Chapter 3

Abstract and Concrete Data Type Optimizations at the UML and C/C++ Level for Dynamic Embedded Software

Christos Baloukas  
Democritus University of Thrace, Greece

Marijn Temmerman  
Karel de Grote Hogeschool, Belgium

Anne Keller  
University of Antwerpen, Belgium

Stylianos Mamagkakis  
IMEC, Belgium

Francky Catthoor  
IMEC, Belgium

Dimitrios Soudris  
National Technical University of Athens, Greece

Serge Demeyer  
University of Antwerpen, Belgium

ABSTRACT

An embedded system is a special-purpose system that performs predefined tasks, usually with very specific requirements. Since the system is dedicated to a specific task, design engineers can optimize it by exploiting very specialized knowledge, deriving an optimally customized system. Low energy consumption and high performance are both valid optimization targets to increase the value and mobility of the final system. Traditionally, conceptual embedded software models are built irrespectively of the underlying hardware platform, whereas embedded-system specialists typically start their optimization

DOI: 10.4018/978-1-60566-750-8.ch003
Abstract and Concrete Data Type Optimizations at the UML and C/C++ Level

INTRODUCTION

Modern embedded systems are a combination of software and hardware, which perform a single or a few dedicated functions. In the context of this chapter, we will focus on embedded systems that perform more than one function and more specifically on the category of nomadic embedded systems. These are mobile devices that have multimedia or wireless communication functionality (or both) and provide to the end user the ease of rich multimedia content anytime, anywhere. To provide this functionality, they rely on hardware platforms, which integrate a number of processing elements, communication infrastructure and memory hierarchy on a single chip (or multiple chips on a board). The processing elements are programmable, which means that they can perform more than one function according to the embedded software that accompanies those (Catthoor & Brockmeyer, 2000). We classify embedded software as application software and system software, as can be seen in Figure 1. Application software performs some tasks relevant to the end user and the system software interfaces with hardware to provide the necessary services for application software. The embedded system designers are responsible for the design, integration and optimization of the hardware, application software and system software components.

In embedded systems, the software is gaining an ever increasing importance. Because hardware becomes more programmable, multiple applications can run on the same hardware platform through dedicated software. In this context, we define as embedded software applications the different programs, which are written for each one of these applications (from now on, we will simply refer to them as applications in the text). It is important to note that applications receive input from and serve directly to the end user of the embedded system. This means that the actions of the user have significant impact on the control flow of the algorithms in the applications, thus making the execution dynamic and event-driven.

The proposed optimization methodology is part of a general design flow. This flow targets a crusade from the executable code. This practice results in suboptimal implementations on the embedded platform because at the source-code level not all the inefficiencies introduced at the modelling level can be removed. In this book chapter, we describe both novel UML transformations at the modelling level and C/C++ transformations at the software implementation level. The transformations at both design abstraction levels target the data types of dynamic embedded software applications and provide optimizations guided by the relevant cost factors. Using a real life case study, we show how our transformations result in significant improvement in memory footprint, performance and energy consumption with respect to the initial implementation. Moreover, thanks to our holistic approach, we are able to identify new and non-trivial solutions that could hardly be found with the traditional design methods.

Figure 1. Hardware and software components in an embedded system
27 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the product’s webpage:

www.igi-global.com/chapter/abstract-concrete-data-type-optimizations/36338?camid=4v1


www.igi-global.com/e-resources/library-recommendation/?id=1

Related Content

Is There a Life After Objects?
Jean Bezivin (2002). Optimal Information Modeling Techniques (pp. 84-94).
www.igi-global.com/chapter/there-life-after-objects/27827?camid=4v1a

Service Oriented Enterprise and Contracted Profit Sharing
www.igi-global.com/article/service-oriented-enterprise-contracted-profit/55124?camid=4v1a

Formal Specification and Implementation of Priority Queue using Stream Functions
www.igi-global.com/article/formal-specification-and-implementation-of-priority-queue-using-stream-functions/103282?camid=4v1a

Requirement Prioritization of Complex Web 2.0 Application based on Effects on Regression Testing: A Hybrid Approach
www.igi-global.com/article/requirement-prioritization-of-complex-web-20-application-based-on-effects-on-regression-testing/134432?camid=4v1a