Energy Aware Fault Tolerant Fixed Priority Task Scheduling in Multiprocessor System

Kiran Arora[1,2]
[1]Research Scholar
IKGPTU, Jalandhar, India
[2]Department of CSE
BHSBIET, Lehragaga,
Sangrur, India
erkiranarora@gmail.com

Savina Bansal[3,4] and Rakesh Kumar Bansal[3,4]
[3]Department of ECE
GZSCCET, Bathinda, India
[4]Maharaja Ranjit Singh
Punjab Technical University,
Bathinda, India
savina.bansal@gmail.com, drrakeshkbansal@gmail.com

***Abstract*—Energy Management and Fault Tolerance are the two main design dimensions for Real Time Embedded Systems. In this paper, the combination of these two design parameters has been exploited. Various energy aware fault tolerant task scheduling techniques have been proposed for Standby-Sparing Systems. In these systems, one processor is used for executing primary tasks whereas other one executes only backup tasks. Energy is saved by applying DVS and DPM techniques or by shifting the backup task as late as possible to reduce the overlapped execution of Primary and Backup copy of a task. But using one processor for sole purpose of executing backups only, wastes the processor capacity as well as affects the energy consumption of the system. In this paper, it is conjectured that allocating mixture of primary and backup tasks on both processor will reduce the energy consumption of the system. The simulated results of the proposed Mix-Allocation policy for fixed-priority periodic real-time tasks in comparison to the recently proposed Standby/Sparing policy justifies our conjecture.**

***Keywords*— energy, fault, fixed-priority, task scheduling**

I. INTRODUCTION

Energy Management has always remained a hot topic in multiprocessor/multicore systems. The reason behind this concern is the increased heat dissipation due to miniaturization of electronic chips in present-day computing systems. Bulk of research work has been proposed in literature for Energy Management. Dynamic Voltage Scaling (DVS) and Dynamic Power Management (DPM) are the two techniques for minimizing energy consumption that have proven to be most effective ones. With DVS technique, reduction in energy consumption is achieved by switching the processor to operate on low voltage levels. DPM on the other-hand puts the processor to sleep state in idle intervals, thereby saving energy.

Another equally important research issue is Fault Tolerance. Computing systems are prone to faults that sometimes lead to runtime errors. Faults can be classified as Transient and Permanent faults. Fault tolerance is normally achieved with the help of redundancy e.g. Time Redundancy or Hardware Redundancy. Transient faults can be tolerated by using Time Redundancy, but Permanent fault can be handled only with Hardware Redundancy. As multicore processors have become a trend of the day, thus hardware redundancy is available easily for tolerating permanent faults.

Energy Management and Fault Tolerance have been extensively studied in the literature, but the co-management of both is also receiving the researcher's attention these days. The reason behind this attention is the increased rate of transient faults with the use of DVS for saving energy. When the processor runs at low voltage levels, the transient fault rate increases significantly [1]. Thus, it is vital to take measures to add fault tolerance in system while using DVS for energy management.

To achieve the objective of handling permanent fault and guaranteeing reliability in the system with respect to transient faults, Standby-Sparing scheme has been proposed by Haque et al. for periodic fixed priority real time tasks [2]. It is based on Primary/Backup approach implemented on dual-processor system. The primary task is executed on one processor using DVS for energy management whereas the backup copy is scheduled on secondary processor that uses only DPM for reducing power consumption. The overlap between the execution of primary and backup is reduced by scheduling the backup as late as possible. Further energy is saved by cancelling the backup as soon as the primary copy of task completes its execution.

The Standby-Sparing technique does not utilize the processor capacity effectively as most of the time backup processor remains in sleep state [2]. The primary processor on the other hand remains busy heavily at high utilizations. This affects the amount of energy consumed by the system. In this paper, it is proposed that if both the processors are used for scheduling primary and backup tasks in a mixed manner, then energy consumption can be reduced as well as the processor capacity can be utilized effectively.

The rest of paper is divided into following sections: Section 2 reviews the Literature Survey. Section 3 elaborates the models and assumptions. Section 4 explains the energy aware fault tolerant task scheduling. Section 5 presents the simulated results and section 6 concludes the paper.

II. LITERATURE SURVEY

The energy aware fault tolerant techniques can be classified based on type of redundancy used i.e. Time Redundancy and Hardware Redundancy. Time redundancy techniques use the available slack for saving energy and executing recovery copies of tasks as well. Hardware Redundancy use the slack for slowdown of processor to save energy and extra hardware for providing fault tolerance. Recovery Task placement and Checkpointing are the most commonly explored techniques that refer to Time Redundancy. Task Replication and Standby-Sparing Techniques fall under the category of Hardware Redundancy.

Reliability aware power management framework (RA-PM) tries to preserve the original reliability of a system in the presence of DVFS. It works with an idea that a recovery copy of a task, which has been considered for voltage reduction, must be scheduled using the available slack before exploiting the slack for DVFS for saving energy [3]. RA-PM for multiple tasks has been demonstrated by Zhu et al. ,where few tasks from the task-set/application are selected for applying RA-PM[4]. Zhao et al. [5] claims that instead of placing recoveries statically for all the tasks of periodic task set, modest recovery allowance with dynamically allocated recoveries helps in achieving high reliability, as long as the allowance can be reclaimed when needed during hyper-period. Zhao et al. [6] has considered the reliability preservation for energy constrained systems for frame based task system (for precedence constraint tasks) as well as for periodic task system. The objective is to maximize reliability under the constraint that energy consumption must not exceed the given energy budget.

According to Zhao et al. [7][8], the RA-PM schemes discussed above are conservative, because allocating multiple recovery blocks decreases the prospects for energy saving by reducing the available slack for DVS. Thus, a new Shared Recovery approach for RA-PM has been introduced by Zhao et al., where in spite of separate recovery copies for scaled tasks, one global shared recovery block is reserved, which can be used by any task at any time in the situation of fault. Han et al.[9] also worked on Energy Efficient Fault Tolerance with Shared Recovery Block.

The combination of DVS and checkpointing for power aware fault tolerance has been studied by various authors [10][11]. A Non-Uniform Checkpointing technique has been presented by Melhem et al. which has an advantage of more energy saving over Uniform Checkpointing. K fault tolerance with Two-State Checkpointing (TsCp) has been presented by Salehi et al. [11]. Power management while tolerating fault has also been presented by Zhang et al. based on checkpointing and DVS combination [12]. He proposed Adaptive checkpointing where the checkpointing interval is not fixed, as it is adapted according to fault rate.

Haque at al. used replication to achieve the reliability while saving energy with DVS [13][14]. Author has presented the interplay of Energy, reliability, frequency and replication. Similar work has been done for dependent tasks [15]. Three level of redundancy has been considered by Salehi et al. i.e. Single Execution (SE), Dual Modular Redundancy (DMR) and Triple Modular Redundancy (TMR).

Energy has been saved with DVS and DPM policies in standby sparing system by Aminzadeh el al. [16]. Different from the conventional hot and cold standby sparing technique, Ejlali el al. has proposed a Low Energy Standby Sparing (LESS) system, where primary saves energy with DVS and spare processor saves energy by using DPM i.e. by keeping the spare processor idle as much as possible[17][18]. Guo et at. has proposed an energy efficient fault tolerance schemes where fault has been tolerated with Hardware Redundancy and energy is saved with DVFS [19]. He used Earliest Deadline first policy as underlying scheduling algorithm .

Standby-Sparing technique has also been exploited by Haque et al. for Fixed Priority real time periodic tasks. Dual queue mechanism has been used for delaying the backups as much as possible on secondary processor [2]. Energy is saved by executing the tasks at lower frequency on primary processor. The backup copy is cancelled as soon as the primary executes successfully.

III. MODELS AND ASSUMPTIONS

A. Task Model

Real time fixed-priority periodic tasks on dual processor system have been taken into consideration. A taskset τ contains tasks such that $\tau = \{\tau_1 \ldots \tau_n\}$. A task τ_i is commonly represented by three parameters: Worst Case Execution Time C_i under the maximum frequency, Relative Deadline D_i and Period P_i. Utilization U_i of a task τ_i is calculated as $\frac{C_i}{P_i}$. The total utilization of taskset i.e. sum of utilizations of all tasks is represented by U_{tot}. The Hyperperiod $Hyp(\tau)$ of a task-set is defined as Least Common Multiple (LCM) of task periods.

τ^B refers to the taskset that contains the backup copies τ_i^B of task τ_i from τ with same timing parameters as τ_i .

B. Power Model

The power consumption of embedded systems can be categorized as dynamic power and static power. The dynamic power (ρ_{dyn}) consumption arises due to charging and discharging of the load capacitance [20][21] [2]. Thus,

$$\rho_{active} = \rho_{dyn} + \rho_{static} + \rho_{ind} \tag{1}$$

Specifically, the power consumption ρ is a function of supply voltage (υ) and clock frequency ($\mathcal{F}$) [20]:

$$\rho_{active} = \varrho C_{ef} \upsilon^2 \mathcal{F} + \rho_{static} + \rho_{ind} \tag{2}$$

where C_{ef} is the total capacitance, ϱ is the gate activity factor. The supply voltage (υ) has a linear relationship with frequency $\mathcal{F}$. Thus, Eq. (2) can be written as,

$$\rho_{active} = \varrho C_{ef} \mathcal{F}^3 + \rho_{static} + \rho_{ind} \tag{3}$$

The maximum CPU frequency $\mathcal{F}_{max}$ has been set to 1. All other values are normalized with respect to it. There exists a critical frequency value i.e. $\mathcal{F}_{crit}$, also called energy-efficient frequency, below which the DVS does not remain effective. Critical frequency value depends on ρ_{static} and ρ_{ind}. In this paper, the default value of ρ_{static} and ρ_{ind} has been set to 5% and 15% of maximum dependent power consumption, respectively. When the processor is not executing anything, then it switches to idle state where it consumes the power equal to 20% of maximum dependent power consumption. The sleep state power consumption has been set to negligible value in the current work.

C. *Fault Model*

Permanent faults occur rarely in the system, but transient faults are very common. Generally, transient faults are modeled with Poisson distribution [6], [22]. It has been shown by Melhem et al. that reduced supply voltage results in exponentially increased fault rate [1]. Thus, average arrival rate of soft error caused by transient faults in DVFS enabled processor at scaled processing frequency $\mathcal{F}$ is as follows:

$$\gamma(\mathcal{F}) = \gamma_0 . g(\mathcal{F}) \quad (4)$$

where γ_0 is the average error rate corresponds to the maximum processing frequency $\mathcal{F}_{max}$. Zhu et.al gave an exponential fault rate model for soft errors caused due to transient faults which is as follows:

$$\gamma(\mathcal{F}) = \gamma_0 . g(\mathcal{F}) = \gamma_0 . 10^{\frac{s(1-\mathcal{F})}{1-\mathcal{F}_{crit}}} \quad (5)$$

where $s(> 0)$ indicates the sensitivity of the rate of soft error to DVFS and $\mathcal{F}_{crit}$ is the minimum energy efficient processing frequency.

Problem Statement: Given a set of real-time periodic tasks on a dual processor system, execute all tasks within given deadline and minimize the energy consumption by determining

1. Allocating tasks such that primary and backup copy of task should not be on the same processor.
2. The promotion time of backup tasks for execution.

In the upcoming section, the required goal has been achieved in two phases.

IV. ENERGY AWARE FAULT TOLERANT TASK SCHEDULING

Despite using the processors in Standby-Sparing configuration, if the processors are used in a mixed manner i.e. primary and backup tasks are allocated on both the processors, then the processor capacity can be utilized effectively, and throughput can be increased. The energy consumption of the system also reduces significantly. The scheme proposed in this paper consists of two phases:

- Allocation Phase
- Scheduling Phase

Partitioning of tasks among two processors has been done in Allocation Phase whereas Scheduling phase schedules the allocated tasks on processor using Fixed-Priority Scheduling Algorithm.

A. *Allocation Phase*

Several heuristics for task allocation have been proposed in literature such as Next-Fit, Best-Fit, First-Fit, and Worst-Fit. To attain better load balancing, further heuristics were proposed that first sort the tasks in non-increasing order of utilization and then allocate to processors. These are Next-Fit-Decreasing, Best-Fit-Decreasing, First-Fit-Decreasing, and Worst-Fit-Decreasing. Among these, Worst-Fit-Decreasing accomplishes the task of load-balancing in best way. The balanced allocation will lead to reduced Response-Time.

Another Modified-Worst-Fit-Decreasing (MWFD) scheme was proposed by Wei et al. for energy efficient multiprocessor system[23]. It has the property to obtain energy efficiency in changing workload situations. MWFD opens all the processors at the beginning and allocates the tasks to processors having minimum workload.

MWFD approach for task allocation has been used in this work. The workload of primary tasks has been set according to the frequency $\mathcal{F}$ = min $(U_{tot}, \mathcal{F}_{crit})$. Primary tasks are allocated before the backup tasks. While allocating backup tasks, care must be taken that primary copy and backup copy of task should not be on the same processor. If both primary and backup copy are on the same processor, then system will not be able to tolerate permanent fault.

Each processor ω_j contains the task-set ψ_j, which contains both primary tasks as well as backup tasks. The allocation process is successful only if each processor ω_j has a schedulable task-set ψ_j at a given frequency $\mathcal{F}$.

1) *Feasibility Analysis*

The taskset ψ_j on processor '*j*' consists of mix of primary tasks and backup tasks. Each processor must be checked for the feasibility of its taskset ψ_j. Time-Demand Analysis technique [2] has been used for checking feasibility of task-set ψ_j.

2) *Algorithm for Allocation*

MA Algorithm phase -1 : - Allocation of tasks on processors

```
Input: Taskset $\tau$ and Taskset $\tau^B$.

Output: Taskset $\psi_j$ on processor $j$ consisted of mix of primary tasks and
backup tasks

1. Set $\mathcal{F}$ = min ($U_{tot}$, $\mathcal{F}_{crit}$);
2. Arrange all tasks in both task sets
   in non-increasing order of their
   utilization.
3.  do
4.   Allocate tasks from $\tau$ using MWFD.
5.   Allocate tasks from $\tau^B$ using MWFD.
6. Check the schedulability of task set
   with Time Demand Analysis.
7.     if schedulability = true, then
8.       return true;
9.     else
10.       Increment the frequency $\mathcal{F}$ by
   0.01.
11.  while ($\mathcal{F} < 1$)
12.  return false;
```

B. *Scheduling Phase*

The Scheduling Phase schedules the jobs as per well-known Fixed-Priority algorithm i.e. Rate Monotonic Scheduling algorithm (RMS). A Ready Queue is maintained for each processor which gives the next ready task for execution on the processor. Jobs are sorted according to the priority in the queue. Job with smaller period will have higher priority than the job with larger period. In the ready queue, primary job is added as soon as it arrives, but the backup job is added at its promotion time.

The promotion time κ_i for backup task τ_i^B can be calculated as:

$$\kappa_i = D_i - R_i \tag{6}$$

where R_i is Response Time of τ_i^B and is calculated using Time-Demand Analysis technique [24][25][2].

Backup job is scheduled as late as possible using the Dual-Priority Scheduling [2]. In this scheme, a lower priority queue is maintained on each processor that contains the backup copies of primary jobs running on alternate processor. A backup job $\tau_{i,k}^B$ is promoted to ready queue at time $\kappa_{i,k}$. As soon as the primary job $\tau_{i,k}$ completes on ω_j , the backup copy $\tau_{i,k}^B$is deleted from the ready queue/lower priority queue of the alternate processor $\omega_{l\neq j}^B$ on which it was scheduled. The scheduling algorithm has been elaborated below:

MA Algorithm phase -2 : - Scheduling of tasks

```
Input: Taskset $\psi_j$ on processor j
consisted of mix of primary tasks and
backup tasks.

    1. If job $\tau_{i,k}$ is released on processor
       $\omega_j$, then a backup copy $\tau_{i,k}^B$ is added
       to the lower priority queue of
       alternate processor $\omega_{l\neq j}^B$.
    2. At promotion time $\kappa_{i,k}$, $\tau_{i,k}^B$ is added
       to Ready Queue on processor $\omega_l^B$.
    3. Take the job $\tau_{i,k}$ from the Ready Queue
       and dispatch it.
    4. If $\tau_{i,k}$ is primary, then
    5.     if $\tau_{i,k}$ is not faulty
    6.     Find the backup job $\tau_{j,k}^B$ and
       delete it.
    7. If Ready Queue is empty,
    8.    Find next_time_to_arrival.
    9. If next_time_to_arrival > critical
       time, then
    10.  Put processor to sleep state.
    11.  Else
    12.  Put processor to idle state.
```

V. EVALUATION

A. *Task Generation*

For each task set 100 periodic tasks are generated. UUnifast [26] [2] algorithm is used to generate the utilizations. Periods are randomly generated between 10ms to 100ms. Deadline is set equal to period and worst-case execution time is calculated as a product of utilization and period.

Generally, real-time tasks do not execute for its worst-case execution time. The Actual execution time is always less than the WCET. Here, ACET has been calculated according to [2][27].

B. *Simulation Results*

Energy consumption has been normalized with respect to No-Power Management Scheme. Fig. 1 and Fig.2 are showing the normalized energy consumption using Standby-Sparing (SS) and Mixed Allocation (MA) with worst-case execution time (WCET) and Actual execution time (ACET), respectively. The difference between the energy consumption increases as the utilization of task-set increases. This is because SS scheme executes more backup copies at high utilization.

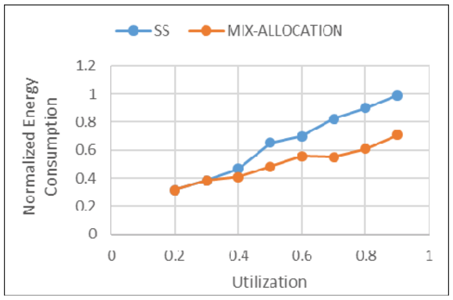

Fig. 1. Normalized Energy Consumption with WCET

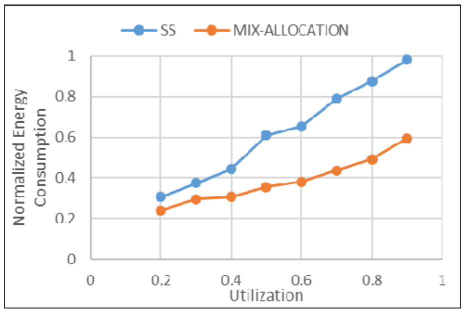

Fig. 2. Normalized Energy Consumption with ACET

Fig. 3 and Fig. 4 shows the extra number of jobs executed by MA scheme than SS scheme. Thus, the new scheme also increases the throughput.

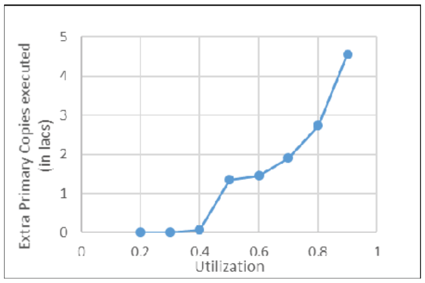

Fig. 3. Extra jobs executed by the MA scheme with WCET

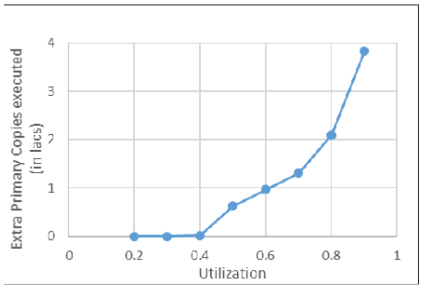

Fig. 4. Extra jobs executed by the MA scheme with ACET

Fig. 5 and Fig. 6 shows the difference between the backup copies executed by MA and SS scheme. The increased number of backup execution will also increase the energy consumption because the backup copy is executed at maximum frequency.

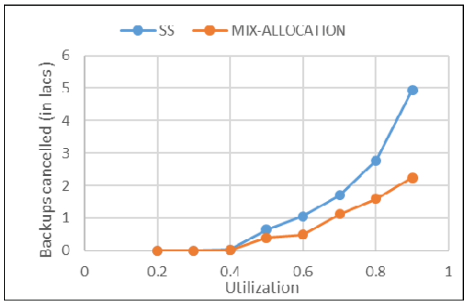

Fig. 5. Number of backups Executed with ACET

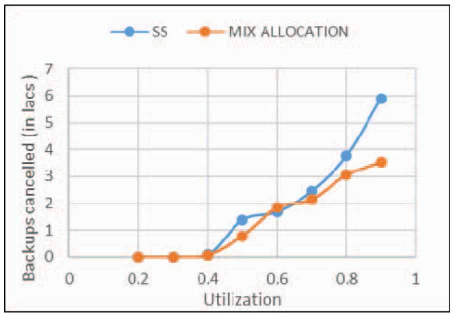

Fig. 6. Number of backups Executed with WCET

VI. Conclusion

The standby sparing scheme and MA scheme for fixed priority real time tasks have been discussed in this paper. From the simulation results, it is concluded that using the processors to execute both the primary and backup in a mixed manner will result in better energy management along with improved system utilization and throughput. The Mix-Allocation strategy can be easily extended to multiprocessor systems.

References

[1] 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, 2004. ICCAD-2004.*, 2004, pp. 35–40.

[2] M. a. Haque, H. Aydin, and D. Zhu, "Energy-aware standby-sparing for fixed-priority real-time task sets," *Sustain. Comput. Informatics Syst.*, pp. 1–13, 2014.

[3] D. Zhu and H. Aydin, "Reliability-Aware Power Management for Real-Time Embedded Systems," .

[4] D. Zhu and H. Aydin, "Energy Management for Real-Time Embedded Systems with Reliability Requirements," *Iccad*, pp. 528–534, 2006.

[5] B. Zhao, H. Aydin, and D. Zhu, "Energy management under general task-level reliability constraints," *Real-Time Technol. Appl. - Proc.*, pp. 285–294, 2012.

[6] B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Trans. Ind. Informatics*, vol. 6, no. 3, pp. 316–328, 2010.

[7] B. Z. B. Zhao, H. Aydin, and D. Z. D. Zhu, "Enhanced reliability-aware power management through shared recovery technique," *2009 IEEE/ACM Int. Conf. Comput. Des. - Dig. Tech. Pap.*, pp. 63–70, 2009.

[8] B. Zhao, H. Aydin, and D. Zhu, "Shared Recovery for Energy Efficiency and Reliability Enhancements in Real-time Applications with Precedence Constraints," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 2, p. 23:1–23:21, 2013.

[9] Q. Han, L. Niu, G. Quan, S. Ren, and S. Ren, "Energy efficient fault-tolerant earliest deadline first scheduling for hard real-time systems," *Real-Time Syst.*, vol. 50, no. 5–6, pp. 592–619, 2014.

[10] R. Melhem, D. Mossé, and E. Elnozahy, "The Interplay of Power Management and Fault Recovery in Real-Time Systems," *IEEE Trans. Comput.*, vol. 53, no. 2, pp. 217–231, 2004.

[11] M. Salehi, M. K. Tavana, S. Rehman, M. Shafique, A. Ejlali, and J. Henkel, "Two-State Checkpointing for Energy-Efficient Fault Tolerance in Hard Real-Time Systems," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 24, no. 7, pp. 2426–2437, 2016.

[12] Y. Zhang and K. Chakrabarty, "Dynamic adaptation for fault tolerance and power management in embedded real-time systems," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 2, pp. 336–360, 2004.

[13] M. Haque, H. Aydin, and D. Zhu, "On Reliability Management of Energy-Aware Real-Time Systems through Task Replication," *IEEE Trans. Parallel Distrib. Syst.*, vol.

9219, no. c, pp. 1–1, 2016.

[14] M. a. Haque, H. Aydin, and D. Zhu, "Energy-aware task replication to manage reliability for periodic real-time applications on multicore platforms," *2013 Int. Green Comput. Conf. Proceedings, IGCC 2013*, 2013.

[15] M. Salehi, M. K. Tavana, and S. Rehman, "DRVS : Power-Efficient Reliability Management through Dynamic Redundancy and Voltage Scaling under Variations," pp. 6–11, 2015.

[16] S. Aminzadeh and A. Ejlali, "A Comparative Study of System-Level Energy Management Methods for Fault-Tolerant Hard Real-Time Systems," *Computers, IEEE Transactions on*, vol. 60, no. 9. pp. 1288–1299, 2011.

[17] A. Ejlali, B. Al-Hashimi, and P. Eles, "A standby-Sparing Technique with Low Energy-Overhead for Fault-Tolerant Hard Real-Time Systems," in *7th IEEE/ACM international conference on Hardware/software codesign and system synthesis,* 2009, no. ACM, pp. 193–202.

[18] A. Ejlali, B. M. Al-hashimi, and P. Eles, "Low-Energy Standby-Sparing for Hard Real-Time Systems," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 31, no. 3, pp. 329–342, 2012.

[19] Y. Guo, D. Zhu, H. Aydin, L. T. Yang, and S. Member, "Energy-Efficient Scheduling of Primary/Backup Tasks in Multiprocessor Real-Time Systems (Extended Version)," pp. 1–21, 2013.

[20] M. Bambagini, M. Marinoni, H. Aydin, and G. Buttazzo, "Energy-Aware Scheduling for Real-Time Systems," *ACM Trans. Embed. Comput. Syst.*, vol. 15, no. 1, pp. 1–34, 2016.

[21] S. Mittal, "A Survey of Techniques For Improving Energy Efficiency in Embedded Computing Systems," *Proc. Int. J. Comput. Aided Eng. Technol.*, pp. 440–459, 2013.

[22] Y. Zhang and K. Chakrabarty, "Energy-aware adaptive checkpointing in embedded real-time systems," *Proc. - Design, Autom. Test Eur. DATE*, pp. 918–923, 2003.

[23] T. Wei, P. Mishra, K. Wu, and H. Liang, "Fixed-priority allocation and scheduling for energy-efficient fault tolerance in hard real-time multiprocessor systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1511–1526, 2008.

[24] N. Audsley, a Burns, M. Richardson, K. Tindell, and a J. Wellings, "Applying New Scheduling Theory To Static Priority Preemptive Scheduling," vol. 17, 1993.

[25] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and\naverage case behavior," *[1989] Proceedings. Real-Time Syst. Symp.*, pp. 0–5, 1989.

[26] E. Bini, "Measuring the Performance of Schedulability Tests ∗," pp. 129–153, 2005.

[27] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," *Des. {Automation} {Conference}, 1999. {Proceedings}. 36th*, no. c, pp. 134–139, 1999.