A Tool Flow for Design Space Exploration of Partially Re-configurable Processors

K. Karuri, A. Chattopadhyay, S. Kraemer
R. Leupers, G. Ascheid, H. Meyr
Integrated Signal Processing Systems,
RWTH Aachen University 52056 Aachen, Germany

Abstract

This demonstration presents a tool flow, based on a specification formalism and assisted by instruction-set synthesis, that greatly simplifies the complex design of partially reconfigurable processors.

1. Introduction

Over the past few years, embedded System-on-Chip (SoC) designs have been relentlessly driven by the continually increasing requirements of *performance and flexibility* of new and complex applications. The emergence of *Application Specific Instruction-set Processors (ASIPs)* as one of the key components of such SoCs can be ascribed to their unique blend of both of these qualities. Due their custom designed nature, ASIPs can combine the programmability/flexibility of general purpose processors with high-throughput and low power consumption. However, the flexibility of an ASIP is only limited to the *soft* changes, i.e. the hardware implementation can not be altered after fabrication. Often this limitation prevents ASIPs from being designed and deployed for a wide range of applications. The kind of *hard* flexibility that can solve this problem, is readily available in fully-reconfigurable architectures [1]. Naturally, *partially re-configurable processors*, that combine the merits of both of these architectural alternatives, are gaining prominence in recent years [2, 3]. These architectures contain an ASIP part and an reconfigurable (i.e. FPGA) part. The added flexibility allows the *re-configurable ASIPs (rASIPs)* to be adapted to a large variety of applications without compromising the performance.

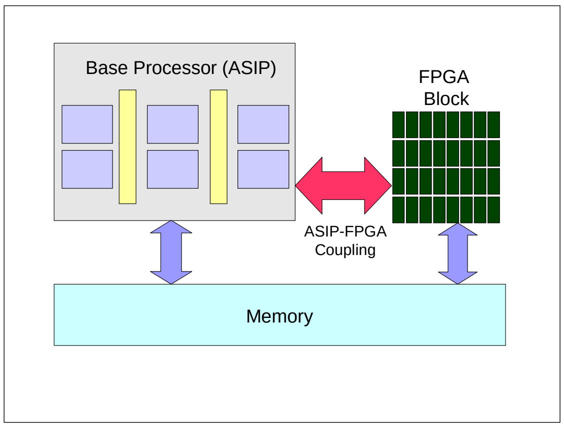


Figure 1. General rASIP Architecture

As can be seen from Figure 1, an rASIP architecture can be broadly divided into three partially overlapping components namely, the *base processor*, the *ASIP-FPGA coupling* and the *FPGA architecture*. Considering the numerous design alternatives for each of these components, it is understandable that the design of rASIP is a demanding task which calls for an efficient design methodology. This demonstration presents a tool flow that simplifies this complex design process by facilitating fast and comprehensive design space exploration in the *PRE-fabrication* phase of an rASIP development, and optimal mapping of various applications in the *POST-fabrication* phase.

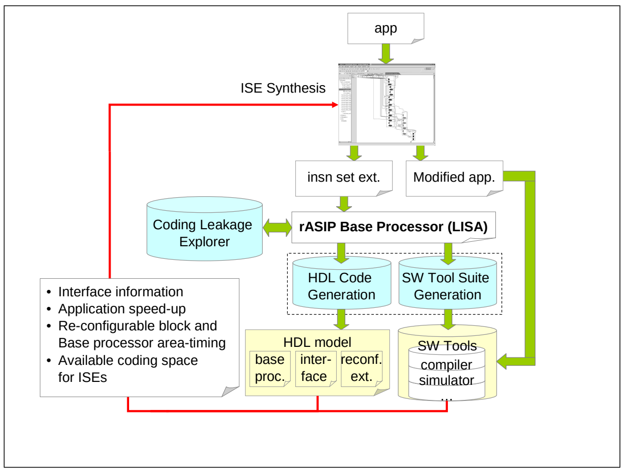
2. Design Flow

The design space exploration and implementation of rASIPs can be naturally sub-divided into two phases, namely PRE-fabrication and POST-fabrication phase - terms which have already been mentioned in section 1. The focus of these two phases are presented in the following.

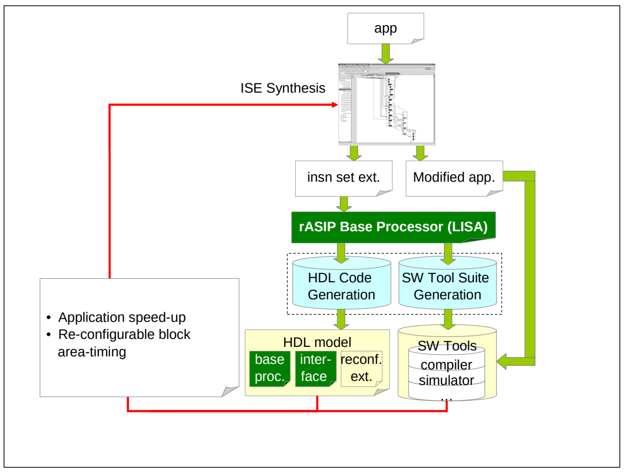
- **PRE-fabrication Design Flow:** This phase of design happens before the rASIP is fabricated. Here, the complete design space is open. The decisions involving all three design sub-spaces (i.e. base processor, processor-FPGA coupling and FPGA architecture) are to be taken in this phase. At the end of this phase, the design is implemented partially on fixed and partially on re-configurable hardware.

- **POST-fabrication Design Flow:** This phase of design happens after the rASIP is fabricated. In this phase, the base processor and the interfacing hardware is fixed. The architecture design space is limited to the possible configurations of the re-configurable block only.

This demonstration presents a design flow that is applicable to both PRE- and POST-fabrication phases of an rASIP design. The design flow is centered around a specification formalism that can encapsulate the description of all three architectural components of an rASIP in a single, unified model. This specification method extends the the *LISA 2.0 Architecture Description Language (ADL)*[4], commonly used to describe ASIPs, by incorporating several new language elements to facilitate description of rASIPs. The required set of software tools (i.e. the compiler, assembler, linker and instruction set simulator) and an RTL model can be automatically generated from such an unified rASIP description for simulation, verification and implementation purposes. Currently, the language extensions allow the designer to place new functional units and resources (such as registers) in the reconfigurable part, and the processor-FPGA interface is inferred from the



(a) PRE-fabrication flow.



(b) POST-fabrication flow. The fixed parts of rASIP design are drawn in dark colour.

Figure 2. rASIP Design Flow

data and control signals needed by these units and resources. The generated HDL for the reconfigurable part can be mapped to any fine-grained, commercially available FPGA. In future, specification and exploration of the reconfigurable architecture – specially coarse-grained embedded FPGAs – is planned.

Another major component of the design flow is an *instruction-set synthesis tool* [5] that can map the computationally intensive parts of an application onto a set of optimal *Instruction Set Extensions (ISEs)*. This tool has been extended to generate LISA description of the identified ISEs which can be easily integrated into an existing rASIP model. Another tool named *coding leakage explorer* attempts to extract available coding spaces for such ISEs from the base processor instruction encoding.

Figure 2.(a) presents the PRE-fabrication design flow. The design space exploration may start with a rudimentary rASIP model (in LISA) which is iteratively refined based on the results of simulation. The unified rASIP description greatly simplifies the process of altering different architectural components and the automatic tool generation process permits easy evaluation of different alternative implementations. In each refinement step, the automatic ISE generation tool is used to synthesize and evaluate different special purpose instructions, and the rASIP micro-architecture is modified to accommodate the most promising ISEs. Such adjustments may include, but are not limited to, the introduction of special purpose registers and fast-memories in the re-configurable part, modification of the base processor-FPGA coupling and decisions on the granularity and configuration of the FPGA architecture.

Figure 2.(b) presents the POST-fabrication design flow which is very similar to the PRE-fabrication design flow in many respects. However, unlike the PRE-fabrication flow, the only changes in the rASIP description can be brought about by mapping application specific ISEs to the re-configurable parts. This is easily achieved by restricting the ISE synthesis tool to generate special instructions under *hard* rASIP architectural constraints.

3. Case Study

The design flow described so far has been applied to a case-study of designing a generic rASIP for private key cryptographic applications The base processor used in this case study was LTRISC – a 32 bit RISC processor with sixteen 32-bit registers. Extensive design space exploration in the PRE-fabrication phase using the *DES*[6] application resulted in several modifications of the base processor and reconfigurable portions. For example, a fast SRAM based scratch-pad was added to the reconfigurable part to give ISEs access to the so called *S-Boxes* (constant tables ubiquitously present in different block-cipher algorithms). These architectural modifications were vindicated in the POST-fabrication phase when experiments were carried out with two similar applications, namely *Blowfish* and *GOST* [6]. Without the architectural changes introduced in the PRE-fabrication phase, the speed-up for these applications were found to be quite low (2.7× and 1.02×, respectively, for Blowfish and GOST). However, after the introduction of the scratch-pad and other related alterations, the speed-up shot upto 3.8× and 1.8× for the respective applications.

4. Conclusion and Future Work

Due to the high flexibility, partially re-configurable processors are attracting significant research interest in today's processor design community. In this demonstration, a specification-driven design framework for rASIPs is presented. The proposed design flow integrates existing and new tools for a seamless design space exploration in both PRE- and POST-fabrication phases. In future, we will concentrate on the design space exploration and implementation of the reconfigurable blocks in the rASIP. Furthermore, the modeling and exploration for dynamically reconfigurable-rASIPs will be targeted.

5. REFERENCES

[1] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk and P. Y. K. Cheung. Reconfigurable Computing: Architectures and Design Methods. *IEE Proceedings. Computers and Digital Techniques*, 152(2):193–207, 2005.

[2] Stretch. *http://www.stretchinc.com.*

[3] B. Mei, A. Lambrechts, D. Verkest, J. Mignolet and R. Lauwereins. Architecture Exploration for a Reconfigurable Architecture Template. In *IEEE Design & Test*, 2005.

[4] CoWare/LISATek. *http://www.coware.com.*

[5] R. Leupers, K. Karuri, S. Kraemer and M. Pandey. A Design Flow for Configurable Embedded Processors based on Optimized Instruction Set Extension Synthesis. In *Design, Automation & Test in Europe (DATE)*, 2006.

[6] Bruce Schneier. *Applied Cryptography.* John Wiley & Sons, 1996.