PFA-TP3M: Permanent Fault Aware Two-Phase Peak-Power Management in Fault-Tolerant Multi-Core Systems

Pargol Hatefi and Mohammad Salehi *

Posted Date: 3 January 2024

doi: 10.20944/preprints202401.0103.v1

Keywords: Peak-Power; Replication; Fault-Tolerance; Multi-Core Systems

Article

PFA-TP3M: Permanent Fault Aware Two-Phase Peak-Power Management in Fault-Tolerant Multi-Core Systems

Pargol Hatefi [1] and Mohammad Salehi [2,*]

[1] Department of Computer Engineering, University of Guilan, Rasht, Guilan, Iran, pargolhatefi@gmail.com
[2] Department of Computer Engineering, University of Guilan, Rasht, Guilan, Iran, mohammad.salehi@guilan.ac.ir

Abstract: It is necessary to manage the power consumption and use fault-tolerance techniques to meet high reliability in Real-Time Embedded Systems, . Hence, in recent years, this is the main reason for providing an ideal approach to decrease power consumption through the worst-case behavior of instantaneous power management (peak-power management), while observing the Thermal Design Power (TDP). Although the multi-core chips in real-time embedded systems provide great opportunity for implementation of Two-phase Triple modular redundancy to meet high reliability, the energy overhead management of concurrently executing tasks is one of the challenges facing designers. In this article, we propose a scheme named Permanent Fault Aware, Two-Phase Peak-Power Management (PFA-TP3M), for scheduling real-time tasks in multi-core systems to eliminate peak power overlap from concurrent running tasks to maintain peak power consumption at the chip TDP. and through mapping different parts of each task on separate cores, prevents consecutive conflicts on cores, which have faced permanent faults to design the fault-tolerant system. Our simulation results show that our proposed scheme provides up to 700x fault tolerance increase (630x on average) compared to the state-of-the-art power management algorithm, an achievement costs up to 14%. (average 11%) increase the length of the task graph compared to it.

Keywords: peak-power; replication; fault-tolerance; multi-core systems

I. Introduction

Many-core processors introduce an excellent potential for parallel computing to increase system performance. However, with the increase of the number of cores the system power consumption may increase beyond the specified Thermal Design Power/Point (TDP) for a chip. TDP determines the system power budget which is the maximum power that can be consumed by the chip. TDP violations cause some cores to reboot automatically or reduce system performance. Therefore designers face the challenge of deciding how to use TDP constraints in real-time embedded systems. An effective solution to meet TDP limitations is to reduce maximum power consumption. [4, 6, 13]. Meanwhile, hard real-time systems presently appear in various application areas, computer systems optimized for the execution of such applications should offer high criteria of reliability, because a system failure may result in costly disastrous events. [13]. Due to technology scaling, transient faults and permanent faults are already the leading cause of failures in digital systems. Hence, the system should tolerate such failures. Task redundancy is a common approach to provide desired reliability and achieve fault tolerance, whereby an N-module redundancy set consists of N copies of each task. TMR is a well-known example of N-module redundancy, consisting of three identical copies of a task whose results are voted on.

In this paper, we first show how the TP3M technique could manage the peak power consumption of the TMR technique on multi core systems by eliminating the overlap of performed tasks (see **Motivational Example**). Overlapping prevention can keep the maximum power consumption below the chip TDP. Then, we propose a two-phase peak power management scheme

 doi:10.20944/preprints202401.0103.v1

(PFA-TP3M) that is capable of tolerating permanent system faults. The PFA-TP3M scheme aims at tolerating task executions from permanent faults. To do this, considering the power traces of tasks', at first, the tasks are partitioned into parts where different parts of each task have different peak power values. Then two-phase TMR is used for scheduling the partitioned tasks. In these two-phase scheduling, a policy called Peak Power and Permanent Fault Aware-Longest Task First (P3FA-LTF) is designed to tolerate permanent faults and reduce maximum power consumption. In the first stage of the proposed scheme, more than half of copies for each task is scheduled on separate cores based on P3FA-LTF policy. The remaining copies of each task are not required if during first phase no fault occurs. Otherwise, in the second stage, to perform majority voting, the remaining copies of the task are scheduled. In this stage, the policy P3FA-LTF is used to manage peak power consumption while tolerates permanent fault. In summary, the PFA-TP3M scheme tries to tolerate permanent faults through executing separate copies of each task on separated cores, while it manages peak power consumption. In order to reduce peak power consumption, the tasks with higher power values overlaps with the tasks that have lower power values. So, the total peak power is kept below the chip TDP.

Motivational Example: This example, provides some insight into show the effect of tolerating permanent fault based Peak Power Aware Longest Task First (PPA-LTF) policy. A quad-core chip with 3W of TDP is considered that executes six tasks {T_1, T_2, T_3, T_4, T_5, T_6} of an application task graph. The dependencies between the tasks is shown in Figure 1a where the worst-case execution time of each task is written above them. Depending on tasks' characteristics and computational load, different tasks' have different power traces. However, in our proposed method, to have an easier presentation, a constant value is assumed for each tasks' peak power so each task consumes 1. 2W of power when executed. In this example, a TMR system (i.e., NMR with N=3) is considered where there each task has three copies and their results are compared to perform a complete majority voting. Figure 1 indicates three possible schedules based PPA-LTF policy to execute task graph which manage peak power and meet TDP constraints. From the beginning of the execution frame, cores with the lowest utilization are prioritized. Two copies of each task are scheduled on priority cores to manage maximum power consumption and stay within the chip TDP. Then, a third copy of the same task is scheduled on the core with the lowest utilization such that maximum power consumption is kept at the TDP of the chip where its execution started at the end of the cycle. terminates the execution of the remaining two copies. In this scenario, which has been presented in [1], two copies of the same task (T_6) are scheduled on a same core (core 1). Assume core 1 is corrupted as a result of permanent fault when the first copy of T_6 is executed, so it produces wrong result that does not match with the result of core 2. Subsequently, the execution switches to the second phase and executes the third copy of T_6. Since the third copy of T_6 is scheduled on core 1, again, it produces a wrong result, therefore, the execution fails in facing a permanent fault, as shown in Figure 1b. For scheduling this task graph, another possible execution scenario is shown in Figure 1c, where the chip TDP is 4W and higher than the chip TDP of first scheduling scenario. Since each task consumes 1. 2W per core, in this case, the total maximum power consumption is equal to 3. 6 W (i.e., less than the chip TDP). However, more copies of the same task can be scheduled on the same core (i.e., two copies of task T_3 on core 2 as well as two copies of task T_5 on core 4). Therefore, TP3M scheme is unable to tolerate permanent faults. In Figure 1d, a scheduling method is presented which tolerates permanent faults since it does not execute same copies of a task on a same core. In this work, each copy of each task is scheduled on separate cores with the lowest utilization. So if the core fails due to a permanent error, the system can tolerate that.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 January 2024 doi:10.20944/preprints202401.0103.v1

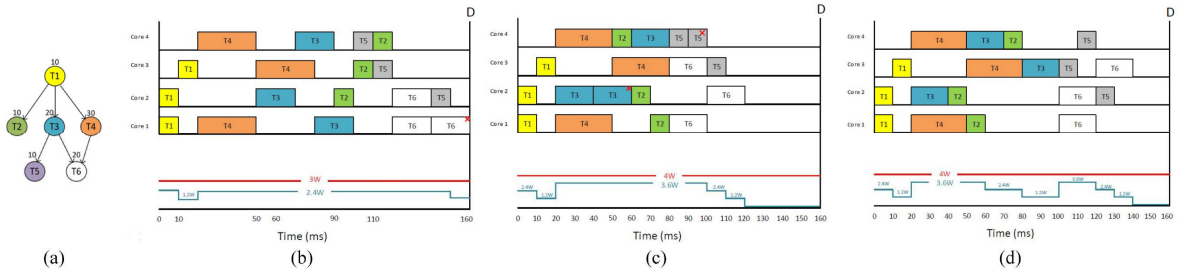


Figure 1. Motivational example of permanent fault tolerant of a TMR system (i.e. NMR with N=3) on a multi-core system with 4 cores: a) An example task graph, b) Without permanent fault tolerant, TDP=3, c) Higher TDP without permanent fault tolerant, TDP=4, d) Permanent fault aware, TDP=4 (our scheme).

Objective: The objective of this paper is to present a Permanent Fault Aware Two-Phase Peak-Power Management (PFA-TP3M) scheme in the event of a fault. PFA-TP3M is a method that uses Two-Phase Peak-Power Management (TP3M) technique to manage peak power overlap between concurrently executed tasks in real-time multi-core embedded systems so that TDP constraints are respected. In this method, the focus is on scheduling a task graph in two phases. In each phase, separate copies of the same task are scheduled on separate cores. While the copies of each task are scheduled based on Peak-Power and Permanent Fault Aware Longest Task First (P3FA-LTF) policy. In P3FA-LTF policy, separate redundant copies of each task are executed on separate cores, in order to tolerate permanent faults. To the best of our knowledge, the fault techniques for real-time multi-core embedded systems only try to tolerate transient faults.

Contributions: The main contributions of this paper are:

- Proposing a permanent fault aware scheduling method that tolerates permanent faults.
- Enabling TP3M scheme to manage peak power overlaps between concurrently executing tasks.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III describes system and application model. In section IV, the PFA-TP3M scheme is presented in details. The experimental results are presented in section V. Finally, the paper is concluded in Section VI.

II. Related Work

A. Energy Consumption management

Several research works, e.g., References [2] and [3], have addressed voltage regulation techniques to reduce the power consumption of N-module redundancy (NMR) in multi-core embedded systems in hard real-time. To have low energy-overhead of N-modular redundancy (NMR), Salehi et al. [2] have proposed a technique where the system operation is divided into two phases, indispensable phase and on-demand phase. In indispensable phase, half-plus-one copies for each task are executed. When there is no fault, the remaining copies are not required and the result of execution must be identical. Otherwise, on-demand phase is required and the remaining copies must be executed to perform a complete majority voting. This approach can decrease one-third of the energy consumption of conventional NMR while conserving its reliability against transient faults. However, transient faults are not the only faults that can occur in execution time, one other fault types are permanent faults that can be continued to the end of execution. Since, in on-demand phase, the redundant copies of each task can be scheduled on the same core with the copies of indispensable phase, this technique can fail in the face of permanent fault. The work in [3] has proposed an energy efficient permanent fault tolerance scheme to tolerate two phase NMR technique in the face of both transient and permanent faults in hard real-time systems. The key idea of this approach is to schedule separated redundant copies of each task on separated cores in both indispensable phase and on-demand phase where the faulty cores are detected and re-assign their tasks to functioning ones.

Although passive redundancy is well-established technique to tolerate transient and permanent faults for multicore platforms, but it incurs peak-power overlaps of concurrent task executions. Peak power overlaps can violate chip TDP where violating TDP can reduce core performance. To meet the TDP constraint, an efficient solution is reducing the peak power consumption in multicore embedded systems.

B. Peak Power Reduction

Several studies have focused on reducing peak power consumption under real-time constraints [4, 5, 6, 7]. Lee et al. [4] have presented a scheduling algorithm to reduce chip-level peak power consumption for real-time tasks. The idea of this algorithm is restricting the concurrent execution of tasks that are assigned to different cores. Lee et al. [5] have proposed a scheme for task-graph models that schedules the tasks by considering data dependency information while reducing the maximum energy consumption. Munawar et al. [6] have presented a scheme for periodic tasks on multi-core systems which manages the peak power consumption through scheduling the sleep cycles for each active core. Another related scheme for periodic real-time tasks on multi-core systems is presented in [7]. Pagani et al. [7] have worked on both energy minimization and peak power reduction. These proposed schemes have concentrated on maximum power reduction without considering any fault tolerance techniques to manage transient and permanent faults.

C. Reliability-Aware Peak Power Management

Reliability and low power consumption are two main objectives in designing real-time embedded systems [1], [8], [9] and [10]. Ansari et al. [1] have presented a reliability-aware peak power management algorithm for periodic task graph models in hard real-time multi-core embedded systems. Their algorithm manages peak power overlaps between concurrently executing tasks where the system reliability is preserved at an acceptable level and the power consumption is kept below the core TDP constraints. Ansari et al. [8] have proposed a primary-backup scheme called Peak-Power-Aware Primary-Backup technique for real-time tasks on core pairs in multi-core systems. This technique schedules original and redundant copies of tasks respectively, with maximum-peak-power-first (MPPF) and maximum-peak-power-*last* (MPPL) policies. Khaksar et al. [9] have presented a method for creating replicas of periodic real-time tasks called ReMap, which consists of three phases: RA-LU Mapping, MPA-EDF scheduling, and RPPA-DVFS Energy Management which are executed sequentially and in the event of successful execution of the previous phase. Standby-sparing is one of the hardware replication schemes that provides high reliability while managing the peak power consumption which is presented in [10]. Ansari et al. [10] have considered a standby-sparing system where the main tasks are scheduled by their proposed PPA-EDF policy on primary cores while the backup tasks are scheduled by their proposed PPA-EDL policy on spare cores to meet the chip TDP constraints. Recently, heterogeneous multi-core systems provide an effective solution wherein every core can have an individual voltage but it is costly for implementation [4]. Due to the heterogeneity, the worst-case execution time and the energy/peak power consumption of tasks change according to the task-to-core mapping, presenting a new challenge for energy minimization and peak power reduction [11].

Proposed approaches in [11, 12] focus on managing peak power consumption below the TDP constraint and the system reliability at an acceptable level in heterogeneous multi-core embedded systems. In [11] the periodic real-time applications are scheduled on different types of cores with voltage/frequency variations to optimize TDP/TSP-constrained energy-reliability for heterogeneous multi-core embedded systems. Ansari et al, [12] have proposed a Dynamic Frequency Scaling (DFS) method to reduce peak power consumption and improve the system reliability. DFS method is based on an island architecture that each island has a separate supply voltage, while each core of the island can operate at a different frequency.

In general, the previous works in the context of multi-core embedded systems either propose energy consumption management without considering peak power like [2] and [3] or consider peak power reduction without considering permanent fault tolerance like [4, 5, 6, 7]. In this scheme, a peak

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 January 2024 doi:10.20944/preprints202401.0103.v1

power management technique (TP3M) is used to achieve peak power reduction real-time multi-core systems and proposed a scheme to tolerate permanent fault to achieve high reliability in multi-core embedded systems.

III. System Models and Assumptions

A. System and Task Model

In this paper, a multi-core system with m cores $C=\{C_1, C_2, \ldots, C_m\}$ is considered, that executes applications with hard real-time requirements. Each application consists of n dependent tasks $\Phi=\{T_1, T_2, \ldots, T_n\}$, the scheduling of tasks is based on the precedence which modeled by a task graph. A sample task graph has some nodes and vertices (see Figure 1a). Each node in the task graph means a task while the data dependencies between the tasks is represented by the directed edges. The worst-case execution time for the task T_i has been written above each node.

B. Power Consumption Model

A system-level power model, P_{total} is assumed where total power consumption is consisted of a static and a dynamic component. The static power, P_s, is consumed even when no computation is carried out which is consisted of the reverse and sub-threshold leakage power. The dynamic power, P_d, includes the effective switched capacitance, supply voltage and operational frequency. The total power of each core can be written as [Salehi] [Ansari]:

$$P_{total} = P_s + P_d = I_{sub}\, V + C_{eff}\, V^2{}_{dd}\, f \tag{1}$$

IV. Our Proposed Method

A. Overview

In this section, at first, our proposed Permanent Fault Aware Two-Phase Peak-Power Management (PFA-TP3M) scheme is explained, then an example is used to illustrate how it works.

Peak-Power and Permanent Fault Aware Longest Task First (P3FA-LTF): The idea of P3FA-LTF scheduling is based on the power profile of the tasks and enables task replication to achieve high reliability in multi-core embedded systems under TDP constraints. To the best of our knowledge, the fault tolerance techniques for multi-core embedded systems that have been presented in the literature try to tolerate the transient faults and do not consider permanent faults. However, most of the fault-tolerant techniques use hardware mechanisms which have area overhead. In other hand, the power management techniques for fault-tolerant systems only try to reduce the average power and do not consider peak power constraints. In this paper, a scheduling algorithm is proposed for real-time applications on multi-core embedded systems which reduces the peak power consumption while tolerating permanent faults. The PFA-TP3M scheme consists of mandatory and conservative phases to take the advantages of fault-free scenario. Half-plus-one copies of each task are scheduled based on P3FA-LTF policy on separate cores in the mandatory phase, in order to perform a complete majority voting, the remaining copies must be scheduled in the conservative phase based on P3FA-LTF policy on separate cores. Separate cores for each redundant copy of tasks, tolerates permanent faults. When no fault occurs, the remaining copies of each task are not executed, which have the power saving result as conventional NMR in the reference [2].

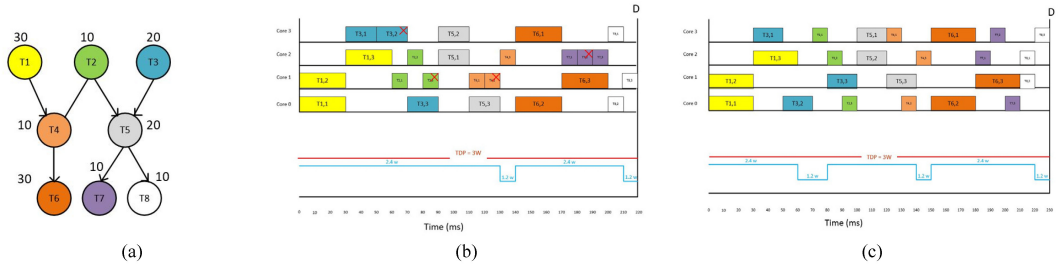
As the final discussion of our proposed method overview, the time required to meet TDP is discussed. Since some tasks are shifted to the next time slots to reduce peak power and tolerate permanent faults simultaneously, more time is needed for execution. More time increases the scheduling length. Our proposed scheme incurs more time overhead as compared to other scheme, the reference [1]. For example, in the motivational example, the [1] scheme (Figure 1c) schedules tasks in t=120ms, however, it does not tolerate permanent faults. While our proposed method (Figure 1d) schedules tasks in t=160ms. Therefore, for meeting TDP, removing peak power overlaps and tolerating permanent fault simultaneously, our proposed method must consider more time slots.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 January 2024 doi:10.20944/preprints202401.0103.v1

B. Illustrative Example

In the following, an example illustrates how our proposed scheme works. To have an easier presentation, a constant value is assumed for each tasks' peak power so that each task consumes 1.2W of power during its execution. However, our proposed scheme works for more complex and larger task graphs. In this example, a quad-core chip with 3W TDP is considered that executes an application task graph with 8 tasks {T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8} with a TMR system (i.e., NMR with N=3). Figure. 2 shows the illustrative example of our proposed scheme for a given task graph (Figure 2a) using list scheduling with our proposed policy. Dependencies between the tasks is shown in Figure 2a where each tasks' worst-case execution time at the maximum supply voltage and the maximum operational frequency is placed above each task. Two copies of each task for the mandatory phase with one copy of each task for the conservative phase is scheduled in Figure 2b and Figure 2c. Figure 2b shows a scheduling based PPA-LTF policy while our proposed scheme based P3FA-LTF policy is shown in Figure 2c. In the mandatory phase, based on the lowest utilization and apart core for each copy of tasks policy, two copies of the task T_1 are respectively scheduled on C_0 and C_1 from the beginning of the execution. In the time slots between 30ms and 60ms on the C_3, the third copy of T_1 is scheduled based on the lowest utilization and apart core for each copy of tasks policy. Then, T_3 is selected based on the level-based longest task first policy, and two copies of T_3 are scheduled on C_3 and C_0 in the time slot [30ms, 50ms] and the time slot [50ms, 70ms], respectively. Of course, based on PPA-LTF policy in [1], the second copy of T_3 can be scheduled in time slots between 50ms and 70ms on the schedule of C_3 (Figure 2b), but scheduling two copies of T_3 on a same core can violate our proposed permanent fault tolerating policy. The third copy of T_3 is scheduled on C_1, in the time slots between 80ms and 100ms. For the next selected task T_2, one copy is mapped to C_3 in the time slots between 70ms and 80ms where, another copy is mapped to C_2 in the time slots between 80ms and 90ms. After scheduling two copies of T_2, third copy of T_2 is scheduled in the time slots between 90ms and 100ms on the schedule of C_0 to achieve three results for performing a complete majority voting. Of course, based on PPA-LTF policy in [1], the third copy of T_2 can be scheduled in time slots between 80ms and 90ms on the schedule of C_1 in conservative phase (Figure 2b), but this can violate our proposed permanent fault tolerating policy. Then, the proposed P3FA-LTF algorithm selects T_5 and maps two copies of T_5 to C_2 and C_3 separately and schedules T_5 in the time slots between 100ms and 120ms on the schedule of C_2 and C_3. In conservative phase, the third copy of T_5 is scheduled on the schedule of C_1 in the time slots between 120ms and 140ms after the execution of the second copy. For the next selected task T_4, two copies of this task are scheduled on C_3 and C_0 in the time slot [120ms, 130ms] and the time slide [130ms, 140ms], respectively. Although, based on PPA-LTF policy, the second copy of T_2 can be scheduled in time slots between 120ms and 130ms on the schedule of C_1 (Figure 2b), but this can violate our proposed permanent fault tolerating policy. At the third level of the task graph, to schedule three tasks T_6, T_7 and T_8, based P3FA-LTF policy, at first, T_6 is selected and two copies of it are scheduled on the schedule of C_3 and C_0 separately at the time slots between 150ms and 180ms. Then, in conservative phase, another copy of T_6 is scheduled on C_1 in the time slots between 180ms and 210ms. For the next selected task T_7, two copies of it are placed on C_2 and C_3 in the time slot [180ms, 190ms] and [190ms, 200ms], respectively. After scheduling of two copies of T_7, the third copy is scheduled on the schedule of C_0 in the time slots between 200ms and 210ms. Of course, three copies of T_7 can be scheduled on C_2 based PPA-LTF policy (Figure 2b), but this can violate our proposed permanent fault tolerating policy. After scheduling of T_6 and T_7, the algorithm selects T_8 and schedules two copies of T_8 on the schedule of C_2 and C_1 separately in the time slots between 210ms and 220ms. Finally, the third copy of T_8 is scheduled in the time slots between 220ms and 230ms. Figure 2c shows the final schedule of Figure 2a task graph, based our proposed P3FA-LTF policy where the peak power consumption of the system is kept below the chip TDP constraint and permanent fault is tolerated.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 January 2024 doi:10.20944/preprints202401.0103.v1



(a) (b) (c)

Figure 2. An example of how our proposed PFA-TP3M method works on a multi-core system with 4 cores: a) An example task graph, b) Scheduling based PPA-LTF policy, c) Scheduling based our proposed P3FA-LTF policy.

V. Experimental Evaluation

In this section, the effectiveness of our proposed PFA-TP3M scheme is evaluated, which consists of the comparison between our proposed scheme and TP3M scheme. In our evaluation, various task sets including real-time applications of MiBench Benchmark suite [14] running on a target multi-core chip are simulated.

In our simulation, several task dependencies graphs with various values of simulation parameters, i.e., execution time, min/max power consumption (see Table 1) are considered. Task sets with 10, 20, 40 and 100 tasks are selected randomly from Table 1 and simulated on processors with 4, 8, 12 and 16 processing cores. For each data point, 10 task sets randomly are generated and the average results are reported in Figure 3 and Figure 4.

Previous work has studied peak power issues and reliability in embedded systems, but they do not consider permanent faults, hence, we focus on permanent fault tolerance. We compare our proposed PFA-TP3M method with peak power management technique (TP3M) [1]. To compare PFA-TP3M with TP3M method, first we used our proposed P3FA-LTF algorithm which reduces the peak power consumption while tolerating permanent faults and then used PPA-LTF algorithm [1] for each task sets in simulation. Schedule length and permanent fault-tolerance rate are the parameters of our simulation to evaluate the proposed method. Higher permanent fault-tolerance and shorter timing length compared to TP3M algorithm [1] are favorable in our evaluation.

Figure 3 shows the permanent fault-tolerance rate where for a task graph with constant number of tasks (e.g., 10 tasks), the rate of permanent fault-tolerance increases as the number of cores increases. It can be observed from Figure 3 that with the increase in core numbers, it is more the probability of mapping different redundant copies of each task on separate cores.

Figure 4 shows the scheduling length based our proposed P3FA-LTF policy compared to the PPA-LTF policy where for a task graph with constant number of tasks (e.g., 10 tasks), scheduling length decreases with the increase of the core numbers. Since by increasing the number of processing cores, our proposed method can more easily map different versions of tasks to different cores during task scheduling.

Since TP3M algorithm which is proposed in [9], does not map separate versions of each task on separate processing cores, it will fail in the case of permanent fault occurs. However, our proposed PFA-TP3M algorithm due to the mapping of separate versions of each task on to separate processing cores, makes the system permanent fault-tolerance. Therefore, with the increase in the number of processing cores that corresponds to the advancement of multi-core processor technology, the proposed method in this thesis can show good performance.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 January 2024 doi:10.20944/preprints202401.0103.v1

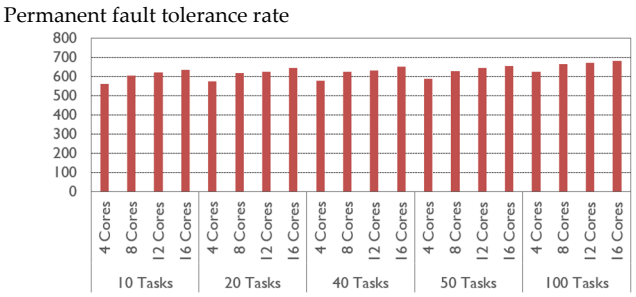


Figure 3. Permanent fault tolerance rate.

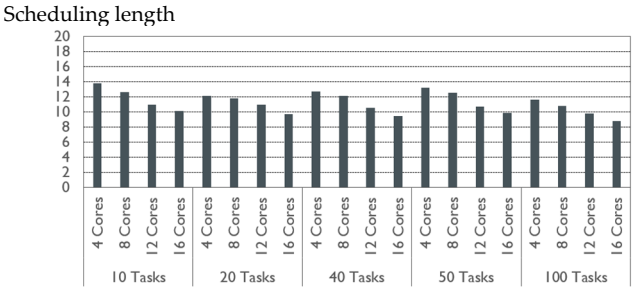


Figure 4. Scheduling length based our proposed P3FA-LTF policy compared to PPA-LTF policy.

Table 1. Characteristics of the benchmark applications.

		BITCOUNT	SUSAN	MATH	CRC32	SHA	QSORT	JPEG	FFT	DIJKSTRA	LAME	GSM
Execution time (ms)		193.15	118.09	1098.40	2078.51	39.36	206.82	47.89	960.88	89.90	3055.44	704.46
Energy consumption (mJ)		112.21	67.95	604.26	1107.95	22.51	120.18	29.44	554.07	56.59	1925.32	409.51
Min. Power (mW)	Dynamic	255.83	272.51	253.19	217.18	272.74	197.61	281.73	252.22	288.13	282.229	282.34
	Static	293.327										
	Total	**549.16**	**565.83**	**546.52**	**510.51**	**566.07**	**490.93**	**575.05**	**545.55**	**581.45**	**575.56**	**575.67**
Max. Power (mW)	Dynamic	576.54	562.61	473.69	431.94	515.79	479.84	536.81	494.01	431.47	458.52	437.3
	Static	293.327										
	Total	**869.87**	**855.94**	**767.01**	**725.27**	**809.12**	**773.17**	**830.14**	**787.33**	**724.80**	**751.85**	**730.63**

VI. Conclusions

Within this paper, we have introduced a method to handle persistent faults on embedded systems with multiple cores. This method utilizes TP3M algorithms to minimize the highest power usage. Additionally, we have created a novel scheduling algorithm called PFA-TP3M. This algorithm ensures that identical tasks are not assigned to the same cores, instead executing each duplicate on separate cores. Our suggested approach aims to avoid consecutive conflicts on cores that have experienced permanent faults in order to create a fault-tolerant system. By employing the TP3M algorithm, we eliminate overlaps in peak power consumption among concurrently running tasks, thereby ensuring that the maximum power usage remains within the limits set by the chip's thermal design power (TDP) constraint. The simulation outcomes demonstrate that our proposed scheme offers a significant increase in fault tolerance, with an average improvement of 630 times (up to 700 times) compared to the peak-power management algorithm. However, this improvement comes at the expense of a schedule length increase of task graphs, averaging around 11% (up to 14%) compared to the aforementioned algorithm.

References

1. Ansari, Mohsen, Sepideh Safari, Amir Yeganeh-Khaksar, Mohammad Salehi, and Alireza Ejlali. "Peak power management to meet thermal design power in fault-tolerant embedded systems." *IEEE Transactions on Parallel and Distributed Systems* 30, no. 1 (2018): 161-173.
2. Salehi, Mohammad, Alireza Ejlali, and Bashir M. Al-Hashimi. "Two-phase low-energy N-modular redundancy for hard real-time multi-core systems." *IEEE Transactions on Parallel and Distributed Systems* 27, no. 5 (2015): 1497-1510.
3. Mireshghallah, FatemehSadat, Mohammad Bakhshalipour, Mohammad Sadrosadati, and Hamid Sarbazi-Azad. "Energy-efficient permanent fault tolerance in hard real-time systems." *IEEE Transactions on Computers* 68, no. 10 (2019): 1539-1545.
4. Lee, Jinkyu, Buyoung Yun, and Kang G. Shin. "Reducing peak power consumption inmulti-core systems without violatingreal-time constraints." *IEEE Transactions on Parallel and Distributed Systems* 25, no. 4 (2013): 1024-1033.
5. Lee, BongKi, Jaehwan Kim, Yeuncheul Jeung, and Jongwha Chong. "Peak power reduction methodology for multi-core systems." In *2010 International SoC Design Conference*, pp. 233-235. IEEE, 2010.
6. Munawar, Waqaas, Heba Khdr, Santiago Pagani, Muhammad Shafique, Jian-Jia Chen, and Jörg Henkel. "Peak power management for scheduling real-time tasks on heterogeneous many-core systems." In *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 200-209. IEEE, 2014.
7. Pagani, Santiago, Jian-Jia Chen, and Jörg Henkel. "Energy and peak power efficiency analysis for the single voltage approximation (SVA) scheme." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, no. 9 (2015): 1415-1428.
8. Ansari, Mohsen, Mohammad Salehi, Sepideh Safari, Alireza Ejlali, and Muhammad Shafique. "Peak-Power-Aware Primary-Backup Technique for Efficient Fault-Tolerance in Multicore Embedded Systems." *IEEE Access* 8 (2020): 142843-142857.
9. Yeganeh-Khaksar, Amir, Mohsen Ansari, and Alireza Ejlali. "ReMap: Reliability Management of Peak-Power-Aware Real-Time Embedded Systems through Task Replication." *IEEE Transactions on Emerging Topics in Computing* (2020).
10. Ansari, Mohsen, Amir Yeganeh-Khaksar, Sepideh Safari, and Alireza Ejlali. "Peak-power-aware energy management for periodic real-time applications." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, no. 4 (2019): 779-788.
11. Ansari, Mohsen, Mostafa Pasandideh, Javad Saber-Latibari, and Alireza Ejlali. "Meeting thermal safe power in fault-tolerant heterogeneous embedded systems." *IEEE Embedded Systems Letters* 12, no. 1 (2019): 29-32.
12. Ansari, Mohsen, Javad Saber-Latibari, Mostafa Pasandideh, and Alireza Ejlali. "Simultaneous management of peak-power and reliability in heterogeneous multicore embedded systems." *IEEE Transactions on Parallel and Distributed Systems* 31, no. 3 (2019): 623-633
13. Buttazzo, Giorgio C. *Hard real-time computing systems: predictable scheduling algorithms and applications*. Vol. 24. Springer Science & Business Media, 2011.
14. Guthaus, Matthew R., Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, and Richard B. Brown. "MiBench: A free, commercially representative embedded benchmark suite." In *Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538)*, pp. 3-14. IEEE, 2001.