Energy efficiency and fault tolerance analysis of hard real-time systems

Sandra Đošić and Milun Jevtić

Abstract - In this paper the tradeoff between dynamic voltage and frequency scaling (DVFS) techniques and faults tolerance are considered. We analyzed this tradeoff using one heuristic-based DVFS algorithm which we designed. The proposed algorithm minimizes the energy consumption of one real-time task set when the transient faults are exceeded. It is assumed that the tasks execute on processors with variable frequency and voltage levels. The simulation results show that our proposed algorithm can be used for real-time systems analysis from the perspective of finding compromise between energy efficiency and fault tolerance.

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

I. Introduction

Hard real-time systems (HRTS) play an important role in many areas of daily life: robotics, cosmic research, automotive industry, process control, factory automation… Those systems 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 HRTS has to ensure that faults in the system do not lead to a failure.

Dependability of one real-time system can be affected by different kinds of faults, including transient, permanent and intermittent faults. Among these, the transient faults are much more common than faults of other two types. Since transient faults have the feature that they occur and then disappear, 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].

Beside high level of dependability, energy efficiency is crucial to many real-time systems due to their limited energy supply and severe thermal constraints of the operating environment. Dynamic Voltage and Frequency Scaling (DVFS) is the most popular and widely deployed technique for reducing power and energy consumption of processors [2], [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.

For HRTS that require both fault tolerance and energy efficiency, there is a lack of efficient solutions. Simply applying fault recovery techniques and energy minimization techniques one after the other only results in inferior quality. This is because minimizing energy first may not leave enough slack for fault recovery, and minimizing energy after fault recovery reservation treats normal task executions and re-executions (for fault recovery) equally, which is equivalent to optimizing the worst case that happens rarely. 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.

The tradeoff between DVFS techniques and faults tolerance is focus of this paper. In accordance with that, we designed one heuristic-based DVFS algorithm and used it for energy efficiency and fault tolerance analysis of HRTS.

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

II. Models description

A. System model

We consider one uniprocessor real-time system with variable CPU frequency f_j (j=1,..., m) where $f_j < f_{j+1}$. The voltage and the operating frequency of the CPU may be switched between m values. This system can be used for one real-time task set execution. We assume a set of n real-time tasks, $\Gamma=\{\tau_1,..., \tau_n\}$ where each tasks are defined by a minimum inter-arrival time T_i, worst case execution time (WCET) C_i and deadline D_i. We assume that $D_i \leq T_i$, 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.

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

Each task is assigned a unique priority p_i and all of them are periodic, fully preemptive and independent. Algorithm for scheduling real-time tasks could be any priority assignment algorithm, [7].

Sandra Đošić and Milun Jevtić are with the Department of Electronics, Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: {sandra.djosic, milun.jevtic}@elfak.ni.ac.rs.

B. Power model

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 [8]. 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 frequency. Beside power, for DVFS techniques energy is equally important and it is defined as the integral of power over time.

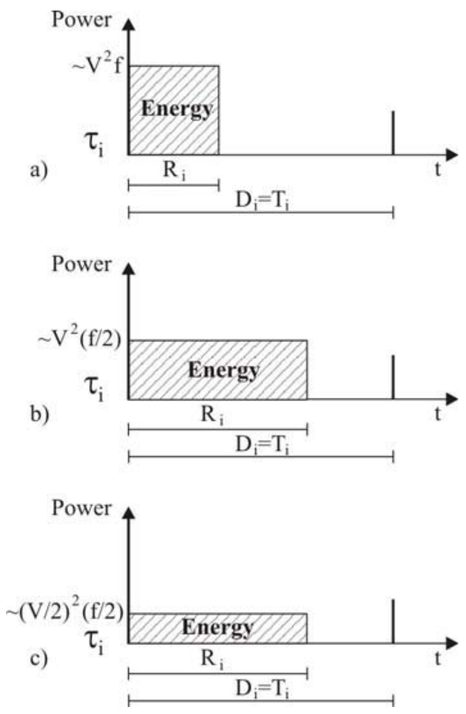

Fig. 1 Power and energy consumption for real-time task τ_i:
a) for frequency f and voltage V
b) for frequency $f/2$ and voltage V
c) for frequency $f/2$ and voltage $V/2$

Fig. 1 a) illustrates energy consumption of one real-time task τ_i on operating frequency f and for supply voltage V. Energy of real-time task τ_i is proportional to the marked rectangle area. Fig. 1 b) presents the situation when the operating frequency f is reduced by half and because of that task needs more time to execute. In that situation processor's power consumption is lower but the energy consumption remains the same. Fig. 1 c) shows the influence on power and energy consumption when supply voltage V is reduces by half. Lowering the supply voltage V the significant amount of energy could be reduced, because of the quadratic relation between power and V. The maximum energy reduction is obtained by lowering the supply voltage and operating frequency simultaneous.

C. Fault model

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 CPU frequency. The re-execution of the corrupted task must not violate timing constraints of any task in Γ.

D. Feasibility model

In our approach we use the response time analysis (RTA) to check the feasibility of fault tolerant real-time task set. 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 [9], [10].

The basic equation characterize for RTA is Eq. (1).

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

With Eq. (1) the response time R_i of a task τ_i could be calculated. Eq. (1) 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)=$\{\tau_j \in \Gamma \mid p_j > p_i\}$. 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.

Fig. 2 illustrates RTA applied on one simple RTS with two periodic real-time tasks τ_i and τ_j. Two faults occur in the system and the time between these two consecutive faults is T_F.

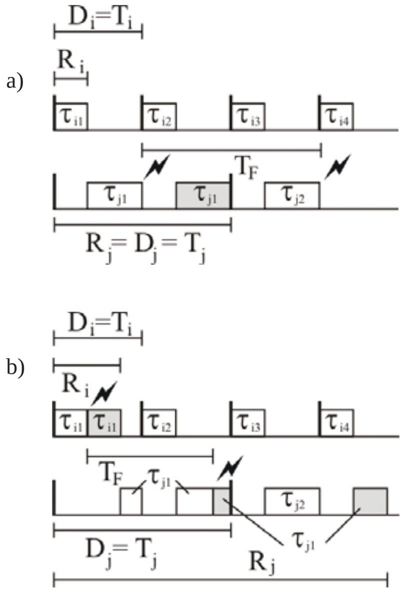

Fig. 2 a) T_F is long enough and RTS is fault tolerant
b) T_F is not long enough that RTS stays fault tolerant

Fig. 2 a) presents the situation when first fault occurs just a little bit before the end of tasks τ_{j1}. System overcomes this fault by executing task τ_{j1} again. This is situation when T_F is long enough and real-time system can overcome these faults. System of these two tasks are schedulable i.e. both tasks execute before their deadlines, D_i and D_j. Response time of tasks τ_i and τ_j are the output results of RTA and they are also shown on Fig. 2 a).

Fig. 2 b) presents scheduling of the same real-time tasks τ_i and τ_j when two faults occur in the system. Now, time between two consecutive faults T_F is not long enough and real-time system cannot tolerate these faults. First fault occurs just a little bit before the end of tasks τ_{i1} execution. Real-time system can overcomes this fault by executing task τ_{i1} again. Second fault occurs just a little bit before the end of tasks τ_{j1} execution. Now time redundancy is not enough to tolerate this fault. Systems starts procedure for overcoming fault by executing task τ_{j1} again but timing characteristics of tasks τ_{j1} cannot be satisfied and τ_{j1} missing its deadline i.e. $R_j > D_j$. This is not acceptable in one hard real-time system, so in this case real-time system is not fault tolerant.

III. Proposed DVFS algorithm

The focus of this section is to explain our proposed DVFS policy which fulfils energy efficiency and faults tolerance requirements. For this purpose we created a heuristic-based algorithm to find appropriate execution frequency for each task, from the real-time tasks set, that minimize energy consumption when faults are absent. The RTA is the basic of our proposed algorithm. This analysis is used to guarantee feasibility of real-time tasks set and fault tolerance.

Fig. 3 shows the pseudo code of our proposed algorithm. The input parameters for the algorithm are:

- CPU 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: inter-arrival time 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: CPU 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. 3 Heuristic algorithm solution

The algorithm starts with assigning the maximum execution 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), as:

$$\Delta P_i = (C_i(f_j)-C_i(f_{j-1}))/T_i.$$

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.

IV. Simulation results

The simulator we realized is based on our proposed heuristic-based algorithm. The input parameters of the simulator are real-time task set characteristics, processor's voltage and frequency levels and fault constraints. On the bases of proposed algorithm, simulator has to find the appropriate execution frequencies for each real-time task that lead to the maximum energy savings for the given fault tolerance constraints. Power consumption of task set are simulator's output result.

TABLE I
Tasks set from Generic Avionics Platform

τ_i	p_i	C_i (ms)	T_i=D_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

We perform simulations with a number of synthesized real-time tasks sets and few real-world applications. The characteristics of on of the 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 CPU frequency levels we used data from [6] based on the published data of Intel Xscale PXA270. The data sheet of this processor is available online at its manufactures' websites. We used specifications listed in Table II.

TABLE II
Frequency and voltage levels of Intel Xscale PXA270

Frequency (MHz)	**Voltage** (V)	**Active power consumption** (mW)
624	1.55	925
520	1.45	747
416	1.35	570
312	1.25	390
208	1.15	279
104	0.9	116
13	0.85	44.2

Fig. 4 shows the simulation results for GAP task set and Intel Xscale PXA270 processor. The x-axis represents the ratio of T_{Fmin} to T_F. T_{Fmin} 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. 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.

The simulation was done for three possible scenarios connected with processor. In the first case we used all 7 voltage levels, in the second 4 (0.85V, 1.15V, 1.35V, 1.55V) and in the third just 2 voltage levels (1.15V, 1.55V).

All three scenarios indicate the same fact that power reduction leads to less fault tolerance and vice versa. It can be concluded that power reduction is better when more voltage levels are included. 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 for the given task set are between 16% and 20%. It can be seen, from the Fig. 4, that 7 voltage levels processor fulfill the power reduction demands. Also, fault tolerances vary for the given power reduction interval, so the best is to choose one with maximal tolerances.

Realized simulator offers the possibility to analyze one real-time task set when the main question is to find compromise between power or energy consumption and fault tolerance constraints. Our opinion is that this simulator could be successfully used in the RTS design proces

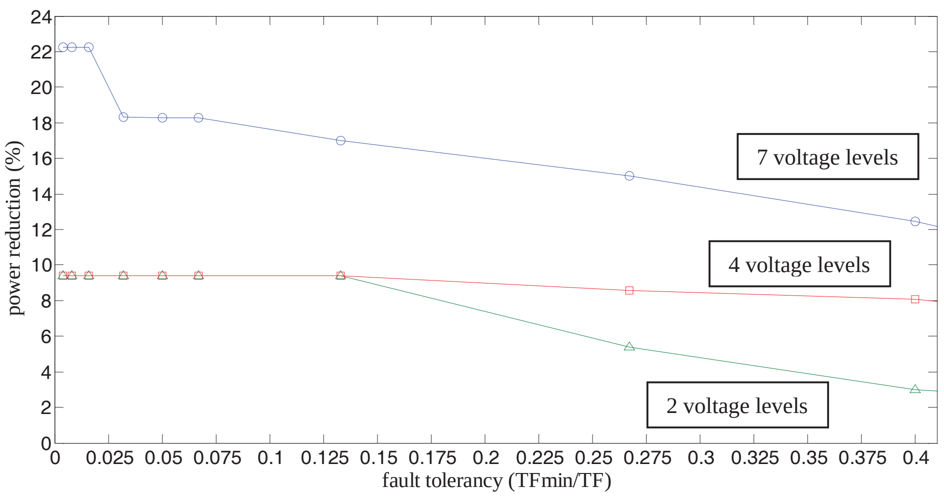

Fig. 4 The simulation results

V. Conclusion

This paper studies the trade-off between energy efficiency and fault tolerance for real-time task sets. Recognizing that the problem in discrete systems is NP-hard, we proposed heuristic-based approach which minimizes energy of task set for the given fault tolerant constraints. Our approach is realized for HRTS analysis when is necessary to examine connection between energy consumption and fault tolerance through time redundancy.

We considered only dynamic power in this paper. It should be noted that during the past decades, transistor sizes entered deep submicron regimes where static power consumption is now a non negligible source of power dissipation even in running mode. Thus, the total power consumption (i.e. dynamic plus static power) has to be optimized instead of simply reducing dynamic power.

Acknowledgement

This paper is supported by Project Grant III44004 (2011-2014) financed by Ministry of Education and Science, Republic of Serbia.

References

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

[2] Woonseok, K., Dongkun, S., Han-Saem, Y., Jihong, K., Sang, M. L., "*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.

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

[4] Zhu, P., Yang, F., Tu, G., Luo, W.,"*Fault-Tolerant Scheduling for Periodic Tasks based on DVFS*", Proceedings of the 9th International Conference for Young Computer Scientists, pp. 2186 – 2191, 2008.

[5] Santos, R. M., Santos, J., Orozco, J. D., "*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] Cottet, F., Delacroix, J., Mammeri, Z., "*Scheduling in Real-Time Systems*", John Wiley & Sons, 2002.

[8] Dakai, Z., Melhem, R., Mosse, D., "*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.

[9] Đošić, S., Jevtić, M., "*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.

[10] Lima, G., Burns, A., "*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.

[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.