View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Ghent University Academic Bibliography

The FASTER vision for designing dynamically reconfigurable systems

M. D. Santambrogio *, C. Pilato*, D. Pnevmatikatos †, K. Papadimitriou†, D. Stroobandt‡, D. Sciuto*

* Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano
† Foundation for Research and Technology - Hellas
‡ Computer Systems Lab, Ghent University

***Abstract*— Extending product functionality and lifetime requires constant addition of new features to satisfy the growing customer needs and the evolving market and technology trends. software component adaptivity is straightforward but not enough: recent products include hardware accelerators for reasons of performance and power efficiency that also need to adapt to new requirements. Reconfigurable logic allows the definition of new functions to be implemented in dynamically instantiated hardware units, combining adaptivity with hardware speed and efficiency. For the Intrusion Detection System example, new rules can be hardcoded into the reconfigurable logic, achieving high performance, while providing the necessary adaptivity to new threats.**

The FASTER (Facilitating Analysis and Synthesis Technologies for Effective Reconfiguration) project aims at introducing a complete methodology to allow designers to easily implement a system specification on a platform combining a general purpose processor with multiple accelerators running on an FPGA, taking as input a high-level description and fully exploiting, both at design- and run-time, the capabilities of partial dynamic reconfiguration. The FASTER project will facilitate the use of reconfigurable hardware by providing a complete methodology that enables designers to easily implement and verify applications on platforms with general-purpose processors and acceleration modules implemented in the latest reconfigurable technology.

I. Introduction

In an ever-changing world there is an increasing demand for computing systems that will be able to adapt to their environment or to meet new application demands. Adaptation by means of changing the software running on processors is not always adequate. For example, many embedded applications require hardware acceleration and it is also necessary for the hardware to be able to adapt to application changes. Reconfigurable hardware can be a key enabler for these systems. Hardware supported adaptation mechanisms provide a cost effective way of coping with changing requirements. This is in addition to providing the flexibility needed to allow functionalities to be defined and easily added or substituted after a system has been manufactured and is already deployed. However, the ability to take relevant reconfiguration issues into account from the initial system specification to the final system design and the mechanisms required to support this additional functionality at runtime are currently lacking.

The FASTER project (Facilitating Analysis and Synthesis Technologies for Effective Reconfiguration) [1] aims at introducing a complete methodology allowing designers to easily implement and verify a system specification on a platform that includes one or more general purpose processor(s) combined with multiple acceleration modules implemented on one or multiple reconfigurable devices. Our goal is that for selected application domains, the envisioned toolchain will be able to reduce the design and verification time of complex reconfigurable systems by at least 20%, providing additional novel verification features that are not available in existing tool flows. In terms of performance, for the selected application domains, the toolchain could be used to achieve the same performance with up to 50% cost reduction compared to programmable SoC based approaches, or exceed the performance by up to a factor of 2 for a fixed power consumption envelope. The FASTER tool-chain is an integrated semi-automatic framework that can assist the designer in the development of reconfigurable heterogeneous MPSoC systems. It starts from a C-based description of one or more applications to be implemented and minimal information about the target device/technology. Then, it allows a progressive refinement of both the designed *applications* and the hardware *architecture* by offering the possibility of integrating different algorithms and tools to tackle the different aspects of the design. Also, it allows specifying decisions in an interactive environment by hiding most of the implementations details to the designer. At its output it generates the necessary artifacts for the early validation of the produced system, allowing the integration with commercial tools for the subsequent steps (e.g. simulation or synthesis).

In the remaining of this paper, Section II describes the overall problem FASTER is aiming to solve and overviews related works with similar goals. Section III discusses the FASTER project objectives, while Section IV focuses on the description of the offline analysis and the framework envisioned and implemented in FASTER for desigining dynamically reconfigurable systems. Finally, Section V wraps up the authors conclusions and presents future research directions.

II. Context definition

Current and future computing systems increasingly require to be flexible and extensible even after the system is operational, in order to cope with changing user requirements,

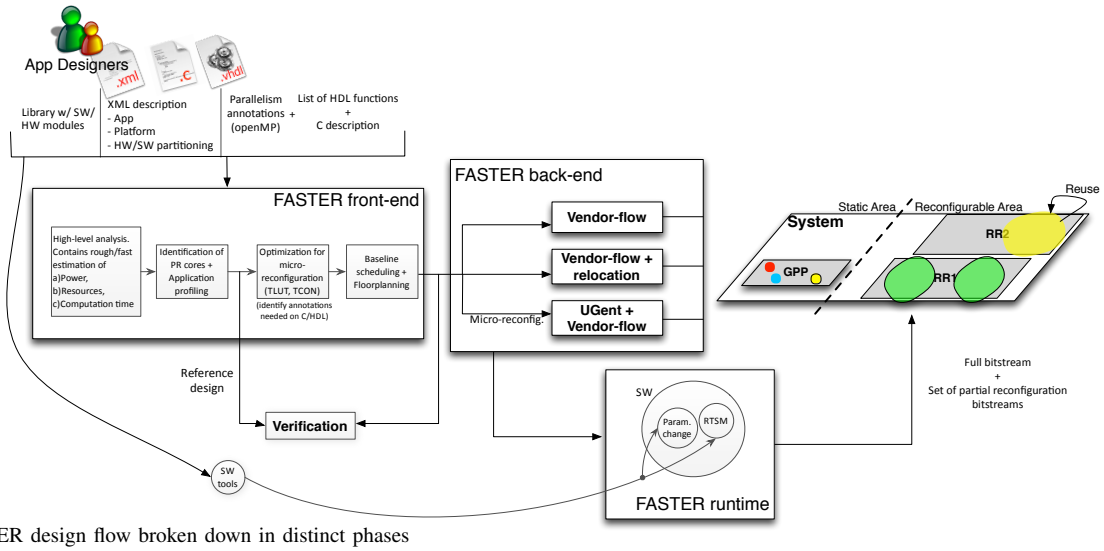

Fig. 1. FASTER design flow broken down in distinct phases

improvements in system features, changing protocol and data-coding standards, evolving demands for support of different user applications, and newly emerging applications in communication, computing and consumer electronics. The FASTER project [1] aims at exploiting the capabilities of dynamic reconfiguration, both at design-time and run-time, by taking as input a high-level description of the application. Previous efforts and EU projects such as MORPHEUS [2], hArtes [3], Reflect [4], S4, Acotes [5], and Andres [6] have dealt with making a tool-chain and addressed similar issues as FASTER, but they focused more on the system-level or architectural aspects of reconfiguration. In fact, they did not explicitly emphasize on the design and runtime aspects of dynamic reconfiguration. This is exactly where FASTER intends to contribute: to introduce partial and dynamic reconfiguration from the initial design of the system all the way to its runtime use. Therefore, we have to define the design concepts capturing both parallelism and reconfigurability as essential system properties and provide efficient and transparent-to-the-user runtime support for partial and dynamic reconfiguration. Reconfigurability can increase both the design time and runtime complexity, and thus an adequate framework for verification and analysis becomes a necessity; the formulation of such a framework constitutes one of the targets of the project. FASTER is focused on exploiting reconfiguration at the hardware description level, so the front-end of these approaches can be used to translate languages such as C into task graphs, and can then be processed with the FASTER scheduling mechanism. Furthermore, we assume that applications are developed or ported by taking into account both parallelism and the potential for dynamic reconfigurability. The FASTER tool-chain will provide the capability to identify the reconfigurability characteristics of an application, determine the best implementation options, and verify the resulting implementation in order to take advantage of these features from the beginning, rather than introducing reconfiguration late in the mapping process, or merely by transforming the code.

Within this context, FASTER project proposes a novel approach for computing system design, focusing on dynamically and partially reconfigurable FPGA-based architectures. It binds classical computing system design and hardware reconfiguration to focus the design attention on the entire platform characterization. Figure 1 illustrates the overall FASTER tool chain. FASTER embraces the development of new tools and the identification and formalization of new models and design methodologies to represent and implement efficient computing systems. The tools will cope with the application requirements as well as the development of area allocation and management algorithms for efficient runtime support of dynamic reconfiguration. FASTER project will extend the design options beyond the region-based reconfiguration design [7] by incorporating micro-reconfiguration [8] in the developed tool set. Micro-reconfiguration can be fast and configuration generation can take place at run-time, for instance triggered by the change of some parameters [9]. The ability to handle both types of reconfiguration opens up a new range of possibilities for run-time reconfiguration, which can offer a versatile framework for serving the design of applications targeting reconfigurable hardware.

III. FASTER Objectives

The project enhances the following five aspects of the design of computing systems:

The *first objective* is to include reconfigurability as an explicit design concept. Starting from the high-level description of the system and its application, the proposed approach will provide a set of different metrics and profiling information along with communication and dependency graphs, oriented to support system-level synthesis onto a hardware platform supporting dynamic reconfiguration.

The *second objective* is to provide the methods and tools for including run-time reconfiguration in every aspect of the design methodology. The methods and tools designed in FASTER will allow for the efficient and transparent use of partial and dynamic reconfiguration at different time scales,

supporting also both explicit parallelism in the application specification and platforms combining multiple software processors with reconfigurable logic. This will lead to a scenario that will enable efficient interfacing between the parallel software and the reconfigurable hardware components.

The *third objective* of the project is to provide an effective framework for analysis, synthesis, and verification to guarantee that the final implementation corresponds to the application requirements and system specifications. Our focus is on developing an integrated tool-chain that supports the verification of both static and dynamic portions of the reconfigurable design.

Our *fourth objective* is to provide efficient and developer/user transparent runtime support for partial and dynamic reconfiguration. Assuming a partially reconfigurable system - either with a single or multiple FPGAs -, we will develop a runtime system that can efficiently handle the online scheduling and placement of reconfigurable system components, using dynamically adaptive schemes to optimize the system operation based on different functional and non-functional requirements defined by the user.

Finally, the *fifth objective* is to provide seamless integration of parallelism and reconfigurability in the specification, irrespective of whether it applies to software or hardware components. The flow will interface the parallel software to the hardware components, and to the runtime manager responsible for partial and dynamic reconfiguration.

IV. FASTER design platform for dynamically reconfigurable systems

The main goal is to analyze each application and define its components, estimate its execution time on the target platform, identify the right level of reconfigurability for it (none, region-based or micro-reconfigurable), the power constraints, the floorplanning constraints (size and shape), the placement requirements (type of resources and connectivity among modules) on the target platform and the baseline schedule for its execution. We consider both static and dynamic reconfiguration and assume a partially reconfigurable system, either in a single device or by using multiple FPGAs. FPGAs add two new aspects to classical VLSI design: *Resource Heterogeneity*, the silicon die of FPGAs generally provides several kinds of resources (e.g., programmable logic cells, memories, multipliers, DSPs and so on); and *Reconfigurability*, the architecture, or the application implemented on the FPGA, may change at runtime a subset of its modules in order to modify its own behavior, without disrupting the operation of the rest circuit. The ideal place in the design flow where these issues are best tackled is the floorplanning phase. This is where heterogeneity can be taken into account and this is where reconfigurability introduces time as a new variable inside the formulation. Once the level of reconfiguration is defined for each application part, the static modules fit for region-based reconfiguration and the micro-reconfigurable modules both need to be implemented. Starting from the original high-level specification, the FASTER tool-chain goes through four different phases to obtain to the partitioned system (i.e., cores which are ready to be scheduled for configuration and execution). Therefore, the output will be the baseline schedule and the description of the application subdivided into components, ready for synthesis using any commercially available tools for the targeted technology.

A. High-level analysis

The purpose of high-level analysis is to provide an analytical model of a reconfigurable design that relates its application attributes to possible implementation parameters, from which metrics such as area, performance and power consumption can be estimated. Such parameters can be extracted from a high-level descriptions such as C or, if the designer will provide these descriptions, in VHDL. It is often advisable to construct an analytical model prior to the detailed design, since the model cand help to identify promising implementations and can be used to guide the design process early on. During the implementation phase, the assumptions and estimations can be verified. Updated values for the implementation parameters are used to verify the overall design goal and to check if critical design constraints are met. The analytical model can be automated by capturing the equations in a spreadsheet format. One can then explore how application attributes and implementation parameters influence performance, area and power consumption. It is also possible to use the analysis to identify bottlenecks in the architecture, whose elimination by having, for instance, a faster reconfiguration time could lead to significant performance improvement. The output of high-level analysis is an estimate of how a design with a given set of application attributes and implementation parameters performs. However, it is not the goal of this task to produce code or to analyze code automatically. Rather, the estimation is intended to guide the identification of features of promising designs.

B. Profiling of applications for identifying region-based, micro-reconfigurable, and static cores

The input of this phase is the original specification, whose analysis results in cores, i.e. groups of operations that compose configurable modules, with optimal sizes. The identification of the region-based reconfigurable cores is performed by analyzing the Control Data Flow Graph of the input application, and trying to identify isomorphic subgraphs in the graph. The choice of finding isomorphic subgraphs is related to the possibility of reusing these components without reconfiguration, thus hiding/minimizing reconfiguration time. Within this context, if distinct components, having the same implementation, are mapped onto the same reconfigurable core, we can execute them on the same physically implemented module with a single initial reconfiguration. Obviously, at the end we have to identify a number of components, isomorphic or not, that cover the entire graph in order to restructure the entire application into a set of interconnected components. Our first step is the core identification phase. The input of this step is the original specification, whose analysis results in cores, i.e. groups of operations that, reconfigured together as configurable modules, have optimal sizes. The second part of the task is the Partitioning phase. Using the previously computed set of modules as its input, this phase produces a

set of feasible covers of the original graph of the specification, following a given policy.

C. Optimization of applications for micro-reconfigurable core implementation

Since every change of a parameter value in the parameterizable micro-reconfiguration results in a reconfiguration of (part of) the implementation, the number of parameter changes should be kept as low as possible. This requires a higher parameter value *locality* in time. In this project, we investigate the introduction of parameters in such a way that the overall implementation can be optimized. In this respect, we will also investigate multi-mode applications where the different modes are similar (but not exactly the same) and investigate how such applications can best be represented to benefit from micro- reconfiguration. In parallel, we will investigate first what features of an application will beneficially map to a micro-reconfigurable configuration. Since such a configuration depends on the number of parameters and the frequency with which their values change, this will be the main focus of our profiling step. We also look into which constructs are better suited for mapping on a TLUT (Tunable LUT) and which ones are better suited for mapping to a TCON (Tunable Connection). For both concepts, an evaluation is needed of the benefits and drawbacks and a trade-off has to be made for several application features to see which concept should be focused on. Finally, we profile applications under consideration to recognize the features that benefit from a micro-reconfigurable implementation. The main task here will be to locate parameters in the application changing their value infrequently enough to benefit from micro-reconfiguration. Finally, the generated graph, called Task Dependency Graph is provided as input to the following phase, i.e. the scheduling phase which generates the complete schedule, taking also into consideration both kinds of reconfigurable modules.

D. Compile-time baseline scheduling

Given the information computed in the previous tasks, an integration between a heuristic reconfiguration-aware scheduler and a floorplacer algorithm will produce the baseline schedule of the tasks on the target platform. This scheduler is a reconfiguration-aware scheduler for dynamically partially reconfigurable architectures that can also manage static reconfiguration and multi FPGAs. It schedules the modules according to the actual hardware composition and availability: given the actual spatial constraints defined by the floorplacer, it schedules the applications trying to reach the minimum scheduling time. The function to be optimized can be chosen based on the user constraints. Output of this task is the list of timings for each module, i.e. start time, and reconfiguration time. However, a static schedule may not be possible for all of the targeted applications. For example, when the execution times of the modules are data-dependent the scheduler will only provide guidelines to the runtime system and the HW scheduler that will manage the system at runtime. A close interaction with the runtime scheduler is therefore foreseen, since the runtime scheduler will receive as input the results produced during the generation phase.

V. Conclusions

The FASTER project implements a complete methodology to allow designers to easily implement and verify a system specification on a platform that includes one or more general purpose processor(s) combined with multiple acceleration modules implemented on one or multiple reconfigurable devices. The FASTER tool-chain accepts input that can be in HDL or C whose initial decomposition could be described with existing formalisms (e.g. OpenMP) and derive the corresponding task graph. Using new graph-theoretic algorithms we will partition the specification in space and time. Then we will pursue a task-cluster definition of a system specification by detecting recurrent structures in the specification itself. These modules are candidates for reconfiguration, thus saving device resources and reconfiguration time. FASTER supports both region- and micro-reconfiguration, which is a technique that reconfigures very small parts of the device. The ability to handle both types of reconfiguration opens up a new range of application possibilities for run-time reconfiguration, as a much broader time frame for the reconfiguration itself is available and the underlying concepts are different for both types of reconfiguration.

FASTER will also develop techniques for verifying static and dynamic aspects of a reconfigurable design at compile time using symbolic simulation, a powerful verification approach for static designs, and extending it to support both static and dynamic aspects of a reconfigurable design. We will also explore techniques for verifying selected static and dynamic aspects of a reconfigurable design at run time with a small impact on speed, area and power consumption. Finally, FASTER will provide a powerful runtime system that will be able to run on multiple reconfigurable platforms and manage the various aspects of parallelism and adaptivity with the least overhead.

Acknowledgment

This work was supported by the European Commission in the context of FP7 FASTER project (#287804).

References

[1] http://www.fp7-faster.eu/, [Online; accessed May 2012].
[2] http://ce.et.tudelft.nl/DWB/, [Online; accessed May 2012].
[3] http://hartes.org/hArtes/, [Online; accessed March 2012].
[4] http://www.reflect-project.eu/, [Online; accessed March 2012].
[5] http://www.hitech-projects.com/euprojects/ACOTES/, [Online; accessed March 2012].
[6] http://andres.offis.de/, [Online; accessed March 2012].
[7] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, "Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAs (Invited Paper)," in *Proceedings of the IEEE Conference on Field Programmable Logic and Applications (FPL)*, August 2006, pp. 1–6.
[8] K. Bruneel, "Efficient Circuit Specialization for Dynamic Reconfiguration of FPGAs," PhD thesis, Ghent University, 2011.
[9] K. Bruneel and D. Stroobandt, "Automatic generation of run-time parameterizable configurations," in *Proceedings of the IEEE Conference on Field Programmable Logic and Applications (FPL)*, August 2008, pp. 361–366.