Neurocomputing 150 (2015) 82–89

Contents lists available at ScienceDirect

Neurocomputing

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

Stochastic vs. deterministic evolutionary algorithm-based allocation and scheduling for XMOS chips

Zorana Banković [a,*], Pedro López-García [a,b]

[a] IMDEA Software Institute, Madrid, Spain
[b] Spanish Council for Scientific Research (CSIC), Spain

ARTICLE INFO

Article history:
Received 18 November 2013
Received in revised form
9 April 2014
Accepted 15 June 2014
Available online 2 October 2014

Keywords:
Deterministic and stochastic scheduling
Multiprocessor multithreaded architectures
Dynamic voltage and frequency scaling
Evolutionary algorithms
Multi-objective optimization

ABSTRACT

We present an approach based on multi-objective evolutionary algorithms for the automatic scheduling and allocation of tasks in a multiprocessor multithreaded architecture, together with an assignment of the appropriate voltage and frequency of each processor in a way the overall energy consumed by the execution of the tasks is optimized and all task deadlines are met. We have implemented both a deterministic scheduling algorithm, where the execution time and the energy consumption of different tasks have a known deterministic value, and a stochastic scheduling algorithm, where the execution time and energy are treated as random variables with corresponding probability density functions, given that in reality these values can vary significantly due to numerous reasons. It is assumed that execution time and energy consumption estimations, both for the deterministic and the stochastic case, are obtained by a static analysis process. It has already been proven for the case of makespan optimization that the stochastic scheduling is underestimated by its deterministic counterpart, and that in many real world situations, the stochastic scheduler outperforms the deterministic one. In this work we prove that for the tested scenario the stochastic scheduler for energy optimization outperforms its deterministic counterpart improving energy consumption by 15.4% in the best case.

1. Introduction

The most common approach for solving the problem of optimal task scheduling is to use a safe estimation of the values that the function to be optimized depends on, such as the execution time of each task or its power consumption. When such estimation gives a single (numeric) value for execution time (and power/energy), we refer to the problem as the *deterministic scheduling*. However, the execution time of a task in reality can vary considerably, due to a number of reasons, e.g. unknown memory access time, operating system effects that cannot be known in advance, etc. For this reason, it is more accurate to treat execution time, as well as energy consumption, which is closely related, as a random variable with a corresponding probability density and/or cumulative distribution function. We refer to this group of problems as *stochastic scheduling* problems. Moreover, there are state-of-the-art results of optimal scheduling for makespan optimization [1,2] that prove that in certain situations the deterministic scheduler provides results that significantly deviate from the optimal ones, and that better results can be obtained using stochastic scheduling. In this work we prove that this is also the case for energy consumption optimization.

Our objective is to optimize the energy consumption through scheduling and allocation of a set of tasks running on multi-processor/multicore[1] and multithreaded voltage and frequency scalable architectures designed by XMOS [3]. In such XMOS chips, threads are pipelined in four stages, where in each stage of the pipeline one instruction from a different thread is executed, so in essence we can say that the threads also run in parallel. Thus, we deal here with two levels of parallelism: multicore with multiple threads on each of them. Furthermore, the XMOS chips have the possibility to dynamically scale voltage and frequency, which can significantly contribute to energy consumption optimization, as we will explain in the following.

The dynamic power consumption due to the switching activity in digital CMOS circuits can be expressed with the following formula:

$$P = \alpha \cdot C_{eff} \cdot V^2 \cdot f$$

where C_{eff} is the effective capacitance, V the voltage supply, f the operating frequency, and α the switching factor. If we decrease the

* Corresponding author.
E-mail addresses: zorana.bankovic@imdea.org (Z. Banković), pedro.lopez@imdea.org (P. López-García).

[1] These two terms are interchangeable throughout this paper.

http://dx.doi.org/10.1016/j.neucom.2014.06.077
0925-2312/

voltage supply and the operating frequency, the dynamic power will decrease significantly. Moreover, the static power, which is the result of the leakage currents, also decreases quadratically with the voltage [4]. Thus, voltage and frequency decrease can achieve significant power and energy savings. This optimization technique is known as Dynamic Voltage and Frequency Scaling (DVFS). However, voltage and frequency decrease slows down the operation of the circuit, and has to be applied in a way the required timing deadlines of all the tasks are still met. Furthermore, the process of scaling voltage and/or frequency introduces additional latencies, which implies that we have to develop a set of requirements that define the applicability of this approach.

Regarding the application of DVFS to the XMOS chips, we assume that different processors/cores can have different (V, f) settings, while the threads running on the same processor/core at the same time must have the same (V, f) setting. We also assume that the wake-up time is big enough in relation to the execution times of the tasks to avoid the shutdown of separate processors. However, as we will see, our approach can be easily extended to take into account the possibility of shutdown.

Given a set of tasks and their corresponding deadlines, our objective is to provide a scheduling and allocation, and also assign voltage and frequency to each processor in a way the total energy consumption is optimized, while meeting the task deadlines. We assume that the tasks are heterogeneous, and they in general have different starting times and deadlines. We further assume that there is no precedence between tasks, and no preemption. Regarding the estimation of the corresponding values for execution time and power/energy consumption, in the deterministic case we assume that there are available analysis tools that give us such estimations, which are a necessary input to the scheduling algorithms we present here. Since this work has been done in the context of the ENTRA project [5], where tools for the estimation at compile time of the energy consumption and execution time of programs are being developed, we have direct access to such tools. For example, we can use the energy analyzer of the CiaoPP tool [6] already developed. In addition we could also use any existing timing analysis tools, for which great amount of work have been devoted. On the other hand, an ongoing work in the ENTRA project [5] is dedicated to deriving probability functions of both execution time and energy consumption, as well as the interdependence between different variables that represent time and energy of different tasks, which is the necessary input to the stochastic scheduler.

In general, the problem of scheduling and allocation is NP-hard. For this reason, different heuristic algorithms have been developed that are capable of obtaining sub-optimal solutions in real time, such as [7], which is based on Artificial Bee Colony for makespan minimization and machine occupation maximization, or [8], which compares the performances of a hybrid genetic algorithm, a hybrid simulated annealing and particle swarm optimization for the flow shop scheduling problem in a manufacturing supply chain. Many of these heuristic approaches use evolutionary algorithms (EAs), e.g. [9] for the vehicle routing problem, and in particular genetic algorithms (GAs) [10–12]. An EA is a well-known bio-inspired approach based on the principle of the survival of the fittest. Its most important advantage is a fast exploration of the search space, which allows the quick finding of acceptable solutions. This is the reason our scheduler is based on EAs. Since DVFS reduces energy, but increases execution time, these two magnitudes are clearly in conflict. For this reason, we use a multi-objective optimization approach, in order to find a trade-off between energy consumption and execution time. We also provide an appropriate representation of solutions that captures the two levels of parallelism, i.e., at both processor and thread level, and at the same time performs allocation and scheduling and identifies appropriate (V, f) settings in real time, exploring in this way the entire search space. As far as we know, the work presented in this paper is the first solution to mentioned type of problems.

The rest of the paper is organized as follows. Section 2 presents the most relevant related work and emphasizes the most important advantages of our approach. Section 3 details the sources of power consumption in CPUs and sets up the constraints that are the basis for generating a solution. Section 4 explains the problem that is being solved and points out the main differences between deterministic and stochastic scheduling. Section 5 details the implemented approach, while Section 6 explains the experimentation environment and presents the most significant results. Finally, Section 7 draws the most important conclusions and gives some directions for future work.

2. Related work

The related work to the one presented here falls into a great variety of topics, yet the ones with the closest relation are energy-aware scheduling approaches using DVFS, in particular the ones based on EAs (GAs). In the following we present these techniques and emphasize the main advantages of our approach.

2.1. Energy-aware deterministic scheduling

Since DVFS can provide significant energy savings, its optimal usage has been extensively studied. Some examples divide scheduling and allocation in two separate steps, such as the one given in [13], where in the first step the allocation problem is solved using Linear Programming, while in the second one the scheduling problem is solved for separate processors using Bin Packing. Another approach [10] solves the scheduling problem using a GA that integrates DVFS in the fitness function. However, such a division of the problem reduces the search space, since it becomes limited by the optimal solution of the first part of the problem, which does not always correspond to the global optimum. For this reason, we believe that better solutions can be achieved by solving the scheduling and allocation problem at the same time, while also accounting for the DVFS. There is one example of GA-based scheduling [11] that combines scheduling, allocation and power management in one process. However, it only deals with voltage scaling.

There is also a significant group of publications on using GAs for the optimal scheduling and allocation in multiprocessor systems with the DVFS feature. For example, the approach presented in [12] aims to minimize both energy and makespan as a bi-objective problem. The same problem is solved in another work [14], but using the island model of parallel GA populations. Another approach [15] treats the problem from two opposite points of view: in the first one, it optimizes the energy given the scheduler length, while in the other one it optimizes the scheduling length given the energy bound. However, none of the solutions include the possibility of two levels of parallelism as in our work, where each processor can have a number of different threads executing in parallel.

2.2. Energy-aware stochastic scheduling

Stochastic scheduling has gained lots of interest over the years, since many different cases include uncertainty. In general, approaches to optimization under uncertainty include various modeling philosophies, the most important being the following ones:

- Expectation minimization.
- Minimization of deviations from goals.
- Minimization of maximum costs.
- Optimization over soft constraints.