Micro-Profiler : A Fine-grained Application Profiler for ASIP Design

Current Application Specific Instruction set Processor (ASIP) design methodologies are mostly based on iterative architecture exploration that uses Architecture Description Languages (ADLs) and retargetable software development tools. However, for improved design efficiency, additional pre-architecture exploration tools are required to help narrow-down the huge design space and making coarse-grained Instruction Set Architecture (ISA) decisions before detailed ADL modeling. Extensive application code profiling is the key in such early design stages. Based on a novel code instrumentation technology, we present a *micro-profiling* approach that fills the current gap between source-level and instruction-level profilers and combines their advantages w.r.t. speed and accuracy.

The proposed *micro-profiler* (μP) is part of an advanced ASIP design flow that builds on state-of-the-art ADL based architecture exploration tools like [2, 3] (fig. 1). We assume the application C source code is given, and an ASIP needs to be designed or customized for the application. In the pre-architecture stage, the designer needs to determine key dynamic execution statistics for early decisions on the ISA and the memory subsystem. This includes e.g. operation execution frequencies, cache hit rates, frequently used C data types together with their dynamic min/max values, as well as bit width of arithmetic operands and constants. Obviously, these data are extremely helpful in selecting the right accelerator functional units or designing an initial ISA. The μP determines these data by executing the C application code on the host, after automatic instrumentation.

The collected statistics may be used in two ways. A GUI front end of the μP presents the data to the designer in tabular and graphical form. In this scenario, the μP acts as a stand-alone workbench tool, guiding the designer during development of an initial ADL processor model. A set of *retargetable software tools* such as C compiler, assembler, linker, loader and instruction set simulator are then automatically generated from the ADL model for fine-grained (micro-architecture level) design space exploration. The final hardware implementation is developed (or automatically generated from the ADL model) if all the design constraints are met.

In another (more automated) usage scenario, the profiling data are passed to an ISA synthesis tool. This tool applies an optimization algorithm similar to [1] in order to synthesize a limited number of complex custom machine instructions for highest speedup, based on a *weighted operation execution profile*. It generates ADL code for custom instructions that can be linked to existing ADL processor templates, e.g. a RISC core. This offers a direct path to micro-architecture level exploration and implementation with existing tools.

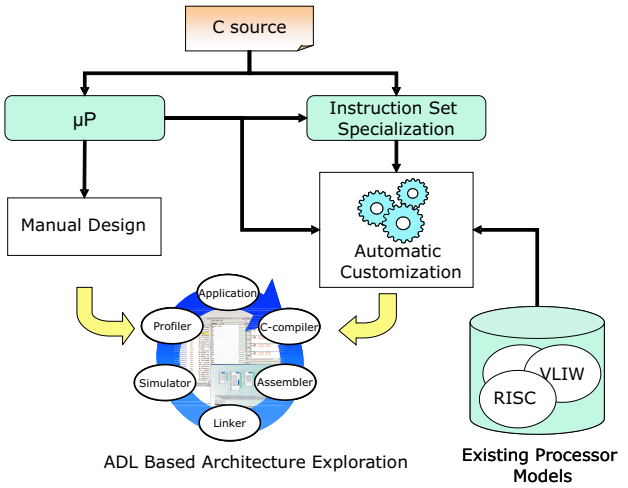

Figure 1: μP in an enhanced ADL based ASIP design flow

1. REFERENCES

[1] K. Atasu, L. Pozzi, P. Ienne: *Automatic Application-Specific Instruction-Set Extensions under Microarchitectural Constraints*, DAC, 2003

[2] LISATek products: http://www.coware.com

[3] A. Halambi, P. Grun, V. Ganesh, A. Khare, N. Dutt A. Nicolau: *EXPRESSION: A Language for Architecture Exploration through Compiler/Simulator Retargetability*, DATE, 1999

.