INTEGRATION the VLSI journal 58 (2017) 91–100

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

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

On supporting rapid prototyping of embedded systems with reconfigurable architectures

I. Koutras*, K. Maragos, D. Diamantopoulos, K. Siozios, D. Soudris

School of Electrical and Computer Engineering, National Technical University of Athens, Zografou Campus, 15780 Athens, Greece

ARTICLE INFO

Keywords:
Reconfigurable architectures
Rapid prototyping
Genetic algorithm

ABSTRACT

Reducing time-to-market while improving product quality is a big challenge. This paper proposes a software-supported framework for rapid prototyping that offers a concurrent fast hardware/software system-level design. The introduced framework enables the constant evaluation and verification of the prototype under development, while it provides automatic functionality mapping to hardware via High-Level Synthesis techniques. We evaluate our framework and its software instantiation with a computer vision algorithm. Based on our experimentation, we show that our approach reduces the development time by almost 64×, it prunes the hardware design space by 34×, while maintaining designs that trade-off high Quality-of-Report on the Pareto frontier.

1. Introduction

Designing full system solutions is a complex task. With vastly increased complexity and functionality especially in the nanometer era, where hundreds of millions of transistors are integrated on a single chip, the design of complex Integrated Circuits (ICs) has become a challenging task. In addition to that, the continuously increased demand for even higher performance, i.e. in terms of operation frequency and power consumption, imposes that new design techniques are absolutely required.

This problem becomes far more important if we take into consideration that software aspects of ICs can account for 80%, or more, of embedded systems development cost, making the conventional way for product development insufficient. For instance, the International Technology Roadmap for Semiconductors (ITRS) [1] predicts that software development costs will increase and will reach rough parity with hardware costs, even with the advent of multi-core software development tools.

Electronic Design Automation (EDA) tools are crucial nowadays for deriving optimal solutions. Existing working flows are built on the fundamental premise that models are fully interchangeable and interoperable among different EDA vendors for the whole physical prototyping process, as in architectural analysis, simulation and synthesis. Even though this concept seems straightforward and promising, it has been proven completely elusive in the world of Electronic System Level: existing solutions do not provide either model interoperability, neither independence between model and software tools. As such, it is often desired to reach the highest possible systemic level of the target application description in order to avoid a possible vendor lock-in.

Apart from the technology-oriented parameters that affect the efficiency and/or the flexibility of a digital system, the tight time-to-market requirements make conventional ways for product development, e.g. start software development after finalising hardware, to lead usually in missed market windows and revenue opportunities. Hence, there is an absolute requirement for software developers to get an early start on their work, long before the Register-Transfer Level of the hardware is finalised.

Towards this direction, and as research pushes for better programming models for multi-processor and multi-core embedded systems, Virtual Platforms (VPs) solve one of today's biggest challenges in physical design: to enable sufficient software development, debug and validation before the hardware device becomes available. More specifically, with the virtualization feature, it is possible to model a hardware platform consisted of different processing cores, memories, peripherals, as well as interconnection schemes, in the form of a simulator. Furthermore, as the task of hardware development progressively proceeds, it is feasible to redistribute to software teams updated versions of the VP, that enable a gradually better description of the target architecture.

The concept of virtualization is also important for hardware architects, as it enables easier verification of Intellectual Properties (IP) kernels. This feature could be employed both in the case where only a few of the application kernels have to be developed in hardware, as well as if incremental system prototyping is performed. In both

* Corresponding author.
E-mail address: joko@microlab.ntua.gr (I. Koutras).

http://dx.doi.org/10.1016/j.vlsi.2017.02.007
Received 30 April 2015; Received in revised form 16 February 2017; Accepted 16 February 2017
Available online 16 March 2017

cases, the virtualization feature provides all the necessary mechanisms for performing co-simulation and verification between the IPs developed in Register Transfer Level (RTL) and the rest application functionalities executed onto the VP.

In this paper we identify common pitfalls during virtual prototyping for hardware/software co-design and propose a software-supported methodology to perform rapid system-level prototyping of complex digital systems. More specifically we:

- Present a modern virtual prototyping platform in Section 2 and explain some of the most time-consuming steps in development.
- Define and extract necessary task information through profiling, high-level synthesis and task execution on an FPGA (Section 3.1).
- Model task mapping as an optimisation problem and solve it with genetic algorithms (Subsection section 3.2).
- Optimise the hardware-assigned tasks assigned to hardware by performing further DSE with the help of FPGA (Subsection section 3.3).
- Evaluate our proposed working flow on the Harris & Stephens Corner Detection Algorithm from Computer Vision (Section 4). From a full-software solution we reach to an optimal mixed (hardware/software) one 6 times faster than a conventional approach to virtual prototyping (Section 5).
- Mention other relevant techniques that can be used for task partitioning, as well as other prototyping frameworks, and explain where and why our proposed toolflow works better (Section 6).

Our conclusion is that rapid prototyping of multi-million gate systems is achievable. We can have prototypes of our system on-the-go, as we modify, add, or optimise the algorithms that describe the system tasks.

2. Background: virtual prototyping

Fig. 1 depicts three consecutive design stages while using a virtual prototyping platform: (i) system modeling, (ii) rapid virtual prototyping and (iii) system integration. Different virtualization environments can be employed for this purpose. Without affecting the generality of VPs, we refer here to the OVP [2], since it is a publicly available and easily extensible approach. Additionally, the increased simulation speed provided by OVPSim ensures that complex systems can be modeled in reasonable amount of time (hundreds of millions of simulated instructions per second). As the OVP models are pre-built, they support fully functional simulation of a complete embedded system. Also, since these models are binary-compatible with the simulated hardware, the developed software can be executed onto the final system without any modifications. This enables faster iteration for the software development teams.

Similarly, hardware developers are also benefited from the adoption of hybrid VP. Since this platform is composed of a simulator and TLM/SystemC models, it exhibits increased flexibility which in turn alleviates many constraints that designers face during the architecture design. More specifically, the former models (related to OVP) describe the software part of the target system (e.g., executed onto an embedded processor), while the TLM/SystemC models provide the design functionality that has been mapped to custom hardware IPs (e.g., FPGA) after system mapping.

Connecting the hardware IPs with the functionality mapped to software is imperative in hardware/software co-design. Platform connectivity between the off-chip world and the functions in software is necessary once functions are implemented in hardware. Accordingly, a communication layer between the hardware-dependent software and the custom hardware IPs is necessary, as well as to provide all the necessary synchronisation for the computation tasks mapped in hardware and software. In the VP approach we refer to, this is implemented as a software stack running on a native host (x86-compatible) and more specifically to Fig. 1 as the “HotTalk API”.

HotTalk API [3] provides a wide class of middleware stack, including device drivers for the host PC, libraries in OVP and transactors in FPGA, so that designers can efficiently test the entire system from early design iterations down to the final system validation with real-world test benches, with the minimum possible effort. The

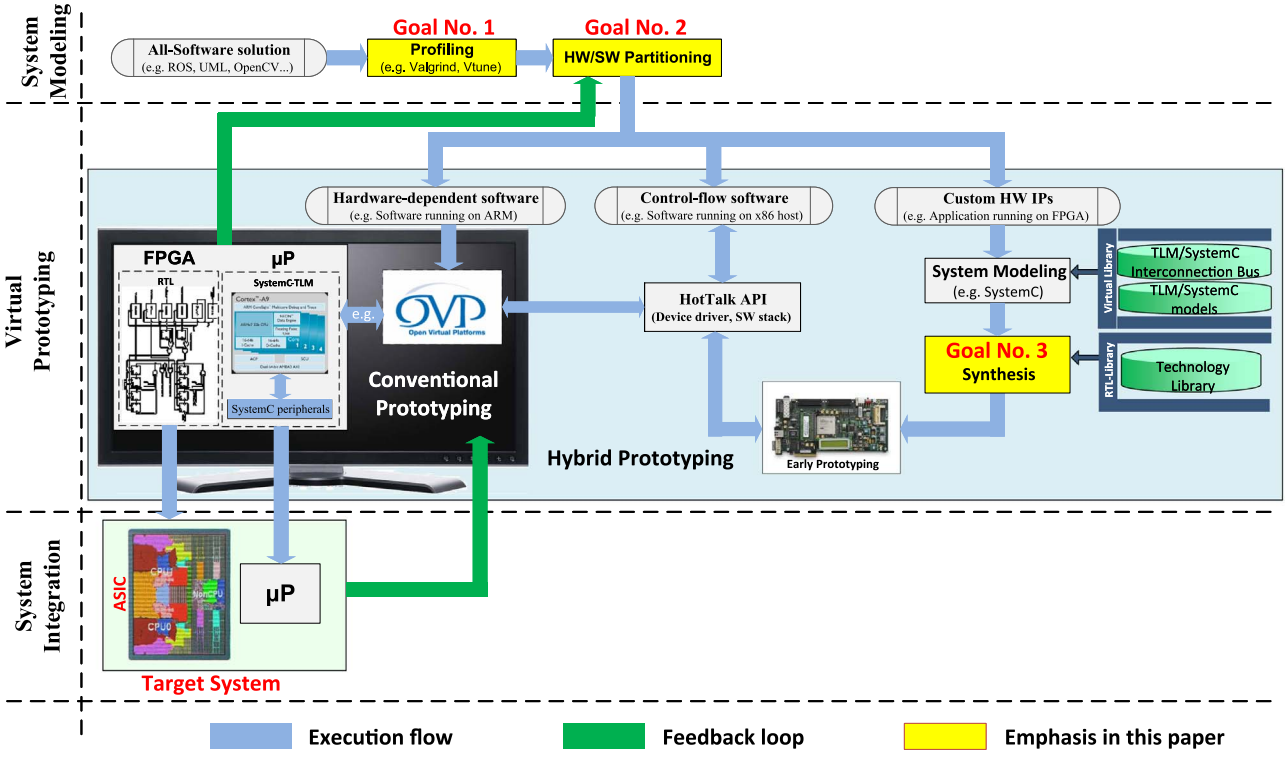

Fig. 1. A hybrid platform for virtual prototyping.