Journal of Systems Architecture 91 (2018) 1–10

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

Run-time mapping algorithm for dynamic workloads using association rule mining

Sima Sinaei, Omid Fatemi*

Electerical and Computer Engineering Department, University of Tehran, Tehran, Iran

A B S T R A C T

Task mapping exploration plays an important role in the high performance achieved by heterogeneous multi-processor system-on-chip (MPSoC) platforms. The dynamic of application workloads in modern MPSoC-based embedded systems are consistently growing. Nowadays, the execution of different applications is done concurrently, and these applications compete for resources in such systems. To cope with the dynamism of application workloads at runtime and improve the efficiency of the underlying system architecture, this paper presents a hybrid task mapping algorithm for multimedia applications. That consists of two phases: design-time and run-time. During design-time, static mapping exploration is performed, and the applications are clustered based on their efficient mapping, then a set of rules for mapping is extracted by Association Rule Mining techniques. During run-time, when a new application enters to the system, this application is classified to one of the existing clusters using the rule sets extracted at design-time phase. The objective of application mapping is to minimize execution time in a predefined budget of energy consumption. A heterogeneous MPSoC system is used to evaluate the proposed algorithm. The experimental results revealed that during run-time by using the proposed algorithm, suitable resources regarding energy consumption and execution time are selected for mapping.

1. Introduction

Modern embedded systems rely more and more on Multiprocessor System on Chip (MPSoC) architectures and often have to support an increasing number of applications and standards [1]. Typically, the target MPSoC architecture platforms are heterogeneous in nature because these systems are capable of providing better performance and energy tradeoffs than their homogeneous counterparts. In these systems, multiple applications can run concurrently and therefore these applications have to contend for system resources. It makes the role of efficiant task mapping more critical in order for the applications' diverse demands on the MPSoC architecture to be met. The task mapping criteria is optimization of energy consumption, computation performance, etc. [2–5].

Mapping of application tasks on MPSoC platform resources can be realized at either design-time (static) or run-time (dynamic) [6]. Design-time mapping techniques use a predefined set of applications with known computation and communication behaviors and a static platform. Therefore, they are not suitable for dynamic workloads in which new applications may appear (i.e., start their execution) on the platform at run-time and cause the number of running application competing for the available platform resources to vary over time. Thus, run-time (dynamic) mapping techniques are required for scenarios where application tasks need to be loaded into the platform at run-time.

The run-time mapping can be done either with or without previously analyzed results. Numerous studies have been carried out on mapping without previously analyzed results, i.e. on-the-fly processing [7–12]. In these methods, assignment of newly arriving tasks on the system resources is done by means of heuristics. The fact that these approaches must be light-weight in terms of processing power (as they are performed at run-time) may lead to lower quality mappings. A probable solution is transferring the compute intensive analysis to the design phase [13–20]. The methods that implement these solutions are called *hybrid* methods. Here, analyzed results that have been obtained at design-time can be used at run-time to accelerate efficient run-time mapping. Hybrid mapping approaches also facilitate a light-weight run-time manager, which is required in modern embedded systems (e.g., smart phones and tablets).

In this paper, a novel hybrid run-time mapping technique for multimedia MPSoC-based embedded systems is proposed. The proposed mapping technique minimizes workload execution time when a predefined budget of energy consumption is allotted. This technique facilitates the handling of new incoming applications not known at design-time. At design-time, a Design Space Exploration (DSE) algorithm is used to select the efficient mapping. As will be explained, this exploration is used to cluster applications. By using this supervised data set via Rule Association Mining, the best mapping rule set is selected. At run-time, when a new and possibly unknown application enters the system, a quick decision about resource selection and mapping has to be made. There isn't sufficient time to explore all or even a small fraction of the mapping possibilities.

A method based on Association Rule Mining (ARM) is proposed to solve the mapping problem. It will take considerably less time if ARM, a machine learning technique, is used to determine the resource demands of the incoming application. This is possible by clustering a set of predefined applications at design-time based on effective features.

* Corresponding author.
E-mail address: omid@fatemi.net (O. Fatemi).

https://doi.org/10.1016/j.sysarc.2018.09.005
Received 5 May 2018; Received in revised form 16 September 2018; Accepted 21 September 2018
Available online 22 September 2018
1383-7621/© 2018 Elsevier B.V. All rights reserved.

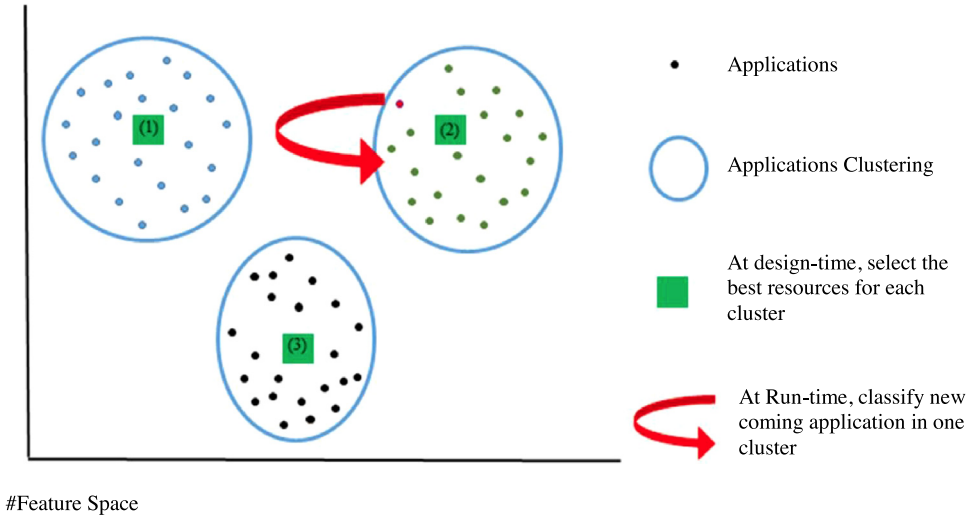

Fig. 1. Clustering at design-time and classification at run-time.

Each cluster is labeled based on the resource demands of the applications belonging to it. A new application is classified into one of the existing clusters by quick profiling of the new application, determination of its features, and using the mapping rule set in the system database. Subsequently, the label specified for that cluster will be used as the new application's resource demand. These steps are illustrated in Fig. 1 in which the contents are arranged in a two-dimensional space, and each point represents one of the applications in the features space.

The design space exploration method is used to determine efficient resource demands of each application in order to place them into clusters. In Fig. 1, application clusters are shown by the circles and the resource demands of each cluster are indicated by the squares. At run-time, application mapping can be quickly determined as indicated by the red arrow based on the effective features. The remainder of this paper is organized as follows: In Section 2, the related work is described. Section 3 describes prerequisites and defines the problem. Details of the proposed algorithms are presented in Section 4. The experimental results are presented in Section 5, after which Section 6 concludes the paper.

2. Related work

The design-time techniques for application mapping were used frequently in the last decade when the application workloads were less complicated and had static behavior. The efficient mapping of these applications was configured at design-time. As the technology progressed, the nature of application workloads became more complicated and dynamism became an inherent part of these workloads. The design-time techniques were not suitable for run-time varying workloads as the number of running application and their entrance time were not known beforehand. These workloads required re-mapping/run-time mapping of applications (e.g. networking and multimedia applications). Run-time mapping can be categorized into on-the-fly mapping and mapping using design-time DSE results. The latter is also called the hybrid method.

In on-the-fly mapping techniques [7–12] any changes or the entry of new application triggers the start of the decision making process. This process usually uses heuristic and greedy algorithms. These algorithms are improved by aiming to optimize specified performance metrics such as application execution time and power consumption. As these mapping techniques have to make decisions online as quickly as possible, the performance quality is often sacrificed; which leads to inadequate mapping.

Hybrid mapping implements the results of design-time mapping to find the starting point in distinguishing task properties and then optimize run-time decisions based on them. Hybrid mapping transfers the time intensive tasks to design-time and therefore can make decisions more accurately and in less time at run-time [13–20]. However, most of these approaches cannot adapt to application dynamism and handle newly incoming applications that were not known during design-time. In this paper, a novel hybrid algorithm for run-time mapping is proposed, which is capable of handling newly incoming applications by using a machine learning technique. The problem definition will be explained in the next section.

3. Problem definition

Application mapping is usually done at system-level design as low-level design requires too much time for evaluating several feasible mapping options. The inputs of system level mapping are typically the behavior of the application, architectural characteristics, and the various relations between them. To better explain the application mapping problem, it is necessary to first identify the application model, the architecture model, and the mapping between them as prerequisites.

3.1. Application model

In this paper, we target the multimedia application domain. For this reason, the Kahn Process Network (KPN) model of computation [21] is used to specify application behavior as it fits well to the streaming behavior of multimedia applications. In a KPN, an application is described as a network of concurrent processes that are interconnected via FIFO channels. An application can be represented as a directed graph $KPN = (P, F)$ where P is set of processes (tasks) p_i in the application and $f_{ij} \epsilon F$ represents the FIFO channel between two processes pi and pj .In Fig. 2, the KPN of a Motion-JPEG (MJPEG) application is shown as an example.

3.2. Architecture model

An architecture model is specified by the set of architectural elements and the connections that link those elements. In this model, the set of architectural elements comprises two distinct sets: the set of processing elements P and the set of memory elements M. An architecture can be modeled as a graph $MPSoC = (PE, C)$, where PE is the set of processing elements used in the architecture and C is a multiset of pairs

Download English Version:

https://daneshyari.com/en/article/11021081

Download Persian Version:

https://daneshyari.com/article/11021081

Daneshyari.com