A hybrid ILP-CP model for mapping Directed Acyclic Task Graphs to multicore architectures

Andreas Emeretlis, George Theodoridis
Dept. of Electrical & Computer Engineering
University of Patras,
Patras, Greece
{emeretlis, theodor}@ece.upatras.gr

Panayiotis Alefragis, Nikolaos Voros
Computer & Informatics Engineering Dept.
Technological Educational Institute of Western Greece,
Patras, Greece
{alefrag, voros}@teimes.gr

Abstract—**Directed Acyclic Task Graphs serve as typical kernel representation for embedded applications. Modern embedded multicore architectures raise new challenges for efficient mapping and scheduling of task DAGs providing a large number of heterogeneous resources. In this paper, a hybrid Integer Linear Programming – Constraint Programming method that uses the Benders decomposition is used to find proven optimal solutions. The proposed method is augmented with cuts generation schemes for accelerating the solution process. Experimental results show that the proposed method systematically outperforms an ILP-based solution method.**

Keywords — multicore architectures, DAG mapping, tasks scheduling, integer linear programming, constraint programming,

I. INTRODUCTION

In order to meet the increasing computational requirements of the current and future applications, modern computing platforms include a large number of heterogeneous cores. The heterogeneity allows addressing the different needs of the targeted application domain, while the large amount of cores allows parallel execution of applications. Nowadays, these platforms are the main used approach for satisfying the hard low-energy requirements. Moreover, they are robust to process variations and circuit failures which characterize the current and future deep submicron silicon technologies. One of the main challenges for this type of platforms is the development of methods for mapping complex real-life applications on them in a manner that exploits their features. In other words, proper methods for assigning the application's tasks to the cores and scheduling them per core are required aiming at optimizing one or more metrics (e.g. execution time, energy consumption etc.)

When the characteristics of the application (e.g. tasks, dependencies between the tasks, execution time of each task per core) are known in advance and do not change in run-time, static mapping methods can be developed. This class of problems includes a large number of real-life applications such as typical signal processing and multimedia algorithms, which are described by static dataflow graphs. Since the task mapping is performed once during compile time and remains unchanged during the product's lifetime, a reasonable design time for obtaining an optimal or a near-optimal solution is justified.

The above problem has been extensively studied in the past [1], [2] and the proposed methods can be classified in two main categories. The first one includes algorithms that are based on: i) heuristics, such as list scheduling [3], [4] or node duplication [5], [6], and ii) stochastic search algorithms, such as genetic algorithms [7], [8]. The advantage of these methods is their low computational complexity that allows them to provide a solution in reduced time. However, they suffer from three major drawbacks. Firstly, since they are based on heuristics, they cannot guarantee the optimality of the produced solution. Secondly, the produced solution may be far from the optimal one, when the assumptions on which these methods are based are not satisfied. Thirdly, they are not easily extensible and have to be redeveloped when new assumptions or constraints are imposed.

The second category includes methods that always return an optimal solution. In this case, the problem is modeled as an optimization problem using either Integer Linear Programming (ILP) [9]-[11] or Constraint Programming (CP) models [12], [13]. The main advantage of these methods is that they guarantee that the produced solution is always optimal. However, they suffer from large computational complexity and increased solution times, which prohibit their use in complex applications.

During the last decade, many research efforts have been performed in the fields of Operations Research and Artificial Intelligence for solving complex optimization problems with reduced solution times. The developed methods are based on hybrid ILP-CP models along with advanced algorithms achieving the solution of complex optimization problems with outstanding speedups [14], [15]. Yet, these methods have not been extensively applied in the field of multiprocessing systems. To the best of our knowledge, only one group adopted this approach for mapping static applications on multicore platforms and achieved remarkable speedups compared to the solution provided by a pure ILP or CP model [16], [17]. However, in these works, the assumed multicore platforms were homogeneous, which, compared to the heterogeneous case, results in a significantly less complex optimization problem. This is due to the fact that in