DURASE: Generic Environment for Design and Utilization of Reconfigurable Application-Specific Processors Extensions

Kevin Martin, Christophe Wolinski
University of Rennes I, Irisa, Inria
France
{martin,wolinski}@irisa.fr

Krzysztof Kuchcinski
Dept. of Computer Science
Lund University, Sweden
krzysztof.kuchcinski@cs.lth.se

Antoine Floch, François Charot
Inria, Centre Rennes - Bretagne Atlantique
France
{anfloch,charot}@irisa.fr

Abstract—The DURASE system enables automatic synthesis of application specific processor extensions that speed-up the application execution. It carries out also corresponding source code transformations to match the newly synthesized extensions. Finally, the synthesized extensions are tightly connected to the target processor and used through newly created instructions.

I. INTRODUCTION

Our DURASE system enables automatic synthesis of application specific processor extensions that speed-up application’s execution. The system carries out also corresponding source code transformations to match the newly synthesized extensions. Finally, the synthesized extensions are tightly connected to a target processor and used through newly created instructions (see Figure 2 for example of the NIOS II processor and its extension). The design flow adopted in the DURASE system is presented in Figure 1. The input to the DURASE system is an application code written in C, a target processor instruction set and an architecture model. The output is a processor extension and application specific instructions for accessing this extension. The processor extension is built using a merged pattern implementing all the selected computational patterns. Our system generates also the transformed application source code, including application specific instructions.

Our design process involves identification of computational patterns and selection of specific patterns that speed up application execution. The pattern identification and selection are executed in two consecutive steps. In the first step, we explore typical computational patterns and identify the most useful ones for a given application. Our method identifies all computational patterns directly from an application graph satisfying all architectural and technological constraints imposed by target processors and FPGA devices. The considered constraints include a number of inputs and outputs, a number of operators, and a delay of the pattern critical path. Therefore the identified patterns can be well tailored to target processors. The identified computational patterns are then used in the mapping and scheduling step where a subset of patterns is selected for implementation.

The developed DURASE system uses advanced technologies, such as algorithms for graph matching and graph merging [6] together with constraints programming methods. Our system uses also a generic compilation platform GECOS recently extended by polyhedral transformations [2]. The internal representation of the DURASE system is the Hierarchical Conditional Dependency Graph (HCDG) capturing both the application control-flow and data-flow [3], [4]. In the HCDG graph nodes are guarded by boolean conditions and polyhedrons depending on loop indexes and parameters. After data-flow analysis [1], each read statement in a graph has a set of possible source contributions relying on their execution contexts and polyhedral conditions. After specific transformations loops are totally or partially unrolled to generate an HCDG which is an input for the pattern generator.

The architecture model of an ASIP processor with an extended instruction set is considered. Extended instructions implement identified and selected computational patterns and can be executed sequentially with the ASIP core instructions.
our generic simplified architecture is depicted in Figure 2. It is composed of one functionally reconfigurable cell implementing a set of computational patterns (selected by the DURASE system) connected directly to the processor’s data-path. The selected patterns are merged by our merging procedure [6] before synthesis. The cell contains also registers for the case when the generated patterns have more than two inputs and one output (case of the NIOS II). The number of registers and the structure of interconnections are application-dependent.

Figure 3 presents an example corresponding to partially unrolled FIR filter. The figure depicts application graph covered with automatically generated patterns, the set of selected patterns and a corresponding schedule. Matches, executed on the processor and on the processor extension, are grouped separately. The additional cycles needed for the data communication between the processor and its extensions as well as data transfers are also presented.

III. CONCLUSION
The DURASE system is based on the improved and extended version of the UPaK kernel [5] and uses advanced technologies, such as algorithms for graph matching and graph merging together with constraints programming methods. It enables the automatic generation of computational patterns from applications written in C and compiled for target processors under different architectural and technological constraints. It also ensures the optimized scheduling of the applications on the new architectures corresponding to the different architectural models of the processor extensions.

REFERENCES