Computers and Electrical Engineering xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Computers and Electrical Engineering

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

Online scheduling and placement of hardware tasks with multiple variants on dynamically reconfigurable field-programmable gate arrays ☆

Thomas Marconi *

School of Computing, National University of Singapore, Computing 1, 13 Computing Drive, Singapore 117417, Singapore
Computer Engineering Lab., TU Delft., Building 36, Room: HB10.320, Mekelweg 4, 2628 CD Delft, The Netherlands

ARTICLE INFO

Article history:
Available online xxxx

ABSTRACT

Hardware task scheduling and placement at runtime plays a crucial role in achieving better system performance by exploring dynamically reconfigurable Field-Programmable Gate Arrays (FPGAs). Although a number of online algorithms have been proposed in the literature, no strategy has been engaged in efficient usage of reconfigurable resources by orchestrating multiple hardware versions of tasks. By exploring this flexibility, on one hand, the algorithms can be potentially stronger in performance; however, on the other hand, they can suffer much more runtime overhead in selecting dynamically the best suitable variant on-the-fly based on its runtime conditions imposed by its runtime constraints. In this work, we propose a fast efficient online task scheduling and placement algorithm by incorporating multiple selectable hardware implementations for each hardware request; the selections reflect trade-offs between the required reconfigurable resources and the task runtime performance. Experimental studies conclusively reveal the superiority of the proposed algorithm in terms of not only scheduling and placement quality but also faster runtime decisions over rigid approaches.

1. Introduction

Nowadays, digital electronic systems are used in a growing number of real life applications. The most flexible and straight forward way to implement such a system is to use a processor that is programmable and can execute wide variety of applications. The hardware, a general-purpose processor (GPP) in this case, is usually fixed/hardwired, whereas the software ensures the computing system flexibility. Since such processors perform all workloads using the same fixed hardware, it is too complex to make the hardware design efficient for a wide range of applications. As a result, this approach cannot guarantee the best computational performance for all intended workloads. Designing hardware devices for a particular single application, referred as Application-Specific Integrated Circuits (ASICs), provides a system with the most efficient implementation for the given task, e.g., in terms of performance but often area and/or power. Since this requires time consuming and very costly design process along with expensive manufacturing processes, it is typically not feasible in both: economic costs and time-to-market. This solution, however, can become interesting when very high production volumes are targeted. Another option that allows highly flexible as well as relatively high performance computing systems is using reconfigurable devices,

☆ Reviews processed and approved for publication by Editor-in-Chief Dr. Manu Malek.

* Addresses: School of Computing, National University of Singapore, Computing 1, 13 Computing Drive, Singapore 117417, Singapore, Computer Engineering Lab. TU Delft, Building 36, Room HB10.320, Mekelweg 4, 2628 CD Delft, the Netherlands. Tel.: +65 6516 2727/+31 15 2786196; fax: +65 6779 4580/+31 15 2784898.

E-mail address: thomas_marconi@yahoo.com

0045-7906/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compeleceng.2013.07.004

such as Field-Programmable Gate Arrays (FPGAs). This approach aims at the design space between the full custom ASIC solution and the general-purpose processors. Often platforms of this type integrate reconfigurable fabric with general-purpose processors and sometimes dedicated hardwired blocks. Since such platforms can be used to build arbitrary hardware by changing the hardware configuration, they provide a flexible and at the same time relatively high performance solution by exploiting the inherent parallelism in hardware. Such systems where hardware can be changed at runtime are referred as runtime reconfigurable systems. A system involving partially reconfigurable devices can change some parts during runtime without interrupting the overall system operation [1,2].

Recently, dynamically reconfigurable FPGAs have been used in some interesting application areas, e.g., data mining [3], automobile [4], stencil computation [5], financial computation [6], Networks on Chip (NoC) [7], robotics [8], Software Defined Radio (SDR) [9], among many others. Exploiting partially reconfigurable devices for runtime reconfigurable systems can offer reduction in hardware area, power consumption, economic cost, bitstream size, and reconfiguration time in addition to performance improvements due to better resource customization (e.g. [1,2]). To make better use of these benefits, one important problem that needs to be addressed is hardware task scheduling and placement. Since hardware on-demand tasks are swapped in and out dynamically at runtime, the reconfigurable fabric can be fragmented. This can lead to undesirable situations where tasks cannot be allocated even if there would be sufficient free area available. As a result, the overall system performance will be penalized. Therefore, efficient hardware task scheduling and placement algorithms are very important.

Hardware task scheduling and placement algorithms can be divided into two main classes: *offline* and *online*. Offline assumes that all task properties (e.g., arrival times, task sizes, execution times, and reconfiguration times) are known in advance. The offline version can then perform various optimizations before the system is started. In general, the offline version has much longer time to optimize system performance compared to the online version. However, the offline approach is not applicable for systems where arriving tasks properties are unknown beforehand. In such general-purpose systems, even processing in a nondetermistic fashion of multitasking applications, the online version is the only possible alternative. In contrast to the offline approach, the online version needs to take decisions at runtime; as a result, the algorithm execution time contributes to the overall application latency. Therefore, the goal of an online scheduling and placement algorithm is not only to produce better scheduling and placement quality, they also have to minimize the runtime overhead. The online algorithms have to quickly find suitable hardware resources on the partially reconfigurable device for the arriving hardware tasks. In cases when there are no available resources for allocating the hardware task at its arrival time, the algorithms have to schedule the task for future execution.

Although a number of online hardware task scheduling and placement algorithms on partially reconfigurable devices have been proposed in the literature, no study has been done aimed for dynamically reconfigurable FPGA with multiple selectable hardware versions of tasks, paving the way for a more comprehensive exploration of its flexibility. Each hardware implementation for runtime systems with rigid algorithms, considering only one hardware implementation for handling each incoming request, is usually designed to perform well for the heaviest workload. However, in practice, the workload is not always at its peak. As a result, the best hardware implementation in terms of performance tends to use more resources (e.g. reconfigurable hardwares, electrical power, etc) than what is needed to meet its runtime constraints during its operation. Implementing a hardware task on FPGA can involve many trade-offs (e.g. required area, throughput, clock frequency, power consumption, energy consumption). These trade-offs can be done by controlling the degree of parallelism [10–15], using different architecture [16,17], and using different algorithms [18,19]. Given this flexibility, we can have multiple hardware implementations for executing a task on FPGA. To better fit tasks to reconfigurable device, since partially reconfiguration can be performed at runtime, we can select dynamically the best variant among all possible implemented hardwares on-the-fly based on its runtime conditions (e.g. availability of reconfigurable resources, power budget, etc) to meet its runtime requirements (e.g. deadline constraint, system response, etc) at runtime. Therefore, it is worthwhile to investigate and build algorithms for exploring this flexibility more, creating the nexus between the scheduling and placement on dynamically reconfigurable FPGAs and the trade-offs of implementing hardware tasks. In this work, we focus on online scheduling and placement since we strongly believe it represents a more generic situation, supporting execution of multiple applications concurrently. In a nutshell, we propose to incorporate multiple hardware versions for online hardware task scheduling and placement on dynamically reconfigurable FPGA for benefiting its flexibility in gaining better runtime performance with lower runtime overhead.

In this article, first of all, we introduce an idea to take into account multiple hardware versions of hardware tasks during scheduling and placement at runtime on dynamically reconfigurable FPGAs. On one hand, because of its variety of choices, in this case, the algorithms have more flexibility in choosing the right version for running each arriving task, gaining a better performance. On the other hand, the algorithms can suffer high runtime overhead since they not only need to find the right location and timing for running the incoming tasks but also they need to choose the best implementation among multiple hardware variants. Next, we do comprehensive computer simulations to show that the idea of multiple implementations of tasks significantly improves the performance of reconfigurable computing systems; however, it suffers high runtime overhead. Since we target a realistic online scenario where the properties of tasks are unknown at compile time, time taken by the algorithms are considered as an overhead to the application overall execution time; hence this issue needs to be solved properly. To handle this issue, we subsequently introduce a technique to amortize this high runtime overhead by reducing search space at runtime, referred as *pruning capability*. With this technique, the algorithm quickly determines all reconfigurable units that cannot be assigned to hardware tasks, avoiding exhaustively searching the space yet yielding the same runtime performance. We then apply this technique to build an efficient algorithm to tackle the scheduling and placement

Please cite this article in press as: Marconi T. Online scheduling and placement of hardware tasks with multiple variants on dynamically reconfigurable field-programmable gate arrays. Comput Electr Eng (2013), http://dx.doi.org/10.1016/j.compeleceng.2013.07.004