54

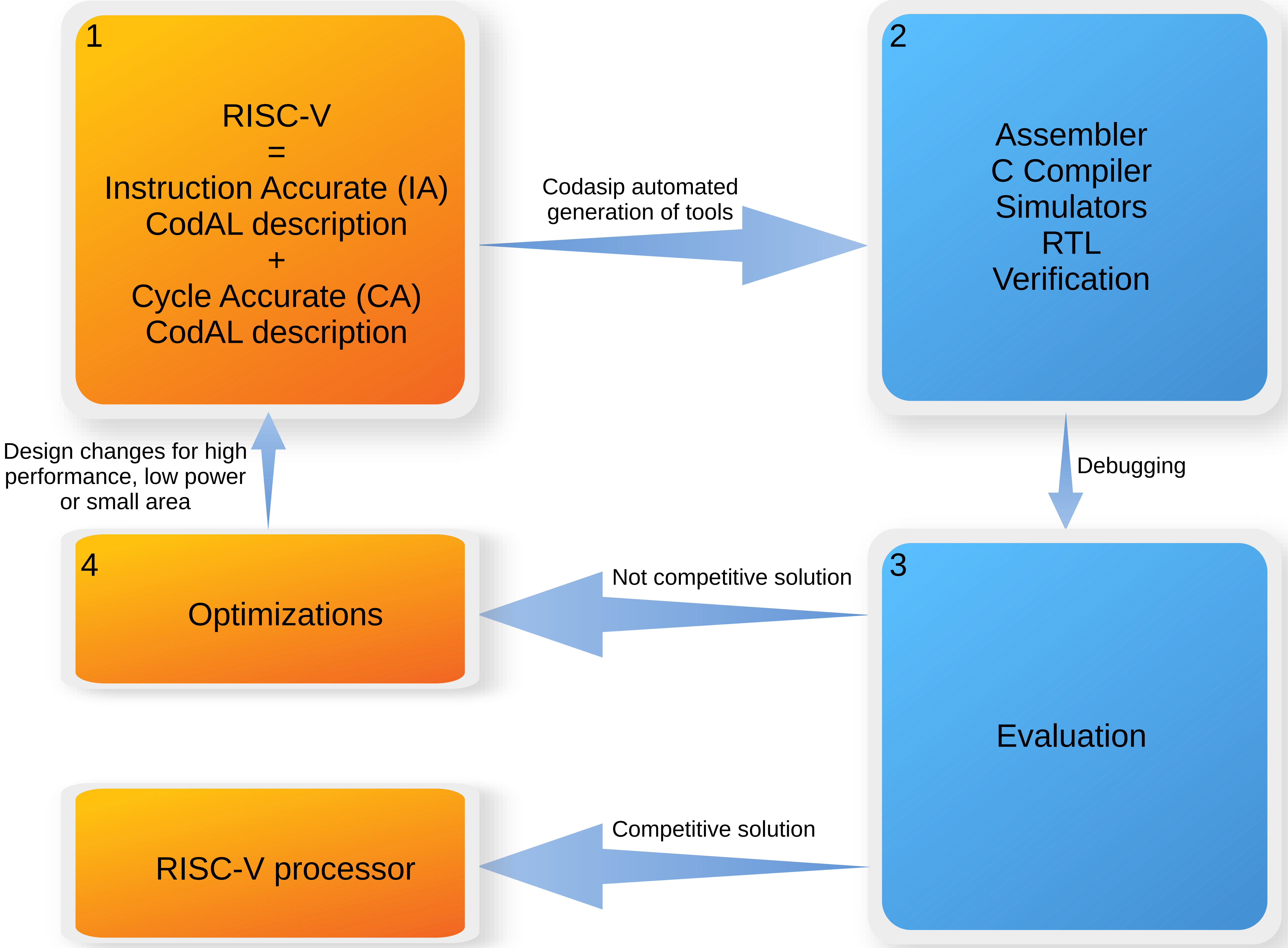
Jiří Barták

Excel@FIT 2016

A Competitive RISC-V Processor Model

High Level Processor Model

Debugging tools and RTL



1

RISC-V
=
Instruction Accurate (IA)
CodAL description
+
Cycle Accurate (CA)
CodAL description

Codasip automated generation of tools

2

Assembler
C Compiler
Simulators
RTL
Verification

Debugging

Design changes for high performance, low power or small area

4

Optimizations

Not competitive solution

3

Evaluation

Competitive solution

RISC-V processor

1

Processor core high-level description in CodAL consists of architectural resources, ISA description, instruction semantics and timing model. This description is divided into two models. Instruction Accurate (IA) model contains description of the instruction set, mainly its semantics. Cycle Accurate (CA) model adds description of architectural resources and timing model. Fully functioning RISC-V CodAL model was created in the matter of weeks as opposed to Berkeley processor that required years to design in the HDL.

2

The Codasip EDA tools provide a solution that considerably speeds up the ASIP design by automated generation of debugging tools, the C compiler, hardware representation and verification environment in order of seconds. IA model is used to generate assembler, LLVM-based C compiler, IA simulator and serves as the reference model for verification. CA model is used to generate CA simulator, RTL description in Verilog or VHDL and verification environment. Generating all tools takes just a few seconds so any change takes effect immediately. IA simulator is used as ISA simulator. It is used to check that all instructions have the right semantics. CA simulator is used to check micro-architecture behavior. The experiments shown that the generated C Compiler is as strong as hand-written one from Berkeley. Generated UVM verification environment is used to check that behavior of generated RTL is the same as the one of IA model.

4

Changes that usually require weeks to process take effect in seconds by generating all tools. Optimizations are the only part that cannot be fully automated. One has to figure out what can be improved and change the model accordingly. Design space can be explored very quickly thanks to automated tool generation so there is no need to consider all the consequences of planned optimization. Almost no time is lost and all the side effects quickly show up. Knowledge gathered from profiling tools allowed me to design new instruction that boosted CRC performance by 34%. There was no need to make changes to compiler for the instruction to be used as it is automatically regenerated and implementation of the added instruction itself took just a few minutes.

3

Codasip provides powerful profiling tool that is able to track usage of instructions, common sequences of instructions, memory access etc. Gathered information can be used to further optimize the processor both in terms of ISA and micro-architecture. All that is needed is to run a series of test applications and result are presented immediately I have used profiling tools while simulating benchmark applications (Dhrystone, Coremark, CRC) to identify performance bottlenecks.