Chapter 7

SYSTEMATIC EMBEDDED SOFTWARE GENERATION FROM SystemC

F. Herrera, H. Posadas, P. Sánchez and E. Villar
TEISA Dept., E.T.S.I. Industriales y Telecom., University of Cantabria, Avda. Los Castros s/n, 39005 Santander, Spain; E-mail: {fherrera, posadash, sanchez, villar}@teisa.unican.es

Abstract. The embedded software design cost represents an important percentage of the embedded-system development costs [1]. This paper presents a method for systematic embedded software generation that reduces the software generation cost in a platform-based HW/SW codesign methodology for embedded systems based on SystemC. The goal is that the same SystemC code allows system-level specification and verification, and, after SW/HW partition, SW/HW co-simulation and embedded software generation. The C++ code for the SW partition (processes and process communication including HW/SW interfaces) is systematically generated including the user-selected embedded OS (e.g.: the eCos open source OS).

Key words: Embedded Software generation, SystemC, system-level design, platform based design

1. INTRODUCTION

The evolution of technological processes maintains its exponential growth; 810 Mtrans/chip in 2003 will become 2041 Mtrans/chip in 2007. This obliges an increase in the designer productivity, from 2.6 Mtrans/py in 2004 to 5.9 Mtrans/py in 2007, that is, a productivity increase of 236% in three years [2]. Most of these new products will be embedded System-on-Chip (SoC) [3] and include embedded software. In fact, embedded software now routinely accounts for 80% of embedded system development costs [1].

Today, most embedded systems are designed from a RT level description for the HW part and the embedded software code separately. Using a classical top-down methodology (synthesis and compilation) the implementation is obtained. The 2001 International Technology Roadmap for Semiconductors (ITRS) predicts the substitution (during the coming years) of that waterfall methodology by an integrated framework where codesign, logical, physical and analysis tools operate together. The design step being where the designer envisages the whole set of intended characteristics of the system to be implemented, system-level specification acquires a key importance in this new
design process since it is taken as the starting point of all the integrated tools and procedures that lead to an optimal implementation [1, 2].

The lack of a unifying system specification language has been identified as one of the main obstacles bedeviling SoC designers [4]. Among the different possibilities proposed, languages based on C/C++ are gaining a wider-consensus among the designer community [5], SystemC being one of the most promising proposals. Although, the first versions of SystemC were focused on HW design, the latest versions (SystemC2.x [6]) include some system-level oriented constructs such as communication channels or process synchronization primitives that facilitate the system specification independently of the final module implementation in the HW or SW partition.

Embedded SW generation and interface synthesis are still open problems requiring further research [7]. In order to become a practical system-level specification language, efficient SW generation and interface synthesis from SystemC should be provided. Several approaches for embedded SW design have been proposed [8–10]. Some of them are application-oriented (DSP, control, systems, etc.), where others utilise input language of limited use.

SoCOS [11, 12] is a C++ based system-level design environment where emphasis is placed on the inclusion of typical SW dynamic elements and concurrency. Nevertheless, SoCOS is only used for system modeling, analysis and simulation.

A different alternative is based on the synthesis of an application-specific RTOS [13–15] that supports the embedded software. The specificity of the generated RTOS gives efficiency [16] at the expense of a loss of verification and debugging capability, platform portability and support for application software (non firmware). Only very recently, the first HW/SW co-design tools based on C/C++-like input language have appeared in the marketplace [17]. Nevertheless, their system level modeling capability is very limited.

In this paper, an efficient embedded software and interface generation methodology from SystemC is presented. HW generation and cosimulation are not the subject of this paper. The proposed methodology is based on the redefinition and overloading of SystemC class library elements. The original code of these elements calls the SystemC kernel functions to support process concurrency and communication. The new code (defined in an implementation library) calls the embedded RTOS functions that implement the equivalent functionality. Thus, SystemC kernel functions are replaced by typical RTOS functions in the generated software. The embedded system description is not modified during the software and interface generation process. The proposed approach is independent of the embedded RTOS. This allows the designer to select the commercial or open source OS that best matches the system requirements. In fact, the proposed methodology even supports the use of an application-specific OS.

The contents of the paper are as follows. In this section, the state of the art, motivation and objectives of the work have been presented. In section 2, the system-level specification methodology is briefly explained in order to