Chapter 35
System–Level Analysis of MPSoCs with a Hardware Scheduler

Diandian Zhang
RWTH Aachen University, Germany

Gerd Ascheid
RWTH Aachen University, Germany

Jeronimo Castrillon
Dresden University of Technology, Germany

Rainer Leupers
RWTH Aachen University, Germany

Stefan Schürmans
RWTH Aachen University, Germany

Bart Vanthournout
Synopsys Inc., Belgium

ABSTRACT

Efficient runtime resource management in heterogeneous Multi-Processor Systems-on-Chip (MPSoCs) for achieving high performance and energy efficiency is one key challenge for system designers. In the past years, several IP blocks have been proposed that implement system-wide runtime task and resource management. As the processor count continues to increase, it is important to analyze the scalability of runtime managers at the system-level for different communication architectures. In this chapter, the authors analyze the scalability of an Application-Specific Instruction-Set Processor (ASIP) for runtime management called OSIP on two platform paradigms: shared and distributed memory. For the former, a generic bus is used as interconnect. For distributed memory, a Network-on-Chip (NoC) is used. The effects of OSIP and the communication architecture are jointly investigated from the system point of view, based on a broad case study with real applications (an H.264 video decoder and a digital receiver for wireless communications) and a synthetic benchmark application.

INTRODUCTION

Heterogeneous Multi-Processor Systems-on-Chip (MPSoCs) are nowadays widely used in the embedded domain, such as in wireless communication and multimedia applications, 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 demanding

from the performance and energy perspective, since it is very difficult to consider different dynamic effects at design time. At the same time, applications are becoming more dynamic in nature, making previous static scheduling approaches obsolete. For these reasons, even MPSoCs for deeply embedded applications employ some kind of runtime scheduling.

The past decade has seen a myriad of techniques for runtime task and resource management, from extensions and optimizations of traditional Operating Systems (OS) to dedicated hardwired solutions for heterogeneous MPSoCs. Today, even commercial platforms such as the Texas Instruments KeyStone II provide hardware support for task management (Biscondi et al. 2012). In such systems, from the software perspective, a programmer uses a high-level Application Programming Interface (API), which internally calls hardware primitives that enable efficient, system-wide application scheduling. In this chapter we analyze MPSoCs with this kind of support. As hardware scheduler, we analyze the so-called OS Instruction-set Processor (OSIP), a custom processor optimized for task management in heterogeneous MPSoCs (Castrillon et al., 2009). OSIP provides a balance between the efficiency of hardwired runtime managers and the flexibility of pure software solutions.

Typically runtime managers are analyzed and benchmarked either in isolation or within an entirely customized MPSoC. This makes it difficult to assess the overhead of the runtime manager when integrated in a different platform, or to understand how platform-wide design decisions affect the overall performance of an application running on the MPSoC. In this chapter we provide a thorough characterization of the performance of OSIP on different types of systems. Two main on-chip interconnect paradigms are covered in the analysis: Bus-based systems, e.g., AMBA buses (ARM Ltd., 2013) or CoreConnect (IBM Corporation, 2013), and larger-scale systems with a Network-on-Chip (NoC) as interconnect (Benini & De Micheli,

2002; Jantsch & Tenhunen, 2003). These two interconnect paradigms account for most of the systems designed today. For the analysis, typical architectural features of bus-based systems (e.g., cache subsystem) and NoC-based systems (e.g., peripherals for Direct Memory Access (DMA)) are considered in this chapter.

The joint characterization of OSIP and the different communication architectures is carried on by varying the number of Processing Elements (PEs) and analyzing the application performance as well as the runtime management overhead. The latter is compared to an off-the-shelf RISC processor and to an ideal manager (i.e., a manager that processes scheduling and mapping requests in zero time). For benchmarking, the H.264 video decoder and a Multiple-Input Multiple-Output (MIMO) digital receiver are used. Additionally, a synthetic benchmark complements the analysis by providing an idea of the limits of OSIP.

The rest of this chapter is organized as follows. After a brief survey of runtime managers, an introduction to OSIP and OSIP-based MPSoCs is given. Next, the impact on application performance of real-life communication architectures is demonstrated by means of examples for bus-based and NoC-based MPSoCs. The baseline systems introduced in the example are then enhanced with architectural features that improve the system performance. These interconnect paradigms and features are analyzed in a comprehensive case-study. Furthermore, the joint effects between OSIP and the communication architecture are extensively investigated, using the above-mentioned H.264 video decoding and MIMO receiver application and an additional generic synthetic application. Finally, conclusions are drawn and an outline for future work is provided.

RUNTIME MANAGEMENT

Various approaches have been proposed in academia and industry for runtime management. Gen-

34 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage: www.igi-global.com/chapter/system-level-analysis-of-mpsocs-with-a-hardware-scheduler/146418

Related Content