EMBEDDED SOFTWARE FOR SoC

Embedded Software for SoC

Edited by

Ahmed Amine Jerraya
TIMA Laboratory, France

Sungjoo Yoo
TIMA Laboratory, France

Diederik Verkest
IMEC, Belgium

and

Norbert Wehn
University of Kaiserlautern, Germany

KLUWER ACADEMIC PUBLISHERS
NEW YORK, BOSTON, DORDRECHT, LONDON, MOSCOW

eBook ISBN: 0-306-48709-8
Print ISBN: 1-4020-7528-6

Created in the United States of America

Visit Springer's eBookstore at: http://www.ebooks.kluweronline.com
and the Springer Global Website Online at: http://www.springeronline.com

DEDICATION

This book is dedicated to all designers working in hardware hell.

TABLE OF CONTENTS

PREFACE

The evolution of electronic systems is pushing traditional silicon designers into areas that require new domains of expertise. In addition to the design of complex hardware, System-on-Chip (SoC) design requires software development, operating systems and new system architectures. Future SoC designs will resemble a miniature on-chip distributed computing system combining many types of microprocessors, re-configurable fabrics, application-specific hardware and memories, all communicating via an on-chip inter-connection network. Designing good SoCs will require insight into these new types of architectures, the embedded software, and the interaction between the embedded software, the SoC architecture, and the applications for which the SoC is designed.

This book collects contributions from the Embedded Software Forum of the Design, Automation and Test in Europe Conference (DATE 03) that took place in March 2003 in Munich, Germany. The success of the Embedded Software Forum at DATE reflects the increasing importance of embedded software in the design of a System-on-Chip.

Embedded Software for SoC covers all software related aspects of SoC design

- Embedded and application-domain specific operating systems, interplay between application, operating system, and architecture.
- System architecture for future SoC, application-specific architectures based on embedded processors and requiring sophisticated hardware/software interfaces.
- Compilers and interplay between compilers and architectures.
- Embedded software for applications in the domains of automotive, avionics, multimedia, telecom, networking, . . .

This book is a must-read for *SoC designers* that want to broaden their horizons to include the ever-growing embedded software content of their next SoC design. In addition the book will provide *embedded software designers* invaluable insights into the constraints imposed by the use of embedded software in a SoC context.

Diederik Verkest
IMEC
Leuven, Belgium

Norbert Wehn
University of Kaiserslautern
Germany

INTRODUCTION

Embedded software is becoming more and more important in system-on-chip (SoC) design. According to the ITRS 2001, "embedded software design has emerged as the most critical challenge to SoC" and "Software now routinely accounts for 80% of embedded systems development cost" [1]. This will continue in the future. Thus, the current design productivity gap between chip fabrication and design capacity will widen even more due to the increasing 'embedded SoC SW implementation gap'. To overcome the gap, SoC designers should know and master embedded software design for SoC. The purpose of this book is to enable current SoC designers and researchers to understand up-to-date issues and design techniques on embedded software for SoC.

One of characteristics of embedded software is that it is heavily dependent on the underlying hardware. The reason of the dependency is that embedded software needs to be designed in an application-specific way. To reduce the system design cost, e.g. code size, energy consumption, etc., embedded software needs to be optimized exploiting the characteristics of underlying hardware.

Embedded software design is not a novel topic. Then, why do people consider that embedded software design is more and more important for SoC these days? A simple, maybe not yet complete, answer is that we are more and more dealing with platform-based design for SoC [2].

Platform-based SoC design means to design SoC with relatively fixed architectures. This is important to reduce design cycle and cost. In terms of reduction in design cycle, platform-based SoC design aims to reuse existing and proven SoC architectures to design new SoCs. By doing that, SoC designers can save architecture construction time that includes the design cycle of IP (intellectual property core) selection, IP validation, IP assembly, and architecture validation/evaluation.

In platform-based SoC design, architecture design is to configure, statically or dynamically in system runtime, the existing platforms according to new SoC designs [3]. Since the architecture design space is relatively limited and fixed, most of the design steps are software design. For instance, when SoC designers need to implement a functionality that is not implemented by hardware blocks in the platform, they need to implement it in software. As the SoC functionality becomes more complex, software will implement more and more functionality compared to the relatively fixed hardware. Thus, many design optimization tasks will become embedded software optimization ones.

To understand embedded software design for SoC, we need to know current issues in embedded software design. We want to classify the issues into two parts: software reuse for SoC integration and architecture-specific software optimization. Architecture-specific software optimization has been studied for decades. On the other side, software reuse for SoC integration is an important new issue. To help readers to understand better the specific contribution of this book, we want to address this issue more in detail in this introduction.

SW REUSE FOR SOC INTEGRATION

Due to the increased complexity of embedded software design, the design cycle of embedded software is becoming the bottleneck to reduce time-to-market. To shorten the design cycle, embedded software needs to be reused over several SoC designs. However, the hardware dependency of embedded software makes software reuse very difficult.

A general solution to resolve this software reuse problem is to have a multi-layer architecture for embedded software. Figure 1 illustrates such an architecture. In the figure, a SoC consists of sub-systems connected with each other via a communication network. Within each sub-system, embedded

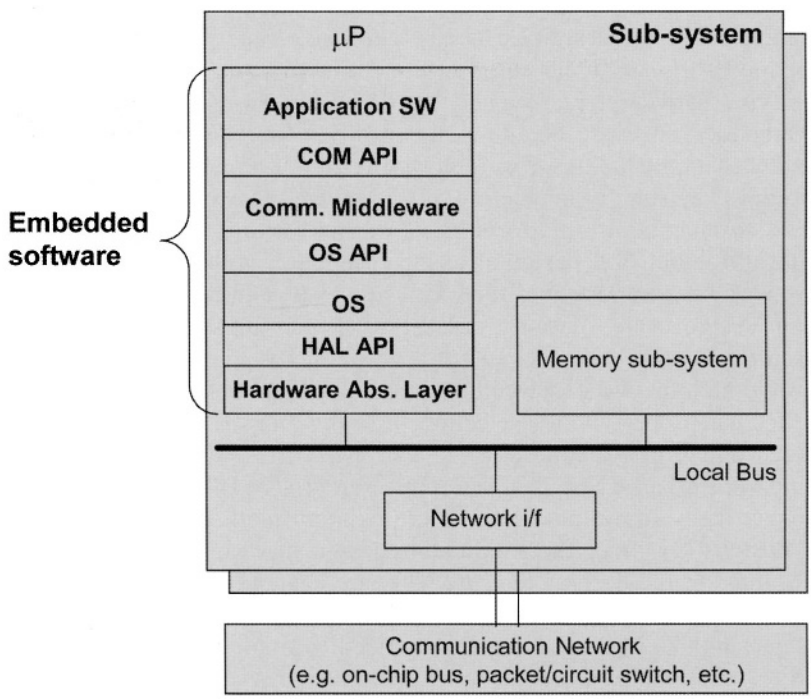

Figure 1. Multi-layer software architecture in SoC.

software consists of several layers: application software, communication middleware (e.g. message passing interface [4]), operating system (OS), and hardware abstraction layer (HAL)). In the architecture, each layer uses an abstraction of the underlying ones. For instance, the OS layer is seen by upper layers (communication middleware and application layers) as an abstraction of the underlying architecture, in the form of OS API (application programming interface), while hiding the details of OS and HAL implementation and those of the hardware architecture.

Embedded software reuse can be done at each layer. For instance, we can reuse an RTOS as a software component. We can also think about finer granularity of software component, e.g. task scheduler, interrupt service routine, memory management routine, inter-process communication routine, etc. [5].

By reusing software components as well as hardware components, SoC design becomes an integration of reused software and hardware components. When SoC designers do SoC integration with a platform and a multi-layer software architecture, the first question can be 'what is the API that gives an abstraction of my platform?' We call the API that abstracts a platform 'platform API'. Considering the multi-layer software architecture, the platform API can be Communication API, OS API, or HAL API. When we limit the platform only to the hardware architecture, the platform API can be an API at transaction level model (TLM) [6]. We think that a general answer to this question may not exist. The platform API may depend on designer's platforms. However, what is sure is that the platform API needs to be defined (by designers, by standardization institutions like Virtual Socket Interface Alliance, or by anyone) to enable platform-based SoC design by reusing software components.

In SoC design with multi-layer software architecture, another important problem is the validation and evaluation of reused software on the platform. Main issues are related to software validation without the final platform and, on the other hand, to assess the performance of the reused software on the platform. Figure 2 shows this problem more in detail. As shown in the figure,

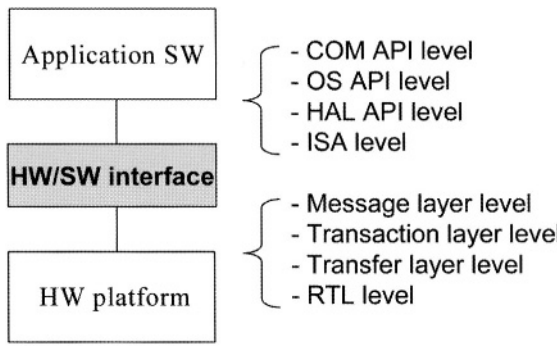

Figure 2. Validation and evaluation of reused embedded software and hardware platform.

software can be reused at one of several abstraction levels, Communicaton API, OS API, HAL API, or ISA (instruction set architecture) level, each of which corresponds to software layer. The platform can also be defined with its API. In the figure, we assume a hardware platform which can be reused at one of the abstraction levels, message, transaction, transfer layer, or RTL [6]. When SoC designers integrate both reused software and hardware platform at a certain abstraction level for each, the problem is how to validate and evaluate such integration. As more software components and hardware platforms are reused, this problem will become more important.

The problem is to model the interface between reused software and hardware components called 'hardware/software interface' as shown in Figure 2. Current solutions to model the HW/SW interface will be bus functional model, BCA (bus cycle accurate) shell, etc. However, they do not consider the different abstraction levels of software. We think that there has been little research work covering both the abstraction levels of software and hardware in this problem.

GUIDE TO THIS BOOK

The book is organised into 10 parts corresponding to sessions presented at the Embedded Systems Forum at DATE'03. Both software reuse for SoC and application specific software optimisations are covered.

The topic of **Software reuse for SoC** integration is explained in three parts "Embedded Operating System for SoC", "Embedded Software Design and Implementation", "Operating System Abstraction and Targeting". The key issues addressed are:

- The **layered software architecture and its design** in chapters 3 and 9.
- The **OS layer design** in chapters 1, 2, 3, and 7.
- The **HAL layer** in chapter 1.
- The **problem of modelling the HW/SW interface** in chapters 5 and 8.
- **Automatic generation of software layers,** in chapters 6 and 11.
- **SoC integration** in chapters 10, 12 and 13.

Architecture-specific software optimization problems are mainly addressed in five parts, "Software Optimization for Embedded Systems", "Embedded System Architecture", "Transformations for Real-Time Software", "Energy Aware Software Techniques", and "Low Power Software". The key issues addressed are:

- **Sub-system-specific techniques** in chapters 18, 19, 26, 29, 30 and 31.
- **Communication-aware techniques** in chapters 23, 24, 27 and 28.
- **Architecture independent solutions** which perform code transformation to enhance performance or to reduce design cost without considering specific target architectures are presented in chapters 17, 20, 21 and 33.

– **Energy-aware techniques** in chapters 22, 23, 24, 34, 35, 36 and 37.
– **Reliable embedded software design techniques** in chapters 4, 25 and 32.

REFERENCES

1. *International Technology Roadmap for Semiconductors,* available at http://public.itrs.net/
2. Alberto Sangiovanni-Vincentelli and Grant Martin. "Platform-Based Design and Software Design Methodology for Embedded Systems." *IEEE Design & Test of Computers,* November/December 2001.
3. Henry Chang, Larry Cooke, Merrill Hunt, Grant Martin, Andrew McNelly, and Lee Todd. *Surviving the SOC Revolution, A Guide to Platform-Based Design.* Kluwer Academic Publishers, 1999.
4. *The Message Passing Interface Standard,* available at http://www-unix.mcs.anl.gov/mpi/
5. Anthony Massa. *Embedded Software Development with eCos.* Prentice Hall, November 2002.
6. *White Paper for SoC Communication Modeling*, available at http://www.synopsys.com/products/cocentric_studio/communication_wp10.pdf

Sungjoo Yoo
TIMA
Grenoble, France

Ahmed Amine Jerraya
TIMA
Grenoble, France