ABSTRACT

Efficient runtime resource management in multi-processor systems-on-chip (MPSoCs) for achieving high performance and low energy consumption is one of the key challenges for system designers. OSIP, an operating system application-specific instruction-set processor, together with its well-defined programming model, provides a promising solution. It delivers high computational performance to deal with dynamic task scheduling and mapping. Being programmable, it can easily be adapted to different systems. However, the distributed computation among the different processing elements introduces complexity to the communication architecture, which tends to become the bottleneck of such systems. In this work, the authors highlight the vital importance of the communication architecture for OSIP-based systems and optimize the communication architecture. Furthermore, the effects of OSIP and the communication architecture are investigated jointly from the system point of view, based on a broad case study for a real life application (H.264) and a synthetic benchmark application.

DOI: 10.4018/978-1-4666-2776-5.ch009
INTRODUCTION

Heterogeneous multi-processor systems-on-chip (MPSoCs) are nowadays widely used in the embedded domain, such as wireless communication and multimedia, since they can provide efficient trade-offs between the computational power, the energy consumption and the flexibility of the system. One big challenge that comes with heterogeneous MPSoCs is system programming. This becomes even more critical, when taking runtime task scheduling and mapping into consideration. In large-scale systems, runtime scheduling and mapping are highly demanded from the performance and energy perspective, since it is very difficult to consider different dynamic effects at design time.

Various approaches have been proposed in academia and industry to address this problem. Generally, these approaches can be categorized into two groups: software solutions and hardware solutions. Software solutions such as TI OMAP (Texas Instruments, Inc., 2010) and Atmel D940 (Atmel Corporation, 2011) typically employ an operating system (OS), which runs on a RISC processor that dynamically distributes the workload to processing elements (PEs). These approaches are very flexible, but have low efficiency, especially when it comes to small tasks. The main reason is the high OS overhead in terms of power, memory footprint and performance.

This problem has been tackled by moving OS functionality to hardware accelerators, both for uni-core platforms (Nakano, Utama, Itabashi, Shiomi, & Imai, 1995; Kohout, Ganesh, & Jacob, 2003; Murtaza, Khan, Rafique, Bajwa, & Zaman, 2006; Nordström & Asplund, 2007) and for multi-core platforms (Park, Hong, & Chae, 2008; Seidel, 2006; Limberg et al., 2009; Lippett, 2004; Nácul, Regazzoni, & Lajolo, 2007; Pan & Wells, 2008). In the following, the hardware solutions for MPSoCs are further discussed.

The hardware OS kernel – HOSK introduced by Park, Hong, and Chae (2008) is a coprocessor that performs scheduling (fair and priority based) on a homogeneous cluster of simplified RISC processors. It features a low multi-threading overhead (less than 1% for 1-kcycle-tasks). However, a dedicated context controller should be included into the RISC processor to exchange context data between the processor and HOSK. This impedes its integration into traditional component-based design with off-the-shelf processors. Furthermore, to our best knowledge, no programming model exists for the HOSK-based MPSoCs.

In the work of Seidel (2006) and Limberg et al. (2009), a hardware scheduler called CoreManager is used to detect task dependency at runtime and schedule tasks. A programming model is provided along with CoreManager, following the synchronous data flow (SDF) model. Hence, this solution is only applied to a limited set of applications. High scheduling efficiency has been reported for CoreManager (60 cycles to schedule a task on average), which however is at the cost of high area overhead.

The approach of SystemWeaver (Lippett, 2004) focuses on the issue of task scheduling and mapping on heterogeneous MPSoCs, supported with a programming model. It has a slightly higher flexibility than HOSK and CoreManager by allowing the user to compose different basic scheduling primitives so as to implement complex scheduling decisions. However, its flexibility is still rather limited and the usage is rather difficult due to its design complexity.

In the SMP architecture introduced by Nácul, Regazzoni, and Lajolo (2007), a hardware RTOS is applied for scheduling a dual-ARM-system, based on the round-robin policy. In the system, for each ARM processor a hardware scheduler instance is required, which makes this approach difficult to scale to a large system without significantly increasing the area.

Hardware supported scheduling has also been considered in dynamically reconfigurable systems-on-chip (RSoCs) to improve the utilization of the reconfigurable components of the system. In the work presented by Pan and Wells (2008), a