Expert Systems With Applications 45 (2016) 385–399

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Multi-objective kernel mapping and scheduling for morphable many-core architectures

Nuno Neves [a,c], Rui Neves [b,c], Nuno Horta [b,c], Pedro Tomás [a,c], Nuno Roma [a,c,*]

[a] INESC-ID, Rua Alves Redol, 9, 1000–029 Lisboa, Portugal
[b] Instituto de Telecomunicações, Av. Rovisco Pais, 1, 1049–001 Lisboa, Portugal
[c] Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais, 1, 1049–001 Lisboa, Portugal

ARTICLE INFO

Keywords:
Optimization
Design methodologies
Multi-core
Reconfigurable architectures
Run-time and dynamic reconfiguration
Energy efficiency

ABSTRACT

A new optimization framework to maximize the performance and efficiency of morphable many-core accelerators is proposed. The devised methodology supports the co-existence of multiple optimization goals and constraints (e.g., computational performance, power, energy consumption and runtime reconfiguration overhead) by relying on a design space exploration approach based on a convenient adaptation of a Multi-Objective Evolutionary Algorithm. In accordance, the proposed algorithm allows the generation of a comprehensive set of execution plans, specifically targeting an efficient runtime adaptation of the processing elements instantiated in morphable slots of the processing structure. The conducted experimental evaluation shows significant gains in terms of the attained performance and energy efficiency when considering both highly parallel and data dependent applications, achieving peak power dissipation and energy consumption reductions as high as 54% and 45%, respectively.

1. Introduction

The increasing demand for computational processing power that has been observed along the last decade has driven the development of heterogeneous systems, typically composed of a host General Purpose Processor (GPP) and one or more accelerating devices, such as Graphical Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASICs), each typically integrating multiple processing elements (PEs). While these systems already allow for significant application acceleration, it is of fundamental importance to improve the offered processing efficiency, while complying with the strict power and energy constraints of both embedded and high performance computing systems. In particular, although it has been widely recognized that energy consumption can lead to important constraints in the processing performance of mobile computing platforms, the observed divergence between device-level energy-efficiency gains and transistor-density (Esmaeilzadeh, Blem, St Amant, Sankaralingam, & Burger, 2011) may require that, in future computing systems, some of the transistors must remain dimmed or powered down most of the time. A solution for such a problem relies on the exploitation of runtime dynamic morphing of processor architectures, by clock/power gating unused hardware resources, by applying Dynamic Voltage and Frequency Scaling (DVFS) techniques or by relying on fine- or coarse-grained hardware reconfiguration (Petrica, Izraelevitz, Albonesi, & Shoemaker, 2013). In particular, by dynamically adapting the PEs architectures to the application kernels and by adopting a carefully management of the available processing resources, not only in terms of its execution, but also in terms of its energy requirements, it is possible to attain high computing performance and energy efficiency (Venkatesh et al., 2011).

However, while several programming frameworks have already been developed with the specific purpose of efficiently exploiting GPUs for general purpose computation (CUDA/OpenCL), morphable hardware accelerators have not received so much attention, specially those deployed on reconfigurable technology. In fact, although some existing frameworks (e.g. Xilinx Vivado, Altera SDK for OpenCL, Maxeler Technologies' MaxCompiler) already provide the means for translating kernels into low-level hardware implementations and for mapping them into FPGAs, the ability for runtime morphing the architectures is usually not taken into account. As a consequence, the development of a new framework capable of efficiently and dynamically mapping and scheduling an application's kernels into the PEs of a morphable accelerator is still highly required.

The herein proposed framework tackles the described problem by incorporating a design space exploration (DSE) tool to derive, in compile-time, a set of execution plans that describe not only an

* Corresponding author at: ECE Department, INESC-ID, IST, Universidade de Lisboa, Rua Alves Redol, 9, 1000–029 Lisboa, Portugal. Tel.: +351213100311.
E-mail addresses: Nuno.Neves@inesc-id.pt (N. Neves), Rui.Neves@lx.it.pt (R. Neves), Nuno.Horta@lx.it.pt (N. Horta), Pedro.Tomas@inesc-id.pt (P. Tomás), Nuno.Roma@inesc-id.pt (N. Roma).

http://dx.doi.org/10.1016/j.eswa.2015.10.004
0957-4174/

efficient mapping of tasks to PEs, but also the scheduling of those mappings to morphable processing hardware. Furthermore, by taking advantage of a multi-objective optimization (MOO) technique, it is possible to generate multiple execution plans, each establishing a different compromise between the application's performance, system power consumption and energy efficiency. Hence, by carefully selecting (in runtime) the most appropriate execution plan and by considering the time and energy overheads resulting from the communication and the reconfiguration/adaptation process, it is possible to make an application execution as adaptive and energy-efficient as possible.

To validate the proposed algorithm and scheduling methodologies, they were conveniently integrated into a previously developed morphable many-core accelerator, managed and controlled by an integrated Hypervisor module. The conducted experimental evaluation in a reconfigurable hardware scenario shows that the proposed multi-objective optimization approach provides significant energy savings under a set of both static and dynamic application workloads, reaching gains as high as 45%.

The remaining of the manuscript is organized as follows: Section 2 summarizes the background and the related work and enumerates the contributions of the herein proposed approach; Section 3 introduces a generic model of the considered processing platform composed by multiple reconfigurable/adaptable accelerators, together with the presentation of its underlying application specifications; Section 4 describes the proposed DSE algorithm, including the definition of the optimization goals and all of the intervening operators included in the adopted Objective Evolutionary Algorithm (MOEA); the algorithm is then validated in Section 5, by considering two case studies; finally, Section 6 concludes the manuscript by discussing and addressing the main contributions and achievements.

2. Related work

The new optimization framework that is herein proposed represents a considerable extension of other MOO design approaches that have been presented in the literature to optimize highly heterogeneous many-core processing systems. Some examples of related contributions comprise: code optimization (Balaprakash, Tiwari, & Wild, 2014; Neugebauer, Marwedel, & Engel, 2015); static and dynamic task scheduling (Behnamian, Zandieh, & Ghomi, 2009; Camelo, Donoso, & Castro, 2011; Cheng, Shiau, Huang, & Lin, 2009; Daoud & Kharma, 2011; Sheikh & Ahmad, 2013; Xu, Li, Hu, & Li, 2014); and a number of DSE approaches based on high-level and system-level synthesis (Erbas, Cerav-Erbas, & Pimentel, 2006; Gschwandtner, Durillo, & Fahringer, 2014; Holzer, Knerr, & Rupp, 2007; Krishnan & Katkoori, 2006), application mapping in multi-processor systems (Mariani et al., 2010; Palermo, Silvano, & Zaccaria, 2009) and heterogeneous systems (Erbas, Erbas, & Pimentel, 2003), as well as routing and communication topology optimizations (Glaß, Lukasiewycz, Wanka, Haubelt, & Teich, 2008). However, although some of these solutions already target morphable processing architectures, they do not take advantage of the runtime reconfiguration/adaptation capabilities of the underlying hardware support.

On the other hand, some DSE algorithms have been proposed to specifically target dynamically reconfigurable platforms. In Miramond and Delosme (2005), it is proposed a tool that defines different contexts for reconfigurable circuits, which are dynamically switched in runtime through partial reconfiguration. The underlying tasks are then assigned to each configuration through spatial and temporal partitioning, by using a local search algorithm. In Czarnecki and Deniziak (2008), it is proposed the usage of conditional task graphs, allowing to model mutually exclusive tasks. According to this algorithm, all tasks are initially assigned to only one GPP module and new solutions are produced using iterative improvement methods. Also based on conditional task graphs, the Evolutionary Algorithm (EA) proposed in Shang and Jha (2002) is used to determine the amount of used resources and to assign tasks to PEs, followed by a re-mapping and scheduling algorithm that makes use of the dynamic reconfiguration capabilities of FPGAs. Another algorithm targeting multi-mode systems is proposed in Wildermann, Reimann, Ziener, and Teich (2011), where it is assumed that the different operating modes can share the same hardware resources by means of partial reconfiguration. A symbolic encoding is also proposed, by combining a SAT solver and an MOEA, allowing the system synthesis for allocation, binding and placement of partially reconfigurable modules, both temporally and spatially.

Although the above referred frameworks and algorithms already take advantage of dynamic reconfiguration, they are mostly focused in HW/SW co-synthesis and HLS problems. In contrast, the herein proposed approach considers CPU-coupled morphable heterogeneous accelerators integrating a number of predefined reconfigurable/adaptable regions, allowing the dynamic morphing of their PE architectures according to energy and runtime execution requirements.

2.1. Background

A rather preliminary approach to the herein proposed optimization strategy was initially considered in the morphable many-core acceleration platform proposed in Neves, Mendes, Chaves, Tomás, and Roma (2015a). Such adaptive processing structures is managed by an Hypervisor module, implemented either in the host computer or locally in the accelerator device. The Hypervisor is responsible for permanently monitoring the accelerator's PEs execution in order to define convenient scheduling decisions (in real-time), and to issue appropriate reconfiguration commands that adapt the architecture of each individual morphable region to the instantaneous characteristics and requirements of the application under execution. Moreover, the devised Hypervisor module can also deploy runtime optimization policies to further promote the performance and energy efficiency, thus guaranteeing an extended battery life and/or minimum power consumption (e.g. in mobile computing platforms), or minimum energy costs in high performance computing clusters. Such devised optimization policies are based on intrinsic and/or external requirements that may demand the system to reduce the power consumption in runtime (e.g. by turning off processing cores) or to ensure a minimum and stationary performance level, while complying with strict peak power or energy constraints. Nevertheless, such an approach incurs in significant overheads that affect the overall efficiency of the platform, since the Hypervisor is required to perform a significant amount of operations in runtime, namely: (i) read the performance counters of the PEs; (ii) process the obtained information through the set of optimization policies; and (iii) schedule and trigger the appropriate reconfiguration commands to the accelerator.

2.2. Contributions

The optimization methodology that is now proposed complements the Hypervisor-based platform proposed in Neves et al. (2015a) with a comprehensive set of DSE optimization tools, executed at compile time. These tools provide a new abstraction level of the underlying morphable processing structures, particularly fitted to the application kernels being migrated from the CPU. With such an approach, the Hypervisor can be provided with further information about the application behavior and requirements, thus mitigating the implicit overheads, and even allowing the anticipation of the required reconfigurations and scheduling decisions, thus hiding the adaptation procedure behind the application execution.