Microelectronics Reliability 53 (2013) 1036–1042

Contents lists available at SciVerse ScienceDirect

Microelectronics Reliability

journal homepage: www.elsevier.com/locate/microrel

Dynamic voltage and frequency scaling algorithm for fault-tolerant real-time systems

Sandra Djosic *, Milun Jevtic

Faculty of Electronic Engineering, University of Nis, Aleksandra Medvedeva 14, 18000 Nis, Serbia

ARTICLE INFO

Article history:
Received 25 October 2012
Received in revised form 26 March 2013
Accepted 26 March 2013
Available online 16 April 2013

ABSTRACT

Many modern real-time systems (RTSs) are required to provide both fault tolerance and energy-efficiency in addition to their main objective to compute and deliver correct results within a specified period of time. Dynamic voltage and frequency scaling (DVFS) technique is known as one of the most effective low-energy technique for RTSs. However, most existing DVFS techniques only focus on minimizing energy consumption without taking the fault-tolerant capability of RTS into account. To solve this problem, in this paper we developed a new heuristic-based fault-tolerant dynamic voltage and frequency scaling (FT-DVFS) algorithm. The goal of the proposed algorithm is to find frequencies at which each task should be executed such that the energy consumed by the set of task is minimized. Beside energy minimization FT-DVFS algorithm has to meet all real-time requirements of individual tasks and to keep the system's ability to tolerate transient faults via task re-execution. The simulation results show that the proposed approach could save a significant amount of energy while preserving the required level of system's fault-tolerance capability when compared with the solutions obtained without energy-minimization.

1. Introduction

A system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed. Many RTSs are required to have continuous reliability so they are usually realized with the ability of tolerating some faults. Fault-tolerant real-time systems (FT RTSs) have to ensure that faults in the system do not lead to a failure. Nowadays, transient faults represent one of the most significant issues in the design of FT RTSs [1]. These faults appear only once and then disappear so we cannot trace them later on. Transient faults occur when an event (e.g., cosmic particle strikes, power supply noise, device coupling) causes the deposit or removal of enough charge to invert the state of a transistor. The inverted value may propagate to cause an error in program execution [2]. CMOS downscaling trends, manifested in the use of smaller transistor feature sizes and lower supply voltages, make digital circuits more and more vulnerable to transient errors with each new technology generation [3].

Fault-tolerance schemes generally rely on some form of redundancy [4]. We are particularly interested in the time redundancy techniques, since they are cost-effective as well as more suitable to applications where there are severe constraints on space and weight. This technique is specially used for overcoming the transient faults [5,6]. Since, among all types of faults the transient faults are much more common [7] we found the problem of their overcoming in RTSs using time redundancy important. In the context of RTSs the most common approach of time redundancy is the re-execution of a faulty task (i.e., running the task affected by a fault again) [8–10]. This is relatively inexpensive method of providing fault tolerance since no extra hardware is required.

Response Time Analysis (RTA) [11] is an important technique for analyzing timing constraints of RTSs. In its various forms it allows an exact calculation of the worst-case response time of tasks in single and multiprocessor RTSs scheduled under fixed priorities. Originally, RTA has been used for analyzing RTSs with the assumption that there is no error during system execution. This fault-free assumption is, in fact, not realistic. Quoting Laprie: ‘‘Non-faulty systems hardly exist, there are only systems which may have not yet failed’’. Burns et al. extended classical RTA to cope with the possibility of errors in the RTSs [12]. Their RTA method assumes the use of static free slack-time in the system schedule for performing re-execution of the faulty task in order to obtain reliable RTS. In [13] we used RTA for analyzing timing constraints of one fault tolerant hard real-time system with time redundancy. Modified version RTA, with the aim of getting more reliable RTS, was presented on our paper [14]. RTA was focus of some recent researches [15–17]. Altmeyer et al. integrates pre-emption costs into RTA for fixed priority pre-emptive scheduling [15]. They showed that this combined approach offers a significant improve-

* Corresponding author. Tel.: +381 18 529 601; fax: +381 18 588 399.
E-mail address: sandra.djosic@elfak.ni.ac.rs (S. Djosic).

0026-2714/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.microrel.2013.03.012

ment in performance for a wide range of different task and cache configurations. In [16] Mubeen et al. implemented and integrated modified real-time analysis technique with an existing industrial tool suite for the development of distributed real-time embedded systems. Yun et al. used RTA in multiprocessor RTSs [17]. They extended basic RTA by taking into account the task stalling caused by contention from interfering cores in multiprocessor systems.

In addition to high reliability, demand for low power consumption has risen sharply, particularly in RTSs where energy is limited (e.g., battery-powered systems). Decreasing the energy consumption not only extends the battery life, but also reduces the packaging and cooling costs of the systems. Since energy consumption of the CPU often dominates to the energy consumption of the entire RTS our attention is focused on the energy efficiency of the processor. Dynamic voltage and frequency scaling (DVFS) technique has proven to be a highly effective method of achieving CPU's low power consumption while meeting the performance requirements [18–20]. A number of modern microprocessors such as AMD's Mobile Athlon [21], Intel's XScale [22] and Transmeta's Crusoe [23] are equipped with the DVFS functionality. The key idea behind DVFS technique is to exploit the quadratic relationship between CPU energy consumption and supply voltage. By lowering the supply voltage and correspondingly the clock frequency simultaneously, the energy consumption of processors can be reduced quadratically [24]. However, the energy saving achieved by DVFS comes at the cost of extended execution time of real-time tasks, which may compromise the timing correctness of RTS. For example in [19] execution time of a real-time task scales linearly with the processing speed i.e., if the operating frequency is scaled the by a factor α, then execution time must be scaled by factor $1/\alpha$. Several DVFS techniques have been proposed to minimize the total energy in the RTS [18,25–28]. These techniques differ in many aspects, such as the scheduling algorithms being on-line or off-line, handling hard or soft deadline requirements, and assuming fixed or dynamic priority assignment. In [18] Kim et al. compare several key DVFS algorithms proposed for hard real-time periodic task sets, analyze their energy efficiency, and discuss the performance differences quantitatively. Pillai and Shin in [29] proposed the ccRM algorithm, which first computes off-line the maximum speed necessary to meet all task deadlines. On-line, the processor speed is scaled down when task complete early. In [28] Gruian presents a method of off-line task stretching coupled with on-line slack distribution. In addition, the paper presents a voltage scheduling method that computes the optimal speed for each execution cycle of the active task. Chen et al. control CPU utilization in distributed RTSs using DVFS technique [30]. Their method is based on adjusting the execution time of the tasks by scaling the processor frequency. In [31] He and Mueller presents energy efficient scheduling in component oriented hard real-time system using DVFS techniques. Kumari and Kumar propose a real-time scheduling algorithm with variable task's speed assignment scheme aims to reduce the energy consumption of the task set [32]. They use DVFS to control operating speed.

While time redundancy techniques use free slack time to improve reliability, DVFS exploits slack time to decrease energy consumption. Because the free slack time is a limited resource, it is obvious that more slack time used for DVFS leaves less time slack for fault tolerance, and vice versa. Therefore, there is a tradeoff between low energy consumption and high fault tolerance. In the context of real-time systems this tradeoff is analyzed in several papers [33–35]. Qadi and Goddard [33] present a DVFS algorithm supporting the canonical sporadic real-time task model. The method, however, is designed only for the EDF (earliest deadline first) priority discipline and it is not applicable to RM (rate monotonic) [36]. Melham et al. proposed techniques to exploit free slacks in task schedules to reduce energy consumption while tolerating faults based on DVFS [35]. They make several simplifying assumptions such as a task is susceptible to at most one fault occurrence and the processor can scale its frequency in continuous range. Nevertheless, practical commercial processors only support several discrete levels of supply voltages and frequencies [21–23].

In this paper, we propose a heuristic-based FT-DVFS (fault-tolerant – DVFS) algorithm for RTSs that attempts to minimize the energy consumed by a real-time task sets under fault tolerance constraints while guaranteeing that each task can complete successfully before its deadline. Our proposed FT-DVFS algorithm is applicable to an important class of RTS modeled as a set of fixed-priority preemptive periodic tasks, with different periods. Several discrete supply voltages and operating frequency levels are taken into consideration. Compensating for the negative effect on reliability, we adopt the faulty task re-execution fault-tolerant technique. The main novelty of our approach is the use of RTA to check the schedulability of fault-tolerant real-time task sets while varying voltage/frequency levels during DVFS power-optimization. To the best of our knowledge, this is the first time that RTA is integrated into a DVFS technique.

The remainder of the paper proceeds as follows. Section 2 introduce the system model and problem statement. The proposed FT-DVFS algorithm is presented in Section 3. Section 4 presents simulation results to validate our work. Conclusions are included in Section 5.

2. Modeling and problem statement

For the FT-DVFS problem, we are given n real time tasks. Each task has a given period, worst case execution time, deadline and priority. Also, a RTS can switch the frequency of its processor by dynamically adapting its voltage level. Hence, we have a set of active operating levels with frequencies and corresponding powers. Furthermore, it is expected that the RTS attains a given level of fault-tolerant capability, which is expressed via a fault-tolerant constraint (will be defined later in this section). The goal of our FT-DVFS algorithm is to assign each task an operating frequency so that the total power consumption is minimized while keeping the real-time task set schedulable under fault-tolerant constraint.

In this section, we first describe models that cover three different aspects of the FT-DVFS problem: real-time, power consumption and fault-tolerance. Then, we formally specify the FT-DVFS problem.

2.1. Task model

We modeled RTS as a set of n periodic real-time tasks, $\Gamma = \{\tau_1, \ldots, \tau_n\}$ where each task τ_i is associated with period T_i, worst case execution time (WCET) C_i and deadline D_i. We also assumed that all tasks start at the time 0 and have deadline D_i equal to period T_i. Each task is assigned a unique fixed priority p_i. Algorithm for scheduling real-time tasks could be any priority assignment algorithm [36]. We adopted a fully preemptive and independent task execution model. We assume a scenario where a single CPU executes a set of real-time tasks. The voltage and consequently the operating frequency of the CPU can be switched among m discrete values. We denoted voltage levels with U_j $(j = 1, \ldots, m)$, where $U_j < U_{j+1}$ and U_m is the maximum voltage and frequency levels with f_j $(j = 1, \ldots, m)$, where $f_j < f_{j+1}$ and f_m is the maximum frequency.

Given a set Γ of n tasks and a set of m frequencies, there are m^n different mappings of the set of frequencies onto the set of tasks. Each mapping can be represented by the so called *frequency assignment vector* $F = (x_1, x_2, \ldots, x_n)$, where $x_i \in \{1, \ldots, m\}$ is the index of the frequency assigned to task τ_i.

We adopt assumption that the WCET of a task scales linearly with the processing speed [19]. Therefore, if we scale the operating frequency by a factor α, then WCET is scaled by factor $1/\alpha$.