TURNUS: a Design Exploration Framework for Dataflow System Design

Brunet, Simone Casale; Mattavelli, Marco; Janneck, Jörn

Published in:
IEEE International Symposium on Circuits and Systems

2013

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Abstract—While research on the design of heterogeneous concurrent systems has a long and rich history, a unified design methodology and tool support has not emerged so far, and thus the creation of such systems remains a difficult, time-consuming and error-prone process. The absence of principled support for system evaluation and optimization at high abstraction levels makes the quality of the resulting implementation highly dependent on the experience or prejudices of the designer. This is particularly critical when the combinatorial explosion of design parameters overwhelms available optimization tools. In this work we address these matters by presenting a unified design exploration framework suitable for a wide range of different target platforms. The design is unified and implemented at high level by using a standard dataflow language, while the target platform is described using the IP-XACT standard. This facilitates different design space heuristics that guide the designer during validation and optimization stages without requiring low-level implementations of parts of the application. Our framework currently yields exploration and optimization results in terms of application throughput and buffer size dimensioning, although other co-exploration and optimization heuristics are available.

I. INTRODUCTION

The research on the co-design of hardware/software systems spans a wide range of topics, from system specification down to low-level code synthesis and optimization. The current absence of a general and integrated design methodology makes the implementation of heterogeneous parallel systems a time-consuming and error-prone engineering process. Our approach uses dataflow (and the dataflow language CAL [3]) as the basis for a unified description of parallel and heterogeneous systems. This makes it possible to define system behavior platform-independently, and to successively synthesize low-level code for both SW or HW processing components (i.e. C/C++, HDL) [4]. The development of the software tool suite TURNUS was motivated be the need for a computer-aided co-exploration framework for the large range of important design parameters. The tools guide designers during the co-exploration and optimization process.

II. THE TURNUS CO-EXPLORATION ENVIRONMENT

TURNUS [1] is integrated in the ECLIPSE environment as depicted in Fig. 1a. The design behavior is specified by means of a dataflow programming language. The target platform architecture is specified by using the IP-XACT standard model description. It can be a composition of heterogeneous processing elements (e.g. CPUs, GPGPUs, FPGAs), memories and interconnect fabric (e.g. Ethernet, PCI-Express). The co-exploration process requires an initial abstract design simulation. During this step, the functional design behavior is validated without information about the implementation platform. The TURNUS simulator extends the basic functionality of the ORCC dataflow CAL simulator [2] by providing code profiling (e.g. executed operators counter, instruction counter) and can generate a scheduler-independent execution trace containing only the dependencies between small fragments of an application’s execution. This dependency structure is stored in a compressed file. Profiling information can be made more accurate and reflective of specific platform properties by importing results from third-party profilers (e.g. ModelSim, GNU gprof). A set of heuristics (e.g. critical path analysis, buffer minimization and optimization, partitioning and scheduling analyses [5]) are available for design space exploration and optimization. As an example we shown how the design throughput and its buffer configuration can be optimized. First the trace dependencies are evaluated using a critical path analysis by supposing an unbounded size buffer configuration. The results outline the most serial part of the program and the designer is guided toward a first design re-factorering. The second step is to find a close-to-minimal buffer size configuration that guarantees a deadlock free execution of the trace. This kind of optimization is important in order to reduce memory usage on modern FPGA or many-core systems in which there are severe limitations on embedded memory. This initial configuration is used for evaluating the maximum critical path length, effectively defining an upper bound for achievable performance as depicted in the example of Fig. 1b. Iterative simulations of the execution trace are then performed taking into account the mapping onto the different processing elements and interconnect fabrics. After that, an iterative algorithm enables the designer to find a trade-off between buffer size configuration and an optimal or near-optimal design throughput. Examples of results of throughput-buffer optimizations for standard video codec implementations are illustrated during the demonstration.

REFERENCES