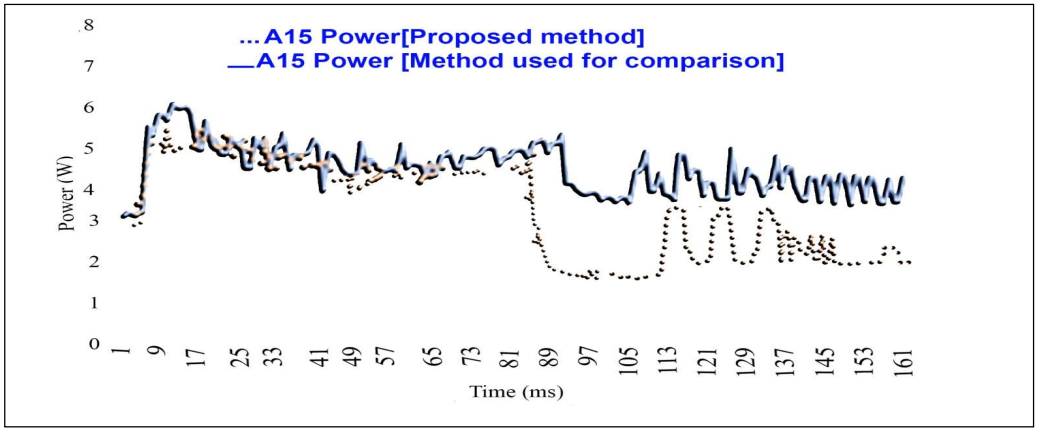

Figure 4: Power Trace Of The Constellation With A15 Cores

The temperature traces of Core 2 and Core 3 are depicted in Figures 5 and 6, correspondingly. The core temperature has been decreased by LMTS considerably. After applying our algorithm and reducing the v/f levels, the temperatures of the processing elements are reduced. Hence, LMTS will be more effective and provide a significant performance improvement whenever more tasks are performed on more number of cores.

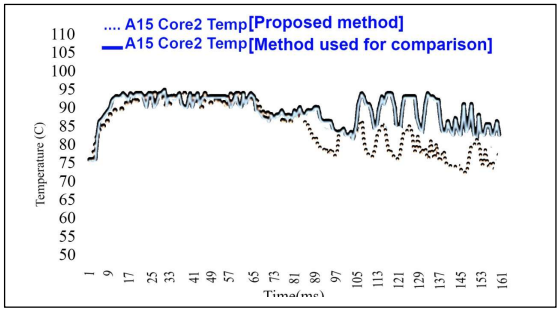

Figure 5: Temperature Trace Of A15-Core2

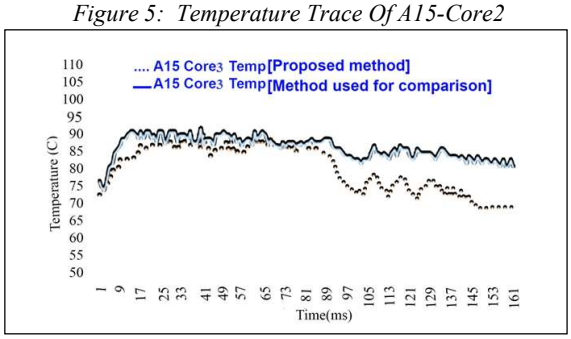

Figure 6: Temperature Trace Of A15-Core3

We evaluate the performance of LMTS under three different scenarios as showed in Figures 7- 9, in which the results are normalized to [8-10]. Mostly, as the applications become complex (e.g., having higher system utilization or numerous tasks), it is very problematic to achieve the substantial reduction in peak power dissipation, temperature, and energy consumption. It is observed that the power consumption of the system is reduced when the number of cores is increased. The proposed migration technique is employed to reallocate the tasks more uniformly to the processing elements at runtime based on their energy consumption.

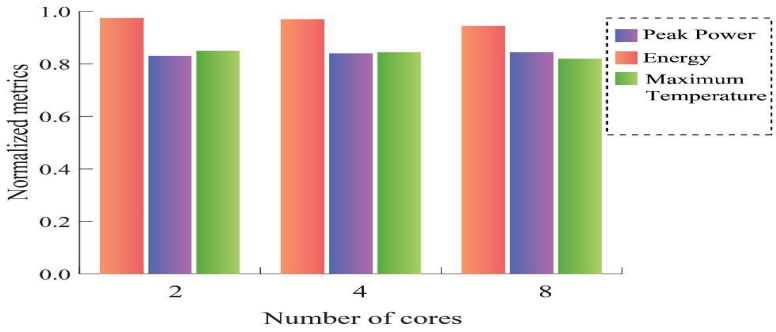

Figure 7: Impact Of Number Of Processing Elements On System Parameters

As LMTS only considers drop in peak power dissipation for each processing element autonomously, it is very difficult to realize a similar drop in power when a fewer number of processing elements is employed. The variation in maximum power is substantial by growing the number of processing elements as shown in Figure 4. Using our approach we can achieve 6.35%, 16.28%, and 21.12% of drop in the maximum power, energy, and temperature, respectively [11-15].

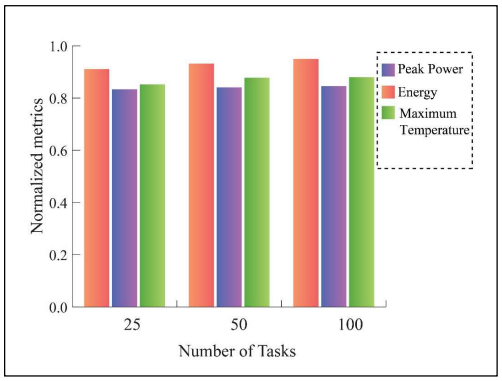

Figure 8: Impacts of number of tasks on system parameters

The effectiveness of LMTS depends on the available laxity in online mode and the possibility of assigning them to the tasks. Therefore, if there is small laxity perceived due to the type of the application in terms of the number of tasks and system utilization, the system parameters including peak power, energy, and temperature drop are very small. In Figure 10, if we increase the utilization,

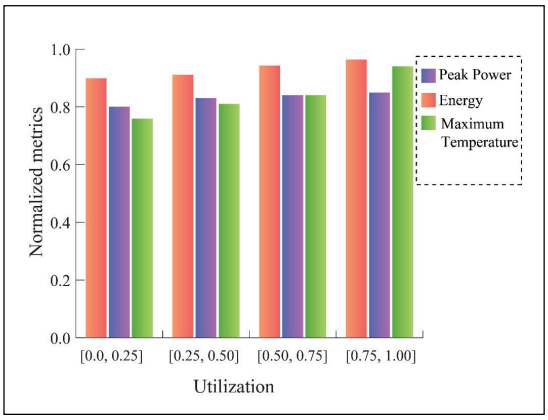

Figure 9: Varying utilization bound

6. CONCLUSION

This work proposes a laxity-aware task scheduling algorithm for mixed-critical system to support correctness, energy management, timeliness and failsafe service level. The proposed approach reduces power dissipation of the multicore processor cnsiderably by applying DVFS method. Our algorithm accepts multiple worklodas simultaneously and create task clusters with one high-level task and a set of low-critical tasks. It computes the extant laxities efficiently and determines the most appropriate cluster to exploit that available laxity by considering its impact on power dissipation and related temperature issues of the system. At the same time, changing the core speed, allocating a right cluster for residual laxity, and choosing a right processing element for task migration in online mode are difficult endeavors and cause deadline defilement which is not suitable for safety-critical applications. Therefore, we develop an runtime scheduler with task migration technique and DVFS to reduce power dissipation and related thermal issue by scheduling tasks at runtime. We also defined cost functions to select the right task cluster to allocate the right processing element by scaling its v/f value or to transfer it to another element. We evaluate the effectiveness of the idle time of the processing element among two successive releases of tasks is decreased. The tasks also tend to execute longer time. Accordingly, the size of laxities that can be employed at online mode is constrained. LMTS provides a minimum 4.215% and a maximum 8.719% of reduction in peak power consumption in this case.

the proposed LMTS method using ODROID XU3 processor with real-time task sets.

Declaration:

Ethics Approval and Consent to Participate:

No participation of humans takes place in this implementation process

Human and Animal Rights:

No violation of Human and Animal Rights is involved.

Funding:

No funding is involved in this work.

Conflict of Interest:

Conflict of Interest is not applicable in this work.

Authorship contributions:

There is no authorship contribution

Acknowledgement:

There is no acknowledgement involved in this work.

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*, vol. 2, 2016, pp. 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*, vol. 517,2017, pp. 397-418, DOI:10.1007/978-981-10-3174-8_35.

[3]. K.Nagalakshmi, and N. Gomathi, " Criticality-cognizant Clustering-based Task

Scheduling on Multicore Processors in the Avionics Domain," *International Journal of Computational Intelligence Systems*, vol. 11,2018, pp. 219–237, DOI:10.2991/ijcis.11.1.17.

[4]. Z.Qian, W. Jianguo, X. Fei and H. Shujuan, " Research on semi-partitioned scheduling algorithm in mixed-criticality system," *Cognitive Robotics*, vol. 1,2021, pp. 214-221. doi: **10.19304/J.ISSN1000-7180.2022.0427**

[5]. 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, pp. 10-12. DOI:10.1109/EMSOFT51651.2020.9244032.

[6]. J.Simó, P. Balbastre, J.F. Blanes, J.L. Poza-Luján and A. Guasque, " The Role of Mixed Criticality Technology in Industry 4.0.," *Electronics*, vol. 10,2021, pp. 226. DOI:10.3390/electronics10030226.

[7]. M.Ansari and S. Safari, " Peak Power Management to Meet Thermal Design Power in Fault-Tolerant Embedded Systems," *in IEEE Transactions on Parallel and Distributed Systems*, vol. 30,2019, pp. 161-173, DOI:10.1109/TPDS.2018.2858816.

[8]. S.Hosseinimotlagh and A. Ghahremannezhad, "On Dynamic Thermal Conditions in Mixed-Criticality Systems," *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS),* 2020, pp. 336-349, DOI:10.1109/RTAS48715.2020.00009.

[9]. 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*, vol. 117,2021, pp.102090, DOI:10.1016/j.sysarc.2021.102090.

[10]. I.Ali, " Reducing Dynamic Power Consumption in Mixed-Critical Real-Time Systems, "*Applied Sciences*, vol. 10, 2020, pp.7256. DOI:10.3390/app10207256

[11]. K.V.Kumar and A.Rajaram, "Energy efficient and node mobility based data replication algorithm for MANET," *International Journal of Computer Science*, *2019*.

[12]. A.P.Sridevi and A.Rajaram, "Efficient Energy Based Multipath Cluster Routing Protocol For Wireless Sensor Networks". *Journal of Theoretical & Applied Information Technology*,vol.*68,2014.*

[13]. A.Rajaram and S.Kannan,"ENERGY BASED ROUTING ALGORITHM FOR MOBILE AD HOC NETWORKS," *Journal of Theoretical & Applied Information Technology*, Vol.*61, 2014.* **DOI:** 10.1109/WD.2008.4812884

[14]. Rajaram, A. and Sathiyaraj, K., 2022. An improved optimization technique for energy harvesting system with grid connected power for green house management. *Journal of Electrical Engineering & Technology*, *17*(5), pp.2937-2949. https://doi.org/10.1007/s42835-022-01033-2

[15]. Kumar, K.V. and Rajaram, A., 2019. Energy efficient and node mobility based data replication algorithm for MANET. *International Journal of Computer Science*, *2019*.