Science of Computer Programming 96 (2014) 156–174

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Model-driven toolset for embedded reconfigurable cores: Flexible prototyping and software-like debugging

Loïc Lagadec[a,b,*], Ciprian Teodorov[a,b], Jean-Christophe Le Lann[a,b], Damien Picard, Erwan Fabiani[a,c]

[a] Lab-STICC MOCS, CNRS 6285, France
[b] ENSTA-Bretagne, France
[c] Université de Bretagne Occidentale, France

HIGHLIGHTS

- We model embedded reconfigurable cores.
- We consider modeling the configuration plane.
- We generate a fully-featured prototype.
- We offer an object-oriented view of the prototype.
- We support high-level debugging through observability, traceability and controllability.

ARTICLE INFO

Article history:
Received 29 October 2012
Received in revised form 14 February 2014
Accepted 17 February 2014
Available online 4 March 2014

Keywords:
Software engineering
Model-driven
Debugging
Reconfigurable computing
System-on-Chip

ABSTRACT

Improvements in system cost, size, performance, power dissipation, and design turnaround time are the key benefits offered by System-on-Chip designs. However they come at the cost of an increased complexity and long development cycles. Integrating reconfigurable cores offers a way to increase their flexibility and lifespan. However the integration of embedded reconfigurable units poses a number of unique challenges in terms of design-space exploration and system exploitation. Over the last few years, model-driven engineering has become one of the most promising methodologies for tackling such challenging software problems.

This paper presents Biniou, a model-driven toolset for embedded reconfigurable core modeling. Biniou is a major step ahead of the Madeo framework that was one of the rare non-commercial environments targeting reconfigurable design automation. In Biniou, the design space is broadened with (re-)configuration modeling aspects, and the exploitation tools are enhanced through the use of multi-level simulation and high-level debugging.

These advancements are illustrated through a case-study focused on the design-space exploration of a coarse-grained reconfigurable architecture and through an examination of the integration of the debug-specific features into the framework. The main benefits of the presented toolset are: efficient domain-space exploration (validation), software design-kit generation (usability), software-like debug facilities (verification).

* Corresponding author.

http://dx.doi.org/10.1016/j.scico.2014.02.015
0167-6423/© 2014 Elsevier B.V. All rights reserved.

1. Introduction

The tremendous evolution pace of the semiconductor industry, has enabled unprecedented advances in the ways we see computer system design. Today's integrated circuits (IC) are more complex and heterogeneous than ever. Being composed of several processors (μP), digital signal processors (DSP), communication networks (NoC), and complex memory hierarchies, they have attained full system functionality into a single chip, and are referred to as System-on-Chip (SoC). These SoCs have the potential to offer several benefits, including improvements in system cost, size, performance, power dissipation, and design turnaround time. Unfortunately, the SoC design process, which schematically consists of successive refinements of an abstract specification towards the physical realization, is becoming an increasingly difficult task.

In the early 1990s the term hardware/software co-design appeared to describe a confluence of problems in IC system development. The prefix *co* is used to denote a joint or partnered development – as opposed to the coincidental development at the same time – of both hardware and software. The key idea behind hardware/software co-design is to find the right trade-off between speed of hardware execution and generality of software, while considering the costs incurred. The main problem in hardware/software co-design is how to design an embedded system that contains both hardware execution (through application-specific ICs[1]) and software execution support (through μP). A critical decision that has a wide effect on overall system cost is how to partition the system into its hardware and software components. A mistake made in this decision can add significant delay and cost to the design process, since correction in this context implies reworking the entire design. The longer this irrevocable decision can be delayed, the better the chance to keep overall system costs to a minimum.

Historically, reconfigurable devices, such as Field-Programmable Gate Arrays (FPGA), provided a solution to this conundrum by offering support for late hardware customization, which – much like software late binding – enables early availability, reuse and tailoring. Compared to ASIC, the agility of reconfigurable architectures comes from the ability to reallocate resources (potentially in the field, partially, and on the fly) to form a new circuit. This favors fast prototyping and early circuit implementation, sometimes even prior to full specification availability.

Today, with the emergence of the fabless[2] business model, new competitors have entered the race of reconfigurable platforms [1–3]. The field of reconfigurable computing has morphed, and besides mainstream FPGA vendors such as Xilinx and Altera, the fabless solution providers offer more specialized reconfigurable devices, ranging from coarse-grained cores for DSP [4], to embeddable units [5].

Embedding reconfigurable cores into an SoC offers a tradeoff between the area overhead and the flexibility of the system that must be carefully considered. Conceptually, the area overhead can be estimated throughout the design cycle by physical synthesis tools (designing the reconfigurable device) while the flexibility is scored using applicative synthesis tools (performing the resource allocation in order to map a portion of the application to the reconfigurable unit).

Unfortunately, traditional solutions require a large amount of manual tuning during the physical synthesis and remain bounded to specific tools during applicative synthesis, reducing the ability to explore new architectural options. The loss of effectiveness of methods and tools to address real hardware over the increasing hardware complexity is referred to as the productivity gap. This grows with every new technological evolution and brings new challenges to the designers [6]. In the scope of SoC design, this trend concerns both architecture and software design. This prohibits short development cycles, and renders the design space exploration (DSE) unaffordable.

To address these issues, in this paper, we introduce an open model-driven toolset for the design of embedded reconfigurable units including generation of both hardware prototypes and their specific SDKs (exploitation tools).[3] We rely on the Madeo framework mixed ADL which provides architectural DSE and retargetable place & route tools [7]. In addition to architectural DSE features (such as sizing LUTs, arithmetic operators, etc.) the Biniou toolset also enables DSE for partial and multi-context dynamic reconfiguration modes, thus widening the design space. Moreover, to facilitate agile development, and to ease exploitation our solution tightly integrates with an innovative software-like debugging infrastructure, named RedPill, which offers high-level signal traceability features and control over the hardware execution throughout all circuit synthesis stages – from the application to the bitstream, and more.

The creation of an integrated toolset targeting design-space exploration of embedded reconfigurable units, such as Biniou, is a complex undertaking requiring flexible and extensible solutions for complex problem-spaces. The solution presented in this paper relies at its core on many years of proactive research and development in the FPGA design-automation field, notably around the Madeo infrastructure [7]. However, addressing the highly-dynamic and challenging field of embedded reconfigurable units, in the context of SoC design, opens the door to a whole new set of problems, notably in terms of tool adaptability, integration, and synergy. Besides providing an answer to these technical and methodological issues, the main contributions of this work are:

- *Configuration plane modeling*. In the context of design-space exploration of embedded reconfigurable architectures, besides computational architecture modeling, the design of the configuration infrastructure plays a very important role

[1] Application-specific ICs are commonly named ASICs.
[2] Fabless manufacturing is the design and sale of hardware devices and semiconductor chips while outsourcing the fabrication to a specialized manufacturer, called a foundry.
[3] In this study we will use SDK, or exploitation flow (tools) to refer to the set of tools aimed at mapping a given application on a target reconfigurable architecture.