Microprocessors and Microsystems 39 (2015) 1012–1028

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Automatic communication-driven virtual prototyping and design for networked embedded systems

Joachim Falk *, Tobias Schwarzer, Liyuan Zhang, Michael Glaß, Jürgen Teich

Hardware/Software Co-Design, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

ARTICLE INFO

Article history:
Received 19 December 2014
Revised 22 July 2015
Accepted 14 August 2015
Available online 2 September 2015

Keywords:
Virtual prototyping
ESL design flow
Communication refinement
SystemC-TLM

ABSTRACT

This work presents a *communication-driven virtual prototyping* approach integrated in an existing ESL design methodology to automatically synthesize, evaluate, and optimize a data-flow application for mixed hardware/software and even networked MPSoCs. While existing synthesis tools are suitable for individual subsystems (e.g., software tasks for CPUs, hardware accelerators), the problem of establishing the communication between different subsystems that may even be simulated at different levels of abstraction is still challenging. As a remedy, we introduce the concept of *bridge* components in our architecture model that, during virtual prototyping, serve as integrators between subsystems that may have different communication protocols and be simulated at different levels of abstraction (e.g., TLM, behavioral level, RTL). We propose to consider bridges throughout the complete ESL design flow: Already during Design Space Exploration (DSE), the characteristics of bridge components such as implementation cost and additional latency on the application can be taken into account. Moreover, we extend the exploration model of the DSE to include required communication-related design decisions, i.e., the mapping of binary code for software tasks and the selection of different synchronization patterns for the communication. For virtual prototyping of implementation candidates derived by the DSE, the bridge components enable to automatically disassemble the system into subsystems and hand each subsystem over to an individual synthesis tool. When integrating the subsystems together, our methodology also synthesizes the interfaces for all bridges which significantly simplifies system integration. As a proof of concept, we present (I) a distributed control application that is transformed into a virtual prototype consisting of six subsystems and (II) a data-flow application from the video processing domain transformed into a virtual prototype consisting of three subsystems. The resulting subsystems can be concurrently simulated at TLM, behavioral level, and RTL. The experiments give evidence of the proposed technique's applicability, the achieved productivity gain, and the resulting simulation performance at the considered levels of abstraction.

1. Introduction

Driven by the rapid development of microelectronics technology, the functionality and the design complexity of modern distributed embedded systems are continuously increasing. To cope with these challenges, *Electronic System Level* (ESL) [1] design methodologies raise the level of abstraction and model the complete embedded system as an *executable specification*, e.g., following the data flow paradigm [2], at system level. At this level, decisions such as the mapping of functional units—modeled as *actors* in data flow—to software or hardware are yet to be made. In fact, those decisions shall be investigated automatically by means of a *Design Space Exploration* (DSE) [3] that evaluates different design decisions and searches for the best implementation candidates. After or even during DSE, an implementation candidate may be evaluated using a *virtual prototype* to determine its quality numbers such as cost, latency, or energy consumption. In *system synthesis* [3], numerous synthesis techniques like software compilation, behavioral synthesis, or communication scheduling are available to derive a virtual prototype. However, these are only suitable for subsystems of the architecture. This typically necessitates a manual disassembly of the system into subsystems that are suitable for the respective synthesis tools. For the subsequent integration process, two challenges arise: (I) Numerous communication domains exist: Signal-based communication, on-chip and

* Corresponding author.
E-mail addresses: falk@cs.fau.de (J. Falk), tobias.schwarzer@cs.fau.de (T. Schwarzer), liyuan.zhang@cs.fau.de (L. Zhang), glass@cs.fau.de (M. Glaß), teich@cs.fau.de (J. Teich).

http://dx.doi.org/10.1016/j.micpro.2015.08.008
0141-9331/

shared memory buses, or even field buses like CAN or Ethernet. (II) During virtual prototyping, a sound trade-off between accuracy and simulation performance may only be achieved by simulating different subsystems at different levels of abstraction such as TLM, behavioral level, or RTL. Thus, the integration process, which has to establish a communication between subsystems wrt. their communication protocols and simulation level of abstraction, becomes a major bottle neck for virtual prototyping in ESL design methodologies.

In this work, we propose an approach for communication-driven automatic virtual prototyping as part of an ESL design flow (see Fig. 1) for data-flow applications on mixed hardware/software and even networked MPSoCs. The approach contributes two main aspects, the first being a bridge concept that tackles the problem of subsystem integration and the second being a significantly extended DSE model that puts focus on memory allocation and different synchronization patterns for communication.

Contribution I **The first key idea** of the proposed communication-driven virtual prototyping is to consider points at which data is transferred between different subsystems. There, we introduce the concept of *bridge* components directly in the architecture model. During virtual prototyping, these bridges serve as integrators between subsystems that may have different communication protocols and be simulated at different levels of abstraction. The number of bridges depends on the number of different subsystems and, hence, once introduced, we do not expect any significant increase in complexity. The proposed bridge components are not only employed for virtual prototyping, but considered throughout the complete ESL design flow: (I) Integrated in the architecture model, the characteristics of bridge components such as implementation cost and additional latency for the application can be taken into account already during DSE. (II) To gather virtual prototypes of implementation candidates, the bridge components enable an automatic disassembly of the system into subsystems, each processed by a specific synthesis tool. (III) Moreover, the interfaces for all bridges are synthesized as well, significantly simplifying the system integration process. (IV) Finally, in the actual system implementation, bridges correspond to real hardware or software components that may be generated, e.g., via interface synthesis [4].

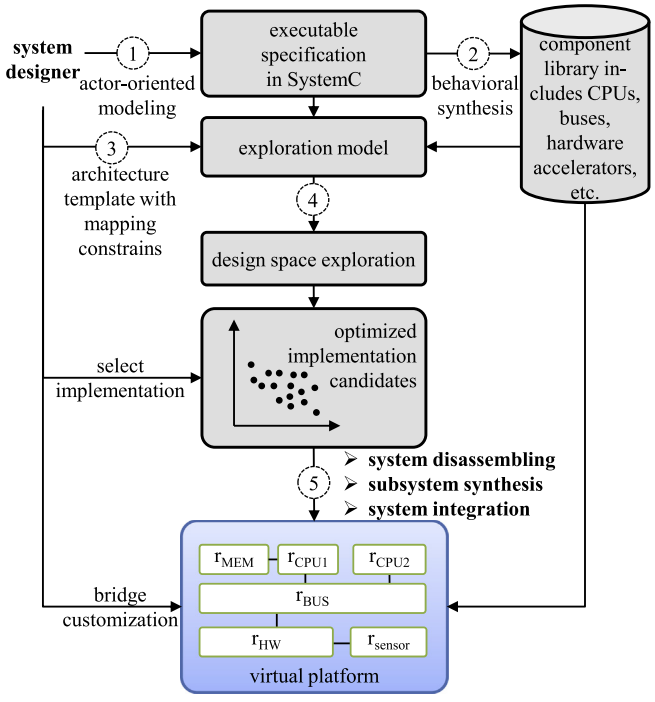

Fig. 1. The proposed virtual prototyping is integrated into an ESL design flow.

Contribution II We extend a state-of-the-art exploration model to consider not only application data to be transmitted in the system, but to also take into account (I) the synchronization patterns for the communication, (II) the storage of binary code for software tasks, as well as (III) enable an automatic derivation of the address forwarding in the bus resources from the symbolic routing information provided by the DSE methodology in [5,6] for our extended exploration model. These aspects are not only essential to automatically synthesize working prototypes, but have to be considered during DSE already since they may significantly influence the system's characteristics, e.g., delays or bandwidth and memory requirements. Thus, additional optimization potential wrt. memory allocation for binary code and different synchronization patterns is exposed to the DSE.

As a proof of concept, a distributed control application, i.e., the Tower of Hanoi puzzle solved by two robot arms, is modeled as a data-flow application. The architecture and, hence, the virtual prototype, consists of six subsystems that are concurrently simulated at TLM, behavioral level, and RTL. The virtual prototype even includes a co-simulation with a physical model of the robot arms in Matlab/Simulink. This co-simulation is realized using our *co-simulation framework* [7] to coordinate the data exchange and the synchronization between the SystemC and the Matlab/Simulink environments. The experiment gives evidence of the proposed technique's applicability and the achieved productivity gain while coming at a very low overhead of specifying bridge components in the system's architecture model. The evidence is supported by a second considered application, i.e., a Motion-JPEG decoder application from the signal processing domain. In this experiment, we additionally show how our concept of *bridge* components can be used to facilitate co-simulation between software derived from actors mapped to CPUs and the rest of the application still modeled at system level as a DFG. In the resulting virtual prototype, the software is executed on an Instruction Set Simulator (ISS) corresponding to the CPU resources and the rest of the application is modeled at SystemC behavioral level.

The remainder of this paper is structured as follows: Section 2 discusses previous and related work. In Section 3, an overview of the ESL design flow is given, as well as the extension we made for DSE. Section 4 presents the proposed communication-driven virtual prototyping methodology. The results of the virtual prototyping applied to a networked control system and a data-flow application from the video processing domain are presented in Section 5 before the paper is concluded in Section 6.

2. Previous and related work

In the following, we first present previous works that are the base for the contributions presented here and discuss related work afterwards.

2.1. Previous work

The fundamentals for our DSE approach are given by a *classic* exploration model proposed in [8]. Subsequently, communication limitations of the classic exploration model have been resolved by enabling the exploration of message routing for multi-hop communications [5] resulting in a state-of-the-art exploration model introduced in [6]. Furthermore, we have shown that these kinds of exploration models can be efficiently explored [6]—even when handling very large test cases—using Boolean satisfiability-based approaches [9] and that the exploration efficiency can even be