Chapter 2
Multithreaded Programming of Reconfigurable Embedded Systems

Jason Agron
University of Arkansas, USA

David Andrews
University of Arkansas, USA

Markus Happe
University of Paderborn, Germany

Enno Lübbers
University of Paderborn, Germany

Marco Platzner
University of Paderborn, Germany

ABSTRACT

Embedded and Real-Time (ERTS) systems have continued to expand at a vigorous rate. Designers of ERTS systems are continually challenged to provide new capabilities that can meet the expanding requirements and increased computational needs of each new proposed application, but at a decreasing price/performance ratio. Conventional solutions using general purpose processors or custom ASICs are less and less able to satisfy the contradictory requirements in performance, flexibility, power, development time, and cost. This chapter introduces the concept of generating semi-custom platforms driven from a traditional multithreaded programming model. This approach offers the advantage of achieving productivity levels close to those associated with software by using an established programming model but with a performance level close to custom hardware through the use of a flexible hardware platform capable of adapting to specialized application requirements. We discuss the underlying concepts, requirements and advantages of multithreading in the context of reconfigurable hardware, and present two approaches which provide multithreading support to hardware and software components at the operating system level.

DOI: 10.4018/978-1-60960-086-0.ch002
INTRODUCTION

Embedded and Real-Time (ERT) systems continue to expand at a vigorous rate. From personal electronics, through transportation, and into the telecommunications industries, ERT systems continue to redefine our personal and professional lifestyles. With each new application and usage, the designers of ERT systems are continually faced with the challenge of fielding systems with expanding performance and capabilities, but within constricting price/performance ratios. Designers are now turning to modern Field Programmable Gate Arrays (FPGAs) to help resolve these conflicting requirements. FPGAs have long held the promise of allowing designers to create systems with performance levels close to custom circuits but with the ability to reconfigure gates with software-like levels of flexibility. While the performance of FPGA-based systems are now very impressive, productivity levels for helping guide the building of FPGA-based systems has historically remained closer to those associated with designing hardware.

In this chapter we show how we have enabled the multithreaded programming model to close the productivity gap and guide the development of systems on programmable chip (SopCs). To aid in design productivity and reduce design time, the application can first be created as a group of threads, and run and tested for functional correctness on a standard desktop PC. Once tested, the threads can then be profiled and evaluated to determine the specific type of computational component required to meet timing and performance requirements.

Using this abstraction, designers are free to implement the individual threads on appropriate combinations of commercially available or custom soft IP cores within FPGA. The soft cores can be general purpose programmable CPUs, more powerful Digital Signal Processing Units (DSPs), semi-custom and extensible processors, or fully custom hardware accelerators.

Important for both designer efficiency and system correctness, the high level communication and synchronization primitives defined by the model to co-ordinate interactions between threads are provided through a standard set of Application Programmer Interfaces (APIs). The use of APIs allows the separation of policy and mechanism. The API calls are provided to the designer for use within the application program and are independent of the computational components on which the threads execute. Only the lower level implementations of the model’s calls are platform specific. We provide these implementations as linkable libraries for general threads that are run on purpose processors, and as includable VHDL for threads that are turned into custom hardware accelerators.

In the remainder of this section we provide an introduction to the generalized multithreaded programming model. We then provide an overview of the relationship between fabrication advancements for FPGA components and corresponding developments in software infrastructure and design abstractions. We finish this section by showing how the multithreaded programming model can be used to help create a complete multiprocessor system on chip. In Section 3, we show how the standard policies of the model are provided through two separate operating systems for FPGAs; Hthreads and ReconOS. Both operating systems provide designers with the same policies but through different mechanisms. Section 4 concludes the chapter with projections on how FPGAs and the multithreaded programming model are moving towards supporting systems with heterogeneous multiprocessor systems on chip.

BACKGROUND

The Multithreaded Programming Model

Although the multithreading programming model has become widely popular within the last few
Related Content

TEA: A Generic Framework for Decision Making in Web Services
Zhaohao Sun, Grant Meredith and Andrew Stranieri (2012). *International Journal of Systems and Service-Oriented Engineering* (pp. 41-63).
www.igi-global.com/article/tea/79238?camid=4v1a

Security Evaluation of Service-Oriented Systems Using the SiSOA Method
www.igi-global.com/article/security-evaluation-service-oriented-systems/61151?camid=4v1a

Security in Service-Oriented Architecture: Issues, Standards and Implementations
www.igi-global.com/chapter/security-service-oriented-architecture/28960?camid=4v1a

Cooperation Patterns and Adaptation Patterns for Service-Based Inter-Organizational Workflows
www.igi-global.com/chapter/cooperation-patterns-and-adaptation-patterns-for-service-based-inter-organizational-workflows/96624?camid=4v1a