Power consumption analysis of fault tolerant real-time systems

Sandra Djosic[1], Milun Jevtic[2] and Milunka Damnjanovic[3]

Abstract – **In this paper we analyze the power consumption of fault tolerant real-time systems. We consider real-time systems where the fault tolerance is achieved running the task affected by a transient fault again using time redundancy. The power consumption analysis is done using one heuristic-based dynamic voltage and frequency scaling algorithm which we designed. The simulation results show that our proposed algorithm can be successfully used for power consumption according to fault tolerance analysis.**

Keywords – **Dynamic voltage and frequency scaling, Fault tolerance, Real-time systems.**

I. Introduction

Real-time systems (RTSs) have been designed in order to be safe and extremely reliable. They are usually realized as real time systems with the ability of tolerating some faults. A fault-tolerant RTS has to ensure that faults in the system do not lead to a failure. Faults could be transient, permanent or intermittent faults. Among these, the transient faults are much more common than faults of other two types. Transient faults have the feature that they occur and then disappear so fault tolerance can be achieved running the task affected by a transient fault again (i.e. re-executing the task). It means that time redundancy can be used as fault-tolerance techniques by using free slack time in the system schedule to perform recovery executions, [1], [2].

Energy efficiency is also crucial to many real-time systems. Dynamic Voltage and Frequency Scaling (DVFS) is the most popular and widely deployed technique for reducing power consumption of processors [3], [4], [5]. Nowadays, DVFS is a commonly used technique for energy management and is supported by many commercial processors [6].

Fault tolerances through time redundancy as well as energy management through frequency and voltage scaling have been well studied in the context of real-time systems. But simply applying fault recovery techniques and energy minimization techniques one after the other results in inferior quality. These techniques use free slack time and since free slack time is a limited resources, it is obvious that more slack time for DVFS technique means less time for fault tolerance, and vice versa. Therefore, there is a tradeoff between low energy consumption and high fault-tolerance. In accordance with that, we designed one heuristic-based DVFS algorithm and used it for RTSs analysis. The analysis refers to the power consumption and fault tolerance through time redundancy for different number available operating frequency levels of the processor used in the RTSs.

The rest of the paper is organized as follows. The first part of Section II describes real-time system, power, fault and feasibility models we used in the paper. The second part of Section II introduces our proposed heuristic-based DVFS algorithms. Section III gives the simulation results and finally, Section IV presents our conclusions.

II. Models and algorithm description

A. Models description

For a system model, we assume one uniprocessor RTS with variable processor's operating frequency f_j ($j = 1, ..., m$) where $f_j < f_{j+1}$. Changing the operating frequency of the processor the voltage also changed and it could be switched between m values. This system can be used for one real-time task set execution. We assume a set of n periodic real-time tasks, $\Gamma=\{\tau_1,..., \tau_n\}$ where each tasks are defined by a period T_i, worst case execution time (WCET) C_i, deadline D_i and priority p_i. We assume that $D_i \le T_i$, for $i = 1, 2, ..., n$. The WCET of real-time tasks corresponds to executing the task at the maximum frequency f_m. For simplicity, we assume that the WCET of a task scales linearly with the processing speed. So, if we scale the operating frequency by a factor α, then WCET must be scaling by factor $1/\alpha$, i.e.

$$C_i(f_j) = C_i(f_m)\, f_m / f_j.$$

Power consumption of an active processor can be modeled as

$$P_A(f) = P_d(f) + P_{ind},$$

where $P_d(f)$ and P_{ind} are frequency dependent power and frequency independent power respectively [7]. Frequency dependent power is

$$P_d(f) = V^2(f)\, C_{ef}\, f$$

where V is supply voltage and it is a function of operating frequency, C_{ef} is the switch capacitance and f is the operating frequency. Beside power, for DVFS techniques energy is equally important and it is defined as the integral of power over time.

We assume that faults can occur during execution of any task. We consider transient faults and assume that the consequences of a fault can be eliminated by simple re-execution of the affected task at its original priority level and at its original oparting frequency. The re-execution of the

[1]Sandra Djosic is with the University of Nis, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Nis, Serbia, E-mail: sandra.djosic@elfak.ni.ac.rs.

[2]Milun Jevtic is with the University of Nis, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Nis, Serbia, E-mail: milun.jevtic@elfak.ni.ac.rs.

[3]Milunka Damnjanovic is with the University of Nis, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Nis, Serbia, E-mail: milunka.damnjanovic@elfak.ni.ac.rs.

corrupted task must not violate timing constraints of any task in Γ.

For checking the feasibility of fault tolerant real-time task set we use the response time analysis (RTA). In the RTA, the fault-tolerance capability of a RTS is represented by a single parameter, T_F, which corresponds to minimum time interval between two consecutive faults that the RTS can tolerate. More about RTA can be found in [8], [9]. The basic equation characterize for RTA is Eq. (1).

$$R_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j + \left\lceil \frac{R_i^n}{T_F} \right\rceil \max_{j \in hp(i) \cup i} (C_j) \tag{1}$$

With Eq. (1) the response time R_i of a task τ_i could be calculated. This equation has three main addends. The first is WCET C_i for a task τ_i. The second presents interference due to preemption by higher priority tasks. We use hp(i) to denote the set of tasks with higher priorities than i, hp(i)=$\{\tau_j \in \Gamma | p_j > p_i\}$. The third addend refers to possible faults in the system. If we assume that inter-arrival time between faults is T_F then there can be at most $\left\lceil \frac{R_i}{T_F} \right\rceil$ faults during the response time R_i of task τ_i. Since these faults could occur during the execution of task τ_i or any higher priority task which has preempted τ_i, each fault may add $\max_{j \in hp(i) \cup i} (C_j)$ to the response time of task τ_i. So, the third addend in Eq. (1) presents an extra time needed tasks recovery due to faults.

Since R_i appears on both sides Eq. (1) is recurrence relations which starts with $R_i^0 = C_i$. The solution is found when $R_i^{n+1} = R_i^n$. If during the iteration process we get that $R_i^{n+1} > D_i$ then task τ_i is infeasible and iteration process must be terminated.

B. Algorithm description

In order to solve the tradeoff problem between low energy consumption and high fault-tolerance, we propose one heuristic DVFS algorithm. The proposed algorithm has to find appropriate execution frequency for each task, from the real-time tasks set, such that energy consumption is minimal when faults are presence. Fig. 1 gives the algorithm in pseudo code form.

For this purpose we created a heuristic-based algorithm to The RTA is the basic of our proposed algorithm. This analysis is used to guarantee feasibility of real-time tasks set and fault tolerance. The input parameters for the algorithm are:

- frequency f_j (j=1,..., m) where $f_j < f_{j+1}$ and m is number of frequency levels;
- characteristics for all n real-time tasks from the set: period T_i, worst case execution time C_i, priority p_i and deadline D_i, for i=1,..., n;
- minimum time interval between two consecutive faults T_F.

```
Input:  operating frequency levels $f_j$ ($j$=1..$m$),
        characteristics for $n$ real time tasks ($C_i$, $D_i$, $T_i$, $p_i$),
        fault tolerant constraint ($T_F$)
_________________________________

(1) for each Task in TaskSet set Task's_Freq to $f_m$ and set
    Task's_Key to true;
(2) repeat step (3) to (7) until there are true Task's_Key in the
    TaskSet;
(3) for each unlock Task in TaskSet do
(4)     temporarily set Task's_Freq to Lower_Task's_Freq;
(5)     if new TaskSet is not feasible
(6)         then set Task's_Key to false;
(7)         else calculate ΔPower as Power(Task's_Freq) –
            Power(Lower_Task's_Freq);
(8) find Task with maximum ΔPower and set Task's_Freq to
    Lower_Task's_Freq;

_________________________________
Output: TaskSet with new frequency assigne to each Task
```

Fig. 1. Pseudo code of the proposed algorithm

The algorithm starts with assigning the maximum operating frequency, f_m, to each real-time task, step (1). Also, at the beginning, all tasks are allowed to change the frequency - we say that all tasks are unlocked. An iteration of the algorithm decreases the frequency of one task for one frequency level. The chosen task is one for which the frequency decrement yields maximum power reduction among all unlocked tasks provided that tasks set remains feasible. To find such task, the algorithm checks all currently unlocked task. For example, frequency index of one unlock task τ_i is temporarily decreased for one frequency level, i.e. from f_j to f_{j-1}, step (4), and feasibility of task-set is tested using Eq. (1), step (5). If task-set is not feasible, τ_i is locked, step (6). Otherwise, if task-set is feasible, the difference between power consumption of τ_i at lower (f_{j-1}) and higher (f_j) frequency is calculated, step (7). Then, τ_i's frequency is changed back to f_i. After checking all tasks, one that remains unlocked and provides the maximal power reduction is selected, and its frequency index is decremented, step (8). Additionally, the selected task is locked if its new frequency equals 1, i.e. corresponds to the lowest execution frequency, f_1. After that, the algorithm enters the next iteration. The algorithm finishes when there are no more unlocked tasks. The frequency assignment to each task is algorithm's output. We previously proved the proposed heuristic algorithm and more about that can be found in [10].

III. Simulation results

We realized simulator based on our proposed heuristic DVFS algorithm. The input parameters of the simulator are number of real-time task and their real-time characteristics: minimum inter-arrival time T_i, worst case execution time C_i on maximum operating frequency f_m, deadline D_i and priority p_i. Also, input parameters are processor's voltage and frequency levels and fault constraints T_F.

TABLE I
TASKS SET FROM GENERIC AVIONICS PLATFORM

τ_i	p_i	T_i=D_i (ms)	C_i (ms)
Nav_Status	1	1000	1
BET_E_Status_Update	2	1000	1
Display_Stat_Update	3	200	3
Display_Keyset	4	200	1
Display_Stores_Update	5	200	1
Nav_Steering_Cmds	6	200	3
Tracking_Target_Upd	7	100	5
Display_Hook_Update	8	80	2
Display_Graphic	9	80	9
Nav_Update	10	59	8

We performed simulations with a number of synthesized real-time task sets and few real-world applications. The characteristics of one real-world application are summarized in Table I. It is a task set taken from the Generic Avionics Platform (GAP) used in [11]. For the processor's frequency levels we used data for Transmeta Crusoe procesor from [12]. The relevant parameters for the processor are listed in Table II.

TABLE II
PROCESSOR FREQUENCIES, VOLTAGES AND POWER

CPU Frequency (MHz)	Voltage (V)	CPU Power (W)
300	1.2	1.3
400	1.225	1.9
533	1.35	3
600	1.5	4.2
667	1.6	5.3

First, we assumed that there were no faults in the system. With this assumption, we used our proposed algorithm to find the appropriate execution frequencies for each real-time task that lead to the maximum energy savings. Fig. 2 shows the simulation results for GAP task set and Transmeta Crusoe processor. We performed simulation for different number of available frequency (voltage) levels. That is represented on the x-axis where 2, 3, 4 and 5 frequency levels include set of frequencies (667MHz, 300MHz), (667MHz, 600MHz, 300MHz), (667MHz, 600MHz, 400MHz, 300MHz), (667MHz, 600MHz, 533MHz, 400MHz, 300MHz) respectively. The y-axis represents the power reduction calculated in percents. This reduction is presented as power saving with respect to the power consumption at maximum frequency.

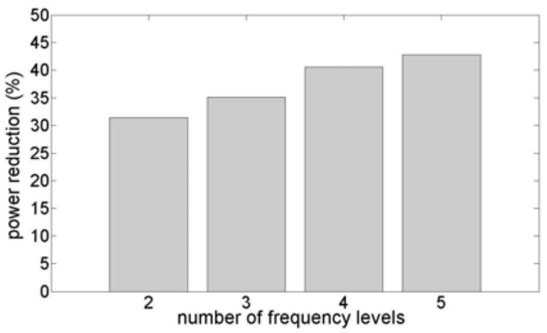

Fig. 2. Power consumption according to number of frequency levels in the absence of faults in the RTS

It can be concluded that power reduction is better when more voltage levels are included. The maximum energy savings is 42.8% for 5 levels and the minimum savings is 31.5% for only 2 frequency levels. The energy reduction is significant even for low number of frequency levels and this clearly shows the effectiveness of our proposed algorithm.

Our next step, in the simulation process, was to consider possible faults appearance in the RTS. This is represented by a single parameter, T_F, which corresponds to minimum time interval between two consecutive faults that the RTS can tolerate. For the input parameters of the simulator we used the same task set and the same processor. Fig. 3 shows the simulation results for different number of available frequency levels. We used the same sets of frequencies (667MHz, 300MHz), (667MHz, 600MHz, 300MHz), (667MHz, 600MHz, 400MHz, 300MHz), (667MHz, 600MHz, 533MHz, 400MHz, 300MHz). The x-axis of the Fig. 3 represents the ratio of T_{Fmax} to T_F. T_{Fmax} is minimum time interval between two consecutive faults that the task set can tolerate on maximal executing frequency and T_F is input simulation parameter. This axis represents the normalized T_F value which is proportional to fault tolerance of the task set. As fault tolerance proportional to time redundancy this axes also could represent free slack time in the systems. The y-axis represents the power saving with respect to the power consumption at maximum frequency calculated in percents.

According to number of available frequency levels the simulation was done for four possible scenarios. All four scenarios indicate the same fact that power reduction leads to less fault tolerance and vice versa. Now due to simulation results, we can better perceive the tradeoff between power consumptions and fault tolerance. For example, let's suppose that power reduction demands are between 40% and 45%. It can be seen, from the Fig. 3, that processor with 4 or 5 frequency levels could fulfill these demands. Also, fault tolerances vary for the given power reduction interval. The best is to choose one with maximal tolerances.

Also, it can be concluded that power reduction is better when more voltage levels are included. With larger number of frequency levels there are more possible task-frequency mapping, so the chance of finding solutions with lower energy becomes higher.

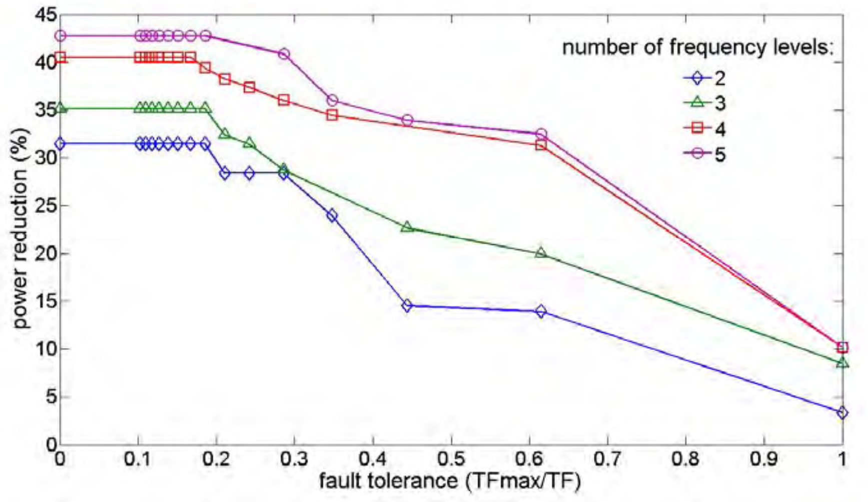

Fig. 3. Power consumption according to fault tolerance for different number of frequency levels

IV. Conclusion

In this paper the power consumption of the real-time systems according to fault tolerance through time redundancy analysis are given. The analysis is based on the heuristic DVFS algorithm which we realized. The proposed algorithm offers the possibility to study the trade-off between energy efficiency and fault tolerance for real-time task sets. Generally, this trade-off in discrete systems is NP-hard, so the heuristic-based approach is imposed as possible solution for RTSs analysis.

On the basis of proposed heuristic-based algorithm we realized simulator. Our simulations results show that power reduction is better when more operating frequency levels are included. This is valid for the case of the absence or the presence of the faults, in the RTS. Our opinion is that this simulator could be successfully used in the RTS design process.

Acknowledgement

This paper is supported by Project Grant III44004 financed by Ministry of Education and Science, Republic of Serbia.

References

[1] S. Đošić, M. Jevtić, “*Scheduling in RTS Using Time Redundancy for System Recovery After Faults*”, Proceedings of papers, Indel 2004, Banja Luka, pp. 146-149, November 2004.

[2] S. Đošić, M. Jevtić, M. Damnjanović, “Analysis of possibilities to overcome the transient faults in real-time systems with time redundancy”, XLVI International scientific conference on information, communication and energy systems and technologies, ICEST 2011, Proceedings of Papers, volume 2, pp 417- 420, Serbia, Niš, June 29 - July 1, 2011.

[3] K. Woonseok, S. Dongkun, Y. Han-Saem, K. Jihong, M. Sang, “*Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems*”, Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS’02*)*, pp. 219 – 228, 2002.

[4] A. S. Ahmadian, M. Hosseingholi, A. Ejlali, “*A Control-Theoretic Energy Management for Fault-Tolerant Hard Real-Time Systems*”, 2010 IEEE International Conference on Computer Design, pp. 173-178, 2010.

[5] R. M. Santos, J. Santos, J. D. Orozco, “*Power saving and fault-tolerance in real-time critical embedded system*”, Journal of system Architecture 55, pp. 90-101, 2009.

[6] “*Intel PXA270 Processor Electrical, Mechanical and Thermal Specification Data sheet*”, www.phytec.com/pdf/datasheets/PXA270_DS.pdf, 2005.

[7] Z. Dakai, R. Melhem, D. Mosse, “*The Effects of Energy Management on Reliability in Real-Time Embedded Systems*”, Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design, pp. 35-40, 2004.

[8] S. Đošić, M. Jevtić, “*Analysis of Real-Time Systems Timing Constrains*”, SSSS2010, 3rd Small Systems Simulation Symposium 2010, February, 12-14, Faculty of Electronic Engineering, Niš, Serbia, pp 56-60, 2010.

[9] G. Lima, A. Burns, “*An Optimal Fixed-Priority Assignment Algorithm for Supporting Fault-Tolerant Hard Real-Time Systems*”, IEEE Transaction on Computers, Vol. 52, No. 10, pp. 1332-1346, October 2003.

[10] S. Đošić, M. Jevtić, “Dynamic voltage scaling for real-time systems under fault tolerance constraints”, accepted for publication on 28th International Conference on Microelectronics, MIEL2012.

[11] Locke, C. D., Vogel, D. R., Mesler, T. J., “Building a Predictable Avionics Platform in Ada: A Case Study”, Proceedings of IEEE Real-Time Systems Symposium, pp. 181–189, 1991.

[12] Ying Zhang, Krishnendu Chakrabarty, “Task Feasibility Analysis and Dynamic Voltage Scaling in Fault-Tolerant Real-Time Embedded Systems” Proceedings of Design, Automation and Test in Europe Conference and Exhibition, Vol.2, pp. 1170 – 1175, 2004.