Microprocessors and Microsystems 45 (2016) 208–215

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Reliability-oriented scheduling for static-priority real-time tasks in standby-sparing systems

Vahidreza Moghaddas, Mahdi Fazeli*, Ahmad Patooghy

School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 14 December 2015
Revised 10 March 2016
Accepted 16 May 2016
Available online 17 May 2016

Keywords:
Energy efficiency
Fault-tolerance
Real-time embedded systems
Scheduling

ABSTRACT

The advent of complicated embedded systems with regard to relentless technology scaling and integration of more components into a single chip, have caused these systems to be less reliable. Moreover, these advancements have accompanied with a drastic increase in energy consumption. However, using energy management techniques such as Dynamic Voltage and Frequency Scaling (DVFS) has made reliability issues to be exacerbated. Hence, energy efficiency and fault-tolerance are two conflicting key objectives in the design of efficient real-time embedded systems. In this paper by considering periodic real-time tasks in standby-sparing system, we propose a novel online scheduling technique, which tackles this problem. The aim of this paper is twofold: (1) to show that scheduling primary and backup tasks in different processors in a moderate speed for static-priority real-time tasks will lead to better energy savings; (2) to explore the effectiveness of this technique for mixed-criticality tasks. Our simulation results reveal a significant energy saving (up to 25%) in comparison with the state-of-the-art scheme based on standby-sparing system, while preserving system's original reliability.

1. Introduction

In recent years, embedded systems have become a pervasive part of our daily lives, which hardly can find some areas in which they do not exist. For instance, embedded systems can be found easily in abundance in automobiles, cellphones, trains, airplanes and even in household appliances. The demanding trend in using embedded systems will grow by a triple rise from 2015 to 2020 [1]. In the past two decades, energy-aware design has become a hot topic in research areas and industry environments. Dynamic Voltage and Frequency Scaling (DVFS) is one of the most well-known and effective system-level methods to save energy in embedded systems [2]. DVFS dynamically and simultaneously adjusts supply voltage and frequency of a system proportional to its load so as to reduce the power consumption [3]. In fact, this would be obtained at the expense of increasing the response time, which in real-time systems can cause deadline violation and be problematic. Thus, many studies have been done toward reducing energy consumption while guaranteeing the timing constraints in DVFS-enabled systems so far [4-6].

In addition to the energy issues, continuous technology scaling and transistor miniaturization has led the reliability to be a major concern in the design of embedded systems [7]. Technology down-scaling increases the rate of radiation-induced faults, i.e. soft errors by tens of times in semiconductor devices [8]. Moreover, the rising power density in ICs creates serious threats and heat challenges, as a 10° increase in operating temperature of electrical components resulting in the device lifetime to be halved [9].

Even though failure in operational times of a device is more because of transient faults rather than permanent ones (with a ratio 100:1 or higher [10]), recovering from a permanent fault must be considered in a comprehensive fault-tolerant scheme. Furthermore, using energy management techniques like DVFS has an exponential negative effect on system reliability [11].

Time-redundancy based approaches for fault-tolerance purposes are of little utility in hard real-time systems, as recovery process within the deadline must be guaranteed [12]. Safety critical systems, which have hard deadlines, need a high level of fault-tolerance, since a deadline miss would cause a catastrophe for the entire system. In fact, these systems use hardware redundancy techniques such as: Triple Modular Redundancy (TMR), duplication, replication [13–15] for fault-tolerance, which consume a considerable energy. Meanwhile, these systems are embedded in harsh environments, so low energy consumption is just as important as having a high reliability and thus this issue needs to be addressed too. On the other hand, these two goals are in contrast with timing constraints. Therefore, there is a trade-off between these objectives.

* Corresponding author.
E-mail addresses: moghaddas@comp.iust.ac.ir (V. Moghaddas), m_fazeli@iust.ac.ir (M. Fazeli), patooghy@iust.ac.ir (A. Patooghy).

http://dx.doi.org/10.1016/j.micpro.2016.05.005
0141-9331/

In the context of jointly considering energy and reliability, [16] was the first study that addressed this issue in standby-sparing system. Primary tasks are executed on primary processor using DVFS energy management technique, while backup tasks are remained reserved in the spare processor to take over when primary task fails, using Dynamic Power Management (DPM) strategy [16]. However, there is no special scheduling strategy in [16] and this work is limited to aperiodic non-preemptive tasks. In [17] the co-management of reliability and energy is considered analytically for a set of frame-based tasks; nevertheless, like [16], it is assumed that a static schedule already exists. In [18], Haque *et al.* proposed a dynamic scheduling technique for dynamic priority workloads in a standby-sparing system, in which primary and backup tasks are scheduled according to Earliest Deadline First (EDF) and Earliest Deadline Late (EDL) respectively. In fact, this approach tries to minimize the overlapping execution of a primary and its corresponding backup task by postponing the backup task as late as possible and executing the primary one as early as possible. For static-priority real-time periodic tasks, the research works in [19, 20] also considered the trade-off between energy and reliability in a standby-sparing system. By using dual-priority scheduling mechanism [21], backup tasks are delayed as much as possible, whereas primary tasks are executed as soon as possible. The difference between [19] and [20] is in the calculation of upper bound on the response time, where in the former one a conservative technique [4] is used, while in the latter one the worst-case response time [22] is deployed. To better utilize the slack times on the spare processor, Guo et al. [23] developed a Preference-Oriented Earliest Deadline (POED) scheduling algorithm such that processors are treated equally and each processing unit will have a mixed combination of primary and backup tasks.

Different from all previous works in standby-sparing system, in this paper, by using concurrency and redundancy approach [13], we present a reliability oriented energy-efficient scheduling method for static-priority workloads that employs both DVFS and DPM for both primary and backup tasks. The proposed scheme with the help of dual priority scheduling [21] is able to delay backup tasks as much as possible, so upon successful execution of a task, its backup is canceled. Furthermore, our scheme is capable of handling tasks with different reliability goals in such a way that high critical tasks are not sacrificed for low importance ones.

The remainder of this paper is organized as follows. In Section 2, we present preliminaries, system models and assumptions along with the problem statement. In Section 3, we address the trade-off between processor speed, energy and reliability. In Section 4, our standby-sparing reliability-oriented scheme is investigated. Next, Section 5 briefly discusses mixed-criticality tasks. In Section 6, simulation results are presented. Finally, Section 7 concludes this paper.

2. System models and problem description

2.1. Task model

In this paper, we consider periodic tasks as this type of task model is the most prevalent task model in papers and in real world [12]. An example of a periodic task T_i is depicted in Fig. 1. A set of n independent periodic hard real-time tasks $\psi = \{T_1, T_2, ..., T_n\}$ has to be executed on 2 identical processors. We call tasks in ψ as *primary* tasks. Each task is described by (r_i, c_i, p_i), where task T_i has the period p_i and the Worst-Case Execution Time (WCET) c_i under the maximum processor speed. Tasks are assumed to have implicit deadlines, i.e. deadlines are equal to the periods ($d_i = p_i$). Each task instance (job) of T_i denoted as T_{ij} releases at $r_{ij} = (j\text{-}1).p_i$ and has the deadline $d_{ij} = j.p_i$. The *utilization* of task T_i namely u_i is

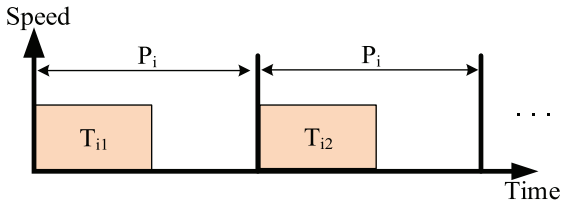

Fig. 1. A periodic task T_i with period P_i.

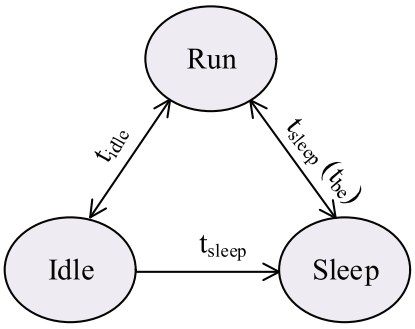

Fig. 2. Power states of a typical processor.

defined as $u_i = c_i / p_i$. The *system utilization* U_{sys} is the sum of the utilizations in ψ.

Processors are DVFS-enabled and capable of executing tasks in l discrete speeds $s_{min} = s_1 < s_2 < ... < s_l = s_{max}$. Since the operating frequency of a CMOS circuit is almost linearly proportional to the supply voltage [24], henceforth, we use parameter s which means voltage and frequency together. Moreover, speeds are normalized to the maximum speed s_{max}, i.e. $s_{max} = 1$.

For fault-tolerance goals in standby-sparing systems, tasks have a backup copy in spare processor with the same timing parameters as the primary task. We denote B_{ij} as the backup of the j-th job of T_i.

2.2. Power and energy model

We model the processor as a Finite-State Machine (FSM) with three different operational power modes as shown in Fig. 2, namely: *Run mode, Idle mode*, and *Sleep mode*. This assumption is also used in [19,20]. However, according to the target platform, these power states can be more or less than three modes, e.g. in [25], there are six power modes.

Tasks are executed in *run* mode and when there is no ready task to execute, the processor transitions to *idle* or *sleep* state. In each mode processor consumes different energy and power consumption being described below:

Run mode: In this mode, we adopt a system-wide power model, where the power of a system operating at supply voltage V_{dd} and frequency f can be expressed as [26–28]:

$$P_{Run} = P_S + P_{ind} + C_{ef} {V_{dd}}^2 f \quad (1)$$

Where C_{ef} is the effective switching capacitance, P_s is static power dominated by the leakage current. Due to the prohibitive overhead of halting the whole system (tens of seconds [26]) in which the static power will be omitted, we suppose the system is on and P_s is always consumed. P_{ind}, the speed-independent active power corresponds to the power components that their speeds cannot be modified like main memory or I/O devices. We can rewrite (1) with parameter s as:

$$P_{Run} = P_S + P_{ind} + C_{ef} s^{\alpha} \quad (2)$$

α is a system-dependent constant which models the relation between voltage and frequency ($2 \le \alpha \le 3$).