2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST), June 10-11, 2020, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.

A Comparative Study of Joint Power and Reliability Management Techniques in Multicore Embedded Systems

Sina Yari-Karin[*], Ali Sahraee[*], Javad Saber-Latibari[*], Mohsen Ansari[*‡], Nezam Rohbani[†], and Alireza Ejlali[*]
[*]Department of Computer Engineering, Sharif University of Technology
[*‡]The Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT)
[†]School of Computer Science, Institute for Research in Fundamental Sciences (IPM)
Email: {sinayari, asahraee, jsaber}@ce.sharif.edu, mohsen.ansari@kit.edu, rohbani@ipm.ir, and ejlali@sharif.edu

***Abstract*— Low power consumption and high-reliability are often major objectives in the design of embedded systems. To reduce power consumption, embedded systems usually employ system-level power management techniques, e.g. Dynamic Voltage Scaling (DVS) and Dynamic Power Management (DPM). To achieve high reliability, embedded systems often exploit fault-tolerant techniques. Fault-tolerant techniques are in a trade-off with energy consumption, peak-power consumption, and temperature. Thus, different methods have been introduced that simultaneously consider reliability and power consumption as the system constraints. Several novel methods have been proposed in previous work to reduce the power consumption of fault-tolerant systems, but there are no published guidelines to help designers to select the best approach for a given application. In this paper, we investigate the effectiveness and efficiency of these methods by evaluating them in an identical simulation environment for an accurate evaluation.**

Index Terms*— Embedded Systems, Fault Tolerance, Power Management.

I. Introduction

Embedded systems are the most widespread computers utilized in many industries such as the automotive industry, medical devices, smart city traffic control, and so on [1][2][3]. The main requirements of embedded systems are high reliability and low power consumption [4][5]. Embedded systems must tolerate both transient and permanent faults especially in the Internet of Things (IoT) devices [8][19][22]. Transient faults are mainly caused by electromagnetic interference and cosmic rays and display themselves as Single Event Upsets (SEUs) in memories [10]. The transient faults are usually resolved by re-execution of the tasks [7][11][42]. While, the permanent faults caused by the production time failures, aging failures, and environmental conditions which require redundancy to tolerate them [10]. However, fault-tolerant methods increase the power consumption of the system due to the use of redundancy [10]. For this reason, power/energy management methods are used to reduce power/energy consumption [12][13][14].

Apart from the reliability concerns, when the level of integration at the chip level increases we face problems like dark silicon, power density, and peak-power consumption, and energy consumption [4][9][15][16]. Power management techniques are used to overcome these problems [17][18]. However, power management techniques degrade system reliability in many cases [4][6][15][19]. The main problem is the reconciliation between fault-tolerance and power/energy management techniques because these thechniques have a negative effect over each other [15][20][21]. There are many methods available for reconciliation mentioned above.

Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) are common system-level methods used to reduce average and peak power consumptions [4][10][15]. The DVFS method reduces power consumption by scaling the voltage/frequency level. DPM also helps to reduce power consumption by shutting down unnecessary parts or when power constraint violation occurs. Thermal Design Power (TDP) and Thermal Safe Power (TSP) are the system-level power constraints in multicore embedded systems [4][23]. TDP is a chip-level constraint and it is equal to the maximum tolerable power that a chip can consume in the safe band [24]. Recently, TSP has been defined as a core-level power constraint and provides an efficient power constraint. It is necessary to keep the power consumption of the cores below TSP to prevent excessive temperature rise [25]. It should be noted that experimental results show that by exploiting power management techniques like DVFS the fault rate increases and to compensate for this increase in the fault rate fault-tolerance methods are mandatory [4][23]. Thus, methods that consider joint power and reliability management are interested.

In general, fault-tolerance methods can be divided into two categories: active redundancy methods and passive redundancy methods [26]. Active redundancy methods use fault detection to increase system reliability [26][29]. One of the most popular active redundancy methods is the standby-sparing method [14][27][28]. In the standby-sparing method, execution is switched to the spare unit when a fault occurs in the primary unit [26]. There are three types of options to assign tasks to primary and spare units, i.e., Hot Standby-sparing (HSS), Cold Standby-sparing (CSS), and Warm Standby-sparing (WSS) [26][29]. In the HSS method, the primary unit and the spare are both set up together, and any input to the primary unit is also given to the spare unit. If a fault occurs in the primary unit, the spare unit immediately begins to execute tasks [26][29]. But in the CSS method, the

978-1-7281-7551-5/20/$31.00 ©2020 IEEE

spare unit is off until a fault happens in the primary unit and after that spare unit starts to execute the tasks [26][29]. The switching speed between primary and spare units in the CSS method is longer than in the HSS method. The WSS method (also known as the de-energized state) is almost like the HSS method but it executes the initial part of the tasks and leaves the part that causes aging. On the other hand, passive methods use fault-masking techniques [26][29]. NMR and task replication are the most popular passive redundancy methods. NMR and task replication methods use redundant hardware to tolerate faults and software replicas to satisfy the reliability target, respectively. In the task replication method replicas, tasks can be executed simultaneously with the main tasks [26][29]. Also, there are hybrid systems that use both active and passive methods [29]. Fig. 1 shows the standby-sparing, NMR, and task replication methods.

In this paper, the power/energy consumption and reliability of different fault-tolerance methods are evaluated. For this aim, two popular method standby-sparing and task replication are chosen. In Section II, the related work will be reviewed. Section III deals with the definition of the problem and the assumptions. Section IV shows the results of the experimental evaluation. And in the end, Section V will present the conclusions of the paper.

II. RELATED WORK

As previously mentioned, we will evaluate two prominent fault-tolerant methods. *i*) Task Replication and *ii*) Standby-Sparing. The previous works related to these two categories are discussed here. Ansari et al. [4] proposed a peak power management scheme called TP3M. They have shown that although the use of the N-Modular Redundancy (NMR) technique can increase the system reliability, it can violate TDP constraint. The TP3M method prevents concurrent execution of tasks based on the power profile of the applications and reduces the overlap of the execution of the main tasks and the replica tasks. The proposed method by Salehi et al. in [30] is a power-efficient reliability management method that uses Dynamic Redundancy and Voltage Scaling (DRVS). This method achieved a significant improvement in reliability by applying various redundancy methods like triple and binary redundancy. The PPARM method presented in [15] is a peak-power-aware scheme that considers power as a critical resource. The increased power causes an increase in temperature which results in fault rate exacerbating. This paper exploits task replication and code version programming to improve the system reliability while at the same time meet the TSP constraint. In [21], the authors have proposed a reliability improvement method that defines a reliability target for the system. It has shown that task replication can improve the reliability of the system but, it has a negative impact on energy consumption. Because of this, the proposed method determines the minimum number of replicas for the reliability target and reduces energy consumption by allocating the appropriate frequency to the replicas. Poursafaei et al. [31] proposed a method that has two online and offline phases. In this method, a reliability target is defined and task replication is responsible for achieving this target. This method finds the number of replicas and frequency of cores to minimize energy consumption. Also, the proposed method benefits from the cancelation of replicas when a task finishes its jobs successfully. The proposed method in [10] mentioned that the fault coverage rate and DVS have a negative impact on reliability. Thus, the proposed method considers the core frequency, the number of replicas, fault coverage, and energy consumption induced by the DVS method simultaneously to minimize energy consumption. Static and dynamic solutions are proposed in the paper [10]. These methods are responsible for minimizing energy consumption by reducing the concurrent execution of replicas. Salehi et al. [32] have considered the overhead of the N-Modular redundancy method and have divided the execution of tasks into two necessary and demand-based phases. In the necessary phase, only half plus one of the replicas is executed and if the result of these replicas is as same as, so the fault did not detect and there is no need to execute the rest of replicas. When a fault is detected, the rest of the replicas will be executed to achieve the correct result.

In the study [14], an energy-aware standby-sparing system for heterogeneous multicore systems is proposed to tolerate both transient and permanent faults. Also, in this work, the primary core exploits DVFS and the spare core exploits only DPM to reduce energy consumption. In [14], Roy et al. have shown that the selection of LP (Low Power) or HP (High-Performance) core and the conscious allocation of frequency on the primary core has a significant impact on the power/energy consumption. Haque et al. [33] have proposed a standby-sparing method for fixed-priority tasks in hard real-time systems. In the proposed method, two queues are provided for the execution of the primary and backup tasks to delay the execution of the backup tasks as far as possible and reduce energy consumption through DPM and DVS methods. The proposed scheme in [27] has presented an energy-efficient and reliable scheduling method for a heterogeneous dual-core system. The proposed method can minimize power consumption by automatically canceling backup tasks and taking the voltage/frequency of each core to a minimum level. Ejlali et al. [7] have asserted that time redundancy methods are preferred than hardware methods. The proposed method uses a standby-sparing method and benefits from dynamic slacks times to minimize energy consumption, and also the primary and spare units exploit DVS and DPM techniques, respectively. Khavari-Tavana et al. [34] have proposed a standby-sparing system that consists of a feedback system in the primary unit and balances the workload to manage energy consumption, and meet the deadline constraint. The mentioned feedback system uses slack times to reduce total energy consumption. Gou et al. [35]

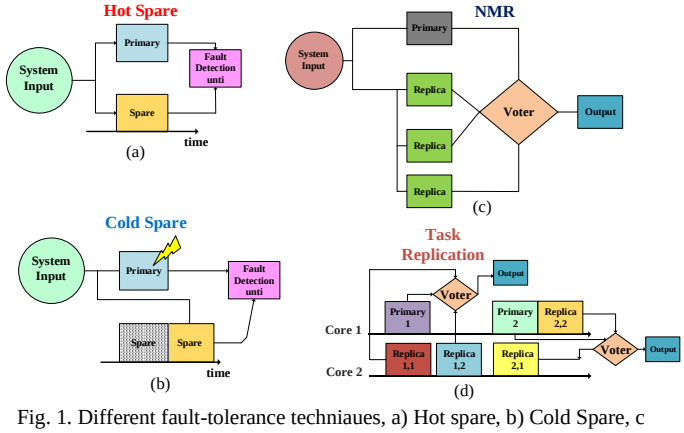


Fig. 1. Different fault-tolerance techniaues, a) Hot spare, b) Cold Spare, c TMR, and d) Task replication methods.

have proposed a standby-sparing method that tolerates one permanent fault and guarantees the reliability of the system for transient fault. In the proposed method two schemes have been discussed, i.e. Paired-SS and Generalized-SS. These methods improve the utilization of the system by exploiting the combinational use of primary and backup tasks. Also, the proposed method uses Preference-Oriented Earliest Deadline (POED) as an (Earliest Deadline First) EDF-based scheduler to minimize energy consumption. The authors in [36] state that there is a trade-off between reliability and power/energy consumption. The proposed method considers this contradiction and presents a reliability-aware scheduling algorithm to minimize energy consumption. The proposed algorithm uses the concept of priority scheduling and considers two queues. One of these queues is responsible for holding the low priority tasks based on worst-case execution time according to the Rate Monotonic scheduling algorithm and the second queue is responsible for the execution of high priority tasks. To achieve a higher performance in the standby-sparing method, the proposed method uses energy-speed based reliability to meet the reliability target of the standby sparing system and reduce energy consumption.

As discussed, there are lots of works that use task replication or standby-sparing methods to reduce power/energy concerning the reliability target in the system. In this paper, we have a glance at these methods and our evaluations will compare these methods to get an estimation of the effect of exploiting these methods. We want to evaluate the differences between the primary core type and frequency selection in the energy/power consumption of standby-sparing and the impact of the number of replicas and frequency value in the task replication method.

III. System Model

A) System Model

Homogeneous and heterogeneous system models are utilized to evaluate the proposed methods for simultaneous management of power/energy consumption and reliability to compare the existing methods. We consider a homogeneous system model like [5] and a heterogeneous model like [14][15]. The homogeneous system allows the cores to be mapped easily because of the uniformity of the cores [4][28]. The heterogeneous systems face more challenges because they have at least two types of low-power and high-performance cores [15]. For this purpose, the concept of the island is used in heterogeneous systems, and the cores within each island can be positioned in different modes. Due to constraints such as power/energy consumption and performance, heterogeneous systems are usually designed as islands with different cores, and the number and type of cores of each island are different from those of other islands [25]. Fig. 2 shows a simple heterogeneous system that consists of two low-power and high-performance islands [25][14].

B) Task Model

The task models used in this work are periodic task and frame-based models like [21] and [14], respectively. The difference between these two models is in the task sets deadlines. The task set $\Gamma = \{T_1, T_2, \dots, T_n\}$ consists of n tasks where each task T_i has a worst-case execution time $WCET_i$, time period π_i, and start time t_i. In the periodic task model, D_i represents the deadline of the task i and D is the shared deadline for all tasks in the frame-based task model. Like most studies on real-time scheduling, we also focus on a set of tasks that are all independent of each other. This task model is not as restrictive as it seems because there are some methods to transfer a set of dependent tasks to independent tasks [37].

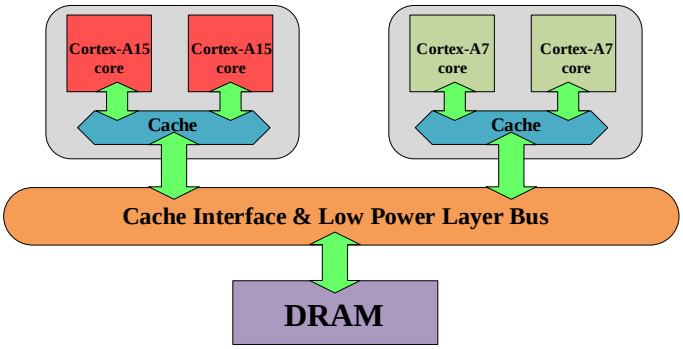


Fig. 2. A simple heterogeneous multicore system with low-power and high performance islands [14][25].

C) Power Model

The power consumption of the system consists of two parts: *i*) dynamic power and *ii*) static power. The power consumption of each core with the operating frequency f_i and the voltage V_i is given by Eq.1 [5][10][23]:

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

Where α is the average number of switches from zero to one and one to zero of the internal signals of the circuit during the clock period. C_L is equal to the total parasitic capacitance of the internal nodes of the circuit.

D) Fault and Reliability Model

The intended system can tolerate both transient and permanent faults. The fault model is also considered as previous studies [27][28][38]. The average rate of system faults depends on the frequency of the core and is obtained by Eq. 2.

$$\lambda(f) = \lambda_0 10^{\frac{d(1-f)}{1-f_{\min}}} \quad (2)$$

Where $\lambda_0 = 10^{-6}$ is the fault rate at the maximum frequency and d is the system sensitivity to operating voltage changes. The reliability of task i according to studies [5][10] is consistent with Eq. 3, where t_i is the execution time.

$$R_i(t_i) = e^{\lambda(f)t_i} \quad (3)$$

In this paper, we evaluate two forms of fault-tolerance technique. *i*) standby-sparing and *ii*) task replication. Unlike NMR, these methods can tolerate faults with fewer spare units, so we have considered these techniques. Reliability in the standby-sparing method depends on the method chosen for fault tolerance. For example, the reliability of the CSS technique with $0 \leq$ fault coverage ≤ 1 and the reliability of primary and backup tasks are given by Eq. 4 [10].

$$R_i(t) = (1 + C\lambda t) e^{-\lambda t} \quad (4)$$

If primary and backup tasks are independent, so the reliability is computed by Eq. 5.

$$R_i(t) = R_p + (1 - R_p) R_s \quad (5)$$

Where R_p and R_s are the reliability of primary and backup tasks, respectively. And if $R_p = R_s$, the reliability of a task can be written as Eq. 6.

$$R_i(t) = 2R + R^2 \quad (6)$$

Therefore, the reliability of the system with the *N* task can be calculated by Eq. 7.

$$R_{system}(t) = \prod_{N} R_i(t) \quad (7)$$

On the other hand, the reliability of the task replication technique with *k* tasks can be calculated by Eq. 8.

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

Where in Eq. 8 *k* is the number of replicas of the task executed on *k* separated cores. Also, system reliability can be written as Eq. 9.

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

IV. Algorithm Discussion

In this paper, two of the most popular algorithms applied to the systems have been considered to evaluate the energy/power consumption of fault-tolerance methods. *i*) The system exploits a spare unit to meet the reliability target [10][14][27][33]. The standby-sparing method is used for this purpose. Given the different policies outlined in the previous sections for the implementation of the primary and spare tasks using the SS method, the next issue is how to select the primary and spare cores. Heterogeneous systems can also have different power/energy outputs by executing the tasks on different islands. On the other hand, choosing the proper frequency also has a great impact on power/energy consumption and reliability. For this reason, the methods proposed using the SS technique consider the type of core and frequency. To evaluate this method, we use the algorithms presented so far to select the type and frequency of the cores in the primary and spare units [14]. *ii*) The algorithm inserts one or more replicas for the system to satisfy its reliability target [15][28]. Sometimes it is preferable to maintain reliability with a stronger mechanism such as task replication. For the reliability target to be inconsistent with the power/energy consumed, a policy must be chosen to achieve the reliability target with the least number of redundancies and reduce the energy/power consumption. For this reason, the number of replicas and the operating frequency of the cores are a challenge for this fault-tolerance method [21].

Roy et al. [14] have divided the type of primary cores and frequency value selection into six categories. This study states that the cautious choice of the type of primary core and overlap between task execution can help to reduce energy consumption. To this end, the methods used in the mentioned paper have been evaluated. These methods can answer two basic questions.

1. Which type of cores should be selected as the primary core? Low power or high-performance?
2. How to consider the frequency of the primary cores to reduce energy consumption?

There are six possible approaches to answer the mentioned questions [14]:

- **Faster/Slower-Static**: The frequency is considered static and the High-performance/Low power cores are considered as the primary cores. In this scheme, the frequency of task i is $f_i = \max(f_i^{ee}, f_u)$. Where f_i^{ee} is the energy-efficient frequency and is defined as the minimum frequency that energy consumption is reduced by DVFS [20]. Also, f_u is related to the tasks' execution time and the deadline and can define as $f_u = \frac{\sum exe\_time_i}{D}$.
- **Faster/Slower-Minimize Overlap (MO)**: This scheme adjusts the frequency of the primary cores so that the primary and backup tasks are executed with the least possible overlap. For this purpose, the frequency of the primary cores is selected such that the execution of the primary task ends before the start of the backup task.
- **Faster/Slower-Overlap-aware (OA)**: Unlike the MO scheme, this scheme states that it is possible to reduce energy consumption by overlapping the execution of the primary and backup tasks. This overlap is obtained by solving the optimization problem in Eq. 10 and Eq. 11:

$$Minimize: E_{f_i}^{primary} + E_{f_i}^{spare} \quad (10)$$

$$Subject\ to: f_U < f_i < Min(f_i^{MO}, f_{Max}) \quad (11)$$

The authors in [21] have proposed a task replication method. In this paper, the energy consumption is minimized by considering a reliability target and the frequency scaling effect. The optimization problem can be expressed as Eq. 12:

$$Minimize: \sum_{core=1}^{n} E_{core}(f_i) \quad (12)$$

Eq. 13 states that it minimizes the energy consumption and the number of replicas, and the execution of replica *k* and *j* from task i should not be on the same core and simultaneous.

$$\begin{gathered} Subject\ to: \\ Mapping\ Task_{i,j}(f_i) \neq Mapping\ Task_{i,k}(f_i) \\ Minimize\ \ \#replicas: \\ Reliability_{system} \geq Reliability_{target} \end{gathered} \quad (13)$$

Various heuristic schemes can be considered to select the tasks for mapping [21]:

- **Largest Energy First (LEF)**: Choose a task that has the largest energy-saving at first.
- **Largest Power First (LPF)**: Choose a task that has the largest power-saving at first.
- **Largest Utilization First (LUF)**: Choose a task that has the largest utilization at first.

The concept of Energy-Frequency-Reliability table (EFR) in [21] is used to implement the above heuristics. This table helps to find the proper number of replicas and frequencies to maintain reliability and reduce power/energy consumption. It has also been pointed out that allocating uniform frequency to replicas is not an optimal method. It may degrade the reliability by decreasing the frequency of one replica, but overall, by choosing a higher frequency for the other replicas, it can be achieved lower power/energy consumption and keeps the reliability target. In the rest of the paper, we will evaluate these methods.

V. Experimental Results and Evaluation

In our evaluation, gem5 [39] and McPAT [40] simulators were used to evaluate the mentioned proposed methods. The gem5 simulator is responsible for defining the system model. As shown in the Table. 1, we considered the ARM-based

Table 1. System Configuration details

System Architecture	ARM Cortex-A7 and ARM Cortex-A15 V/F: [0.85 V to 2 V] / [1.5 GHz to 3.5 GHz]		
Memory	Main	4GB, 8 bank per rank, Access time = 100 cycle DRAM	
	Cache	L1	32KB 4-way SRAM
		L2	1MB 16-way STT-RAM

platform and used ARM-Cortex A7 and ARM-Cortex A15 as widely used cores in embedded systems to evaluate the proposed methods. We suppose that our system has equipped with DVFS and DPM techniques. When it is needed, we can use these techniques to manage power/energy consumption. McPAT has also been used to obtain the information needed such as power consumption and execution time on the target system. The task set is also selected from the MiBench benchmark, designed for embedded system applications [41]. Table. 2 shows the results of simulating the execution of tasks on the target system. The Python programming language is also used to implement the proposed algorithms.

In our evaluation, we make our task set by generating 200 random tasks to evaluate the mentioned methods. To evaluate the proposed standby-sparing methods, the heterogeneous system model is considered to be an ARM big.LITTLE system [14][25]. Also, we considered a homogeneous multicore system to evaluate the task replication method. We obtained energy and power consumption for different utilizations in the mentioned systems to evaluate the proposed methods and compare the differences between these two methods. Also, the reliability target value is calculated through Eq. 1 and Eq. 2 for standby-sparing and task replication techniques at different frequencies, respectively. At any given moment, the system reliability at any frequency should not be less than the reliability target. In the following, we will examine each of the proposed methods.

Standby-Sparing: To evaluate the standby-sparing algorithms, we implemented Static, MO and OA algorithms in fast-primary (FP) and slow-primary (SP) modes and executed the mentioned task set on a heterogeneous system. We considered different utilization value between utilization=0.2 to utilization=1. It is well to mention that according to the optimization issues and frequency assignment scheme, the SP-OA and SP-MO are almost in the

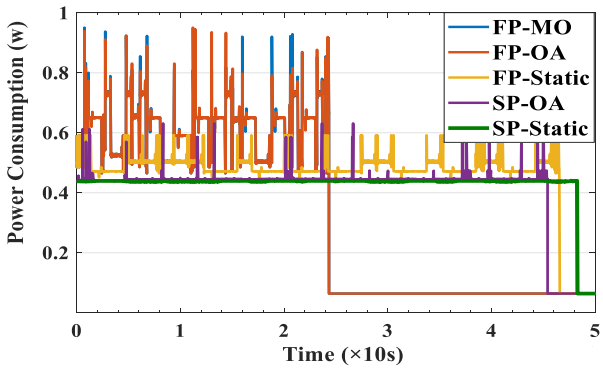


Fig. 3. The power profile of standby-sparing methods.

Table 2. Simulation Results

Task	Execution Time (ms)		Average Power (W)		Energy (mJ)	
	A7	A15	A7	A15	A7	A15
basicmath	242.37	86.188	0.412	0.690	99.9515	59.54
bitcount	35.314	11.56	0.423	0.708	14.96	8.19
crc32	2635.87	1273.69	0.413	0.618	1090.93	787.67
dijkstra	43.74	16.36	0.414	0.677	18.15	11.08
fft	259.01	88.88	0.412	0.695	106.86	61.80
jpeg	39.24	12.99	0.422	0.749	16.59	9.73
lame	1514.84	540.18	0.411	0.698	623.75	377.53
patricia	145.90	43.54	0.414	0.634	60.51	27.615
qsort	33.15	9.73	0.413	0.640	13.69	6.23
susan	26.26	8.62	0.414	0.720	10.87	6.21

same behavior, and we just show the SP-OA in our evaluations. Fig. 3 shows the power profile of six methods under per-chip utilization=0.8. The mentioned algorithms have been evaluated with different per-chip utilizations. The experimental results show that the execution time of Static methods is more than MO and OA methods and this is one of the reasons for increased energy consumption and reduced performance. As shown in Fig. 3 and Fig. 4, methods that consider dynamic frequency for standby-sparing techniques and perform overlapping between primary and backup tasks can reduce energy consumption relative to the static selection method by an average of 31.44% and up to 40.63%. Also, as shown in Fig. 5 workload has a significant impact on deciding which method should be selected. For example, in utilization=0.2, it is better to minimize the overlap by the OA method instead of using MO to schedule the tasks that have some overlaps between primary and backup execution. On the other hand, Fig. 5 shows when the utilization-level increases, choosing the core type is more effective. For example, in utilization=1, when we use the FP-MO scheme, we can achieve 19.95% saving in average power consumption. This is while, in utilization=0.6, this saving is 13.3%. Therefore, we can assert that when a standby-sparing method is applied, we should consider workload, primary core type, and primary core frequency. According to our constraints and system specification, to achieve a low power/energy system, we can add some overlaps between the execution of tasks and thereby reach our goal.

Task Replication: The homogeneous system is considered to evaluate the task replication method. This is because we want to run equally the primary and replicas, and there is no need to synchronize tasks and replicas. We

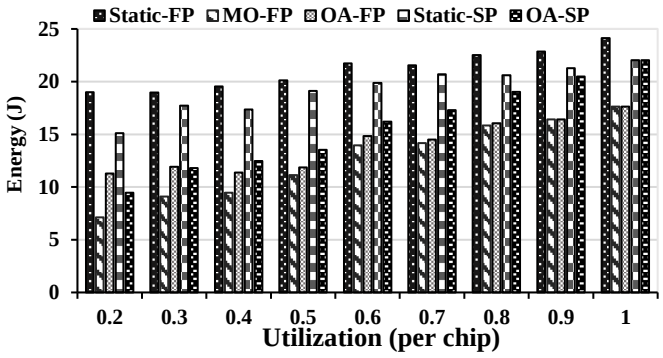


Fig. 4. Impact of frequency, primary core type, and utilization in energy of standby-sparing methods.

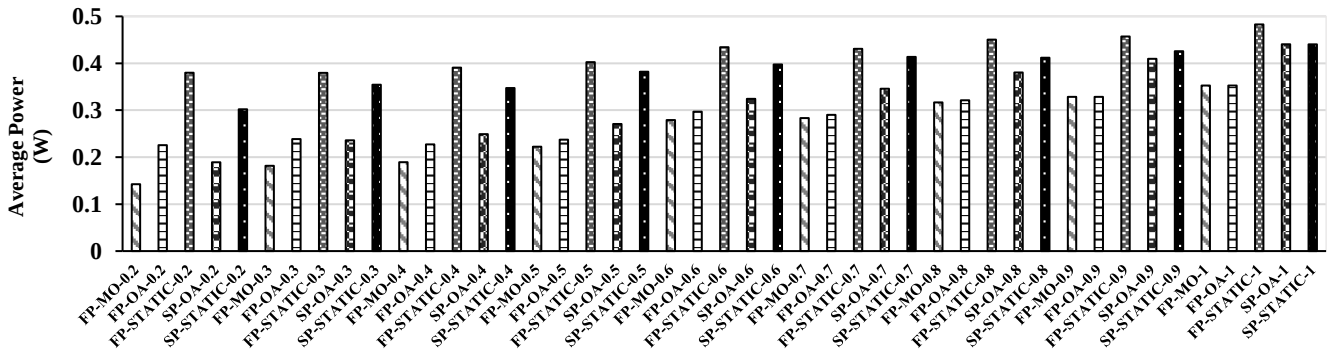


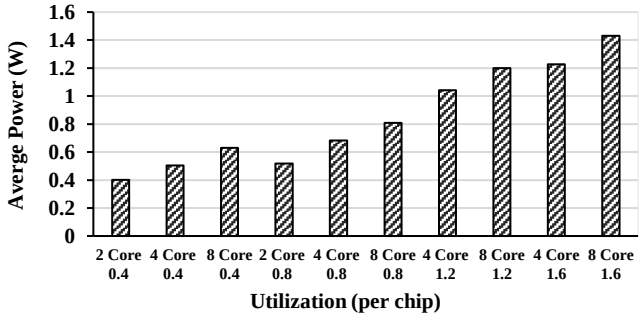
Fig. 5. Average power of standby-sparing method with different policies and different utilizations.

consider the utilization of the cores based on the maximum number of the running task without any adding replicas. We evaluate the impact of frequency and the minimum number of replicas on energy consumption by increasing utilization from utilization=0.2 to utilization=1.6 and by increasing the number of cores from 2 to 8 under a given reliability target. We assume three 2-Core, 4-Core, and 8-Core systems. On the other hand, we should note that to be able to show the effect of increasing the number of replicas. The amount of utilization mentioned here is equal to the sum of the productivity of all cores, which can have a value greater than one.

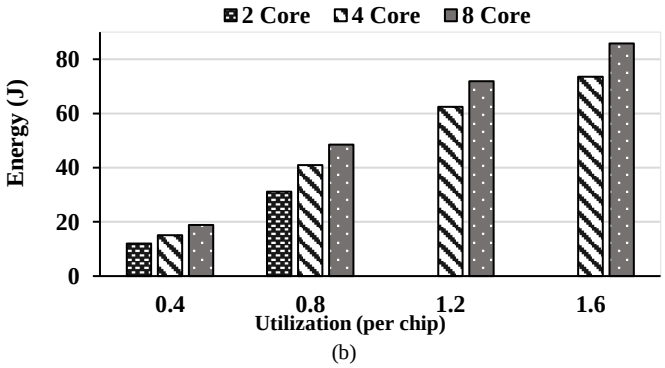
When we increase the utilization of a dual-core system up to 0.8 and for a system with four cores up to 1.6, based on our simulations, we cannot add any replicas to the system. Thus, we consider the amount of utilization that can be added to the system. It was noted earlier that we have three methods for selecting tasks and frequency of cores: LEF, LPF, and LUF [21]. In our simulations, we choose the best scheme to achieve low power/energy consumption in Fig. 6. Fig. 6a shows the average increase in the power consumption is 25% (up to 36.40%) as the core number and utilization increase. Fig. 6b shows the average increase in energy consumption of task replication is 22.48% (up to 36.39%) based on the number of cores and utilization. According to the simulation results, we can assert that in the task replication technique we are looking for the minimum number of replicas to minimize energy/power consumption. For this purpose, the reliability target specifies the replication level. Also, in a specific utilization, we can minimize the number of replicas to get more performance and low power/energy consumption by deciding the best policy of task selection according to energy, power, and utilization.

Standby Sparing vs. Task Replication: To have a fair comparison between standby-sparing and task replication methods, we suppose a dual-core system and we consider the same utilization for all methods. The system is considered to be a dual-core system because we consider the SS methods as a method for heterogeneous systems that require at least two cores to utilize the low-power and high-performance characteristics simultaneously. For this reason, we should also consider the dual-core for the task replication system so that we can run tasks and replicas at the same time. We evaluate the methods under two different utilization-level, i.e. utilization=0.4 and utilization=0.8. In Fig. 7, the effect of heterogeneity on both power consumption and execution time can be seen. Fig. 7a shows that generally the standby sparing methods, especially those that consider the overlap between primary and backup tasks (MO and OA), have a better execution time than task replication. It has also been shown that using SS methods can reduce average power by 29.77% on average (up to 52.77%) compared to task replication methods. Fig. 7b shows increasing the number of replicas in some cases can act similarly or better than standby-sparing method especially when the frequency scaling is not allowed. Our estimates show that the standby-sparing method can perform energy saving on average 13.31% and up to 49.09% better than the task replication method.

In conclusion, standby-spring and task replication are traditional methods that consider both reliability and power/energy consumption simultaneously and can design a low power (energy efficient) fault-tolerant system. The frequency and type of core (LP or HP) used today have a great impact on power/energy and reliability. Therefore, when using the methods mentioned above the challenge of the type of core and its frequency must be taken into account. The

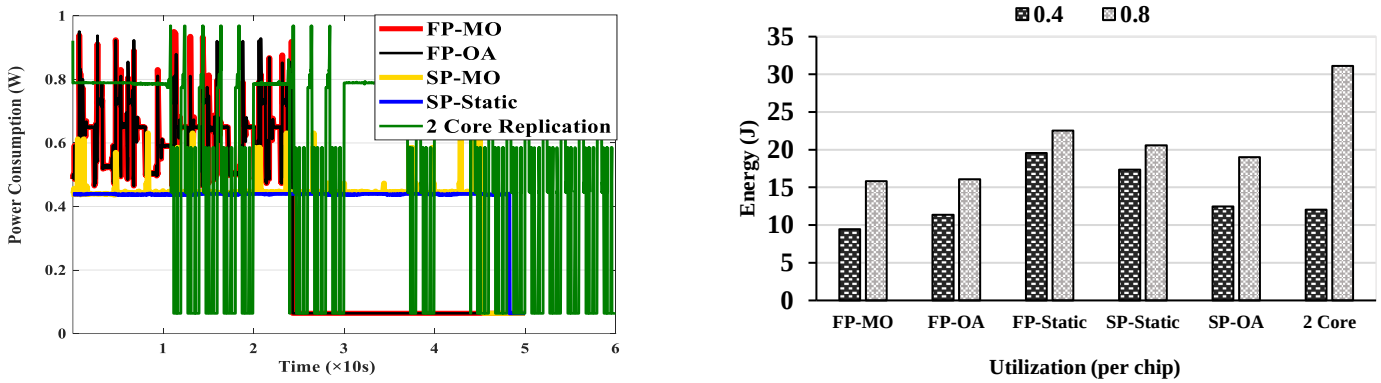


(a)



(b)

Fig. 6. a) Average power, b) Energy, of task replication method with different utilization-levels.



(a) (b)

Fig. 7. Comparison between spare and task replication. a) Power Consumption. b) Energy Consumption.

results of the simulations have shown that in general to discuss the power/energy consumption the choice of a dynamic frequency has better effects than the static frequency. The task replication method can achieve higher reliability than the reliability target, facing increased power/energy consumption. So if the power/energy challenge is more important, the standby-sparing technique can work better.

VI. Conclusions

Low power consumption and high-reliability are two main objectives in designing hard real-time embedded systems. Most of the embedded systems which are utilized in safety-critical applications employ fault-tolerant techniques to achieve the required reliability level. However, fault-tolerant techniques incur considerable power overhead. Several novel methods have been proposed in previous work to reduce the power consumption of fault-tolerant systems, but there are no published guidelines to help designers to select the best approach for a given application. In this paper, we compare different fault-tolerant management methods used for reducing the power consumption of embedded systems.

References

[1] E.A. Lee and S.A. Seshia, "Introduction to Embedded Systems, A Cyber-Physical Systems Approach", *Second Edition, MIT Press*, ISBN 978-0-262-53381-2, 2017.

[2] G. Buttazzo, "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications" in *Springer US*, vol. 24, pp. XVI-524, 2011.

[3] P. Marwedel, "Embedded system design", in *Springer*, vol. 1, 2006.

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

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

[6] Z. Shirmohammadi, M. Ansari, S. K. Abharian, S. Safari, and S. G. Miremadi, "PAM: A Packet Manipulation Mechanism for Mitigating Crosstalk Faults in NoCs," in *IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Liverpool, UK, pp. 1895-1902, 2015.

[7] A. Ejlali, B.M. Al-Hashimi, and P. Eles, "A Standby-Sparing Technique with Low Energy-Overhead for Fault-Tolerant Hard Real-Time Systems," in *Proc. International Conference on Hard-ware-Software Codesign and System Synthesis (CODES+ISSS 2009)*, Grenoble, France, pp. 193-202, October 2009.

[8] B. Safaei, A. M. H. Monazzah, T. Shahroodi, and A. Ejlali, "Objective function: A key contributor in Internet of Things primitive properties," in *Proceedings of the Real-Time and Embedded Systems and Technologies (RTEST)*, Tehran, Iran, pp. 39-46, 2018.

[9] R. Narimani, B. Safaei, A. Ejlali, "A comprehensive analysis on the resilience of adiabatic logic families against transient faults," in *Integration*, 2020.

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

[11] S. Safari, S. Hessabi, and G. Ershadi, "LESS-MICS: A Low Energy Standby-Sparing Scheme for Mixed-Criticality Systems," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020. (Early access)

[12] S. Safari, M. Ansari, G. Ershadi and S. Hessabi, "On the Scheduling of Energy-Aware Fault-Tolerant Mixed-Criticality Multicore Systems with Service Guarantee Exploration" in *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2338-2354. 2019.

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

[14] A. Roy, H. Aydin, and D. Zhu, "Energy-aware standby-sparing on heterogeneous multicore systems," *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Austin, TX, pp. 1-6, 2017.

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

[16] S. Safari, M. Ansari, M. Salehi, and A. Ejlali, "Energy-Budget-Aware Reliability Management in Multi-Core

Embedded Systems with Hybrid Energy Source," *The CSI Journal on Computer Science and Engineering (JCSE)*, vol. 15, no. 2, pp. 31-43, 2018.

[17] B. Safaei, A. A. M. Salehi, M. Shirbeigi, A. M. H. Monazzah, and A. Ejlali, "PEDAL: power-delay product objective function for internet of things applications," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC)*, pp. 892-895, ACM, 2019.

[18] M. Khatir, A. Moradi, A. Ejlali, M. T. Manzuri Shalmani, and M. Salmasizadeh, "A secure and low-energy logic style using charge recovery approach," *Proceeding of the 13th international symposium on Low power electronics and design (ISLPED '08)*, Bangalore, 2008, pp. 259-264.

[19] B. Safaei, A. M. H. Monazzah, M. B. Bafroei, and A. Ejlali, "Reliability side-effects in internet of things application layer protocols," in *Proceedings of the 2nd IEEE International Conference on System Reliability and Safety (ICSRS)*, pp. 207-212, IEEE, 2017.

[20] D. Zhu, R. Melhem and D. Mosse, "The effects of energy management on reliability in real-time embedded systems," in *IEEE/ACM International Conference on Computer-Aided Design* (ICCAD), CA, USA, pp. 35-40, 2004.

[21] M. A. Haque, H. Aydin, and D. Zhu, "Energy-aware task replication to manage reliability for periodic real-time applications on multicore platforms," 2013 *International Green Computing Conference Proceedings*, Arlington, VA, pp. 1-11, 2013.

[22] B. Safaei, A. A. M. Salehi, A. M. H. Monazzah, and A. Ejlali, "Effects of RPL Objective Functions on the Primitive Characteristics of Mobile and Static IoT Infrastructures," in *Microprocessors and Microsystems*, vol. 69, pp. 79-91, Elsevier, 2019.

[23] M. Ansari, M. Pasandideh, J. Saber-Latibari, and A. Ejlali, "Meeting Thermal Safe Power in Fault-Tolerant Heterogeneous Embedded Systems," in *IEEE Embedded Systems Letters*, vol. 12, no. 1, 2020.

[24] Intel Corporation, "Dual-core intel Xeon processor 5100 series datasheet, revision 003," August 2007.

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

[26] E. Dubrova, "Fault-Tolerant Design" in *Springer*, 2013.

[27] A. Roy, H. Aydin and D. Zhu, "Energy-efficient primary/backup scheduling techniques for heterogeneous multi-core systems," *Eighth International Green and Sustainable Computing Conference (IGSC)*, Orlando, FL, pp. 1-8, 2017.

[28] M. A. Haque, H. Aydin, and D. Zhu, "Energy-Aware Standby- Sparing Technique for Periodic Real-Time Applications," in *Proceedings of the IEEE 29th International Conference on Computer Design (ICCD'11)*, pp. 190-197, Oct. 2011.

[29] K. S. Trivedi, "Probability and Statics with Reliability, Queuing, and Computer Science Application" in *John Wiley and Sons Ltd*, 2016.

[30] M. Salehi, M. Khavari Tavana, S. Rehman, F. Kriebel, M. Shafique, A. Ejlali, and J. Henkel, "DRVS: Power-efficient reliability management through Dynamic Redundancy and Voltage Scaling under variations," 2015 *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Rome, 2015, pp. 225-230.

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

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

[33] M. A. Haque, H. Aydin, and D. Zhu, "Energy management of standby sparing systems for fixed-priority real-time work-loads," *2013 International Green Computing Conference Proceedings*, Arlington, VA, pp. 1-10, 2013.

[34] M. K. Tavana, M. Salehi and A. Ejlali, "Feedback-Based Energy Management in a Standby-Sparing Scheme for Hard Real-Time Systems," 2011 *IEEE 32nd Real-Time Systems Symposium*, Vienna, pp. 349-356, 2011.

[35] Y. Guo, D. Zhu, H. Aydin, J.J. Han, L.T. Yan, "Exploiting primary/backup mechanism for energy efficiency in dependable real-time systems," in *Journal of Systems Architecture*, vol 78, pp. 349-356, 2017.

[36] V. Moghaddas, M. Fazeli, and A. Patooghy, "Reliability-oriented scheduling for static-priority real-time tasks in standby-sparing systems," in *Microprocessors and Microsystems*, vol 45, pp. 208-215, 2016.

[37] S. Kodase, S. Wang, Z. Gu, and K.G. Shin, "Improving Scalability of Task Allocation and Scheduling in Large Distributed Real-Time Systems Using Shared Buffers," in *Proceedings of the IEEE 9th Real-Time Technology and Applications Symp. (RTAS)*, pp. 181-188, 2003.

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

[39] Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator, " in *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[40] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," 2009 42nd *Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, New York, pp. 469-480, 2009.

[41] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudg, and R. 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), Austin, TX, USA, pp. 3-14, 2001.

[42] A. Hoseinghorban, A. M. H. Hosseini Monazzah, M. Bazzaz, B. Safaei and A. Ejlali, "COACH: Consistency Aware Check-pointing for Nonvolatile Processor in Energy Harvesting Systems," in *IEEE Transactions on Emerging Topics in Computing*, 2019.