ASIP Design in the Lissom Project

Adam Husár, Zdeněk Přikryl, Karel Masařík, Tomáš Hruška
{ihusar, iprikryl, masarik, hruska}@fit.vutbr.cz,
Brno University of Technology, Faculty of Information Technology, CZ

1 Introduction

- Electronic System Level (ESL) design methodologies use usually one general-purpose core together with specialized application-specific instruction set (ASIP) cores that accelerate computation
- SoC (System on Chip) or MPSoC (Multi-Processor SoC) design is expensive and time-consuming without specialized tools
- The Lissom project is focused on development of a language ISAC for multi-core processor description and on a set of tools that automatically generate tool-chain, simulators and hardware description that accelerate MPSoC design

2 Design Space Exploration

- Design space exploration (DSE) [1] is a process of search for optimal architecture for a specific application
- Automatic toolchain, simulator, and hardware generation allows fast DSE
- Fast DSE saves more time for testing and verification of the final system

3 Project Lissom

- The goal of the Lissom project is to provide a development environment for single- and multi-core processor design
- Specialized architecture description language is used to desribe processor architecture and microarchitecture
- Tools that take ISAC model as input and automatically generate C language compiler, assembler, linker, different types of simulators and synthesizable hardware description were implemented
- Graphical user interface based on Eclipse is provided

Processor	Model type*	Model source lines	Auxiliary C code lines
MIPS	A	2800	1300
ADOP	M	3700	1400
ARMv5	A	1200	1100
PicoBlaze	A	630	300
Chili II	M	2650	960
Chili III	A	1790	1170
VEX	A	1400	450
8051**	M	5000	1000
TI MSP430	A	2000	500
MicroBlaze	M	3300	1800

MIPS32 is in version Release 1, with floating-point instructions and Release 1 DSP Application Specific Extension instructions,
ADOP is an experimental processor, also was manufactured with 350nm process node technology from automatically generated hardware description obtained by using Lissom tools,
Chili II and III are VLIW DSP processors designed by OnDemand Microelectronics
*A – architectural (instruction-accurate), M – microarchitectural (cycle-accurate)
**Model 8051 includes several peripherals

Fig. 1: Complete processor models described using the ISAC language

4 ISAC Language

- ISAC is a mixed architecture description language, originally based on the LISA [2] language
- Allows to create models on two levels of accuracy: instruction-accurate model (only instruction set with simplified behavior) and cycle-accurate (with detailed information about microarchitecture)
- Instruction set is described using operations, where each operation can have its syntax, binary coding and behavior described.

```
//processor resources like registers and memories
RESOURCES {
   REGISTER bit[32] regs;
   //…
}

//general purpose registers description
OPERATION reg REPRESENTS regs {
   //register names in format $0 - $31
   ASSEMBLER { "$" ~ regnum=#U };
   //store register number as a 5-bit binary value
   CODING { regnum=0bx[5] };
   //return register number value
   EXPRESSION { regnum; };
}

OPERATION opc_add {
   ASSEMBLER { "ADD" };
   CODING { 0b100000 };
   EXPRESSION { OPC_ADD; }; //OPC_ADD is a constant
}
// mnemonics and operation code for the SUB instruction
// is described similarly as the operation code for the ADD
// instruction

//as opc can be used either opc_add or opc_sub
GROUP opc = opc_add, opc_sub;

OPERATION instr_arithm {
//used instances of operations and groups
INSTANCE reg ALIAS {rd, rs, rt}; INSTANCE opc;

//textual representation
ASSEMBLER { opc rd "," rs "," rt };

//binary representation
CODING { 0b000000 rs rt rd opc };

//instruction's behavior described using C language
BEHAVIOR {
   switch (opc) {
      case OPC_ADD: regs[rd] = regs[rs] + regs[rt];
      case OPC_SUB: regs[rd] = regs[rs] - regs[rt];
   }
   };
}
```

Fig. 2: Description of instructions ADD and SUB in the ISAC language

5 Tools for processor programming

- We provide tools for low level processor programming, such as assembler, linker, archiver, disassembler and several tools for object file format conversions
- C compiler is based on the LLVM platform, instruction semantics usable for compiler generation can be automatically extracted from the ISAC model, compilers for MIPS and ADOP architectures were generated from the extracted information [3]

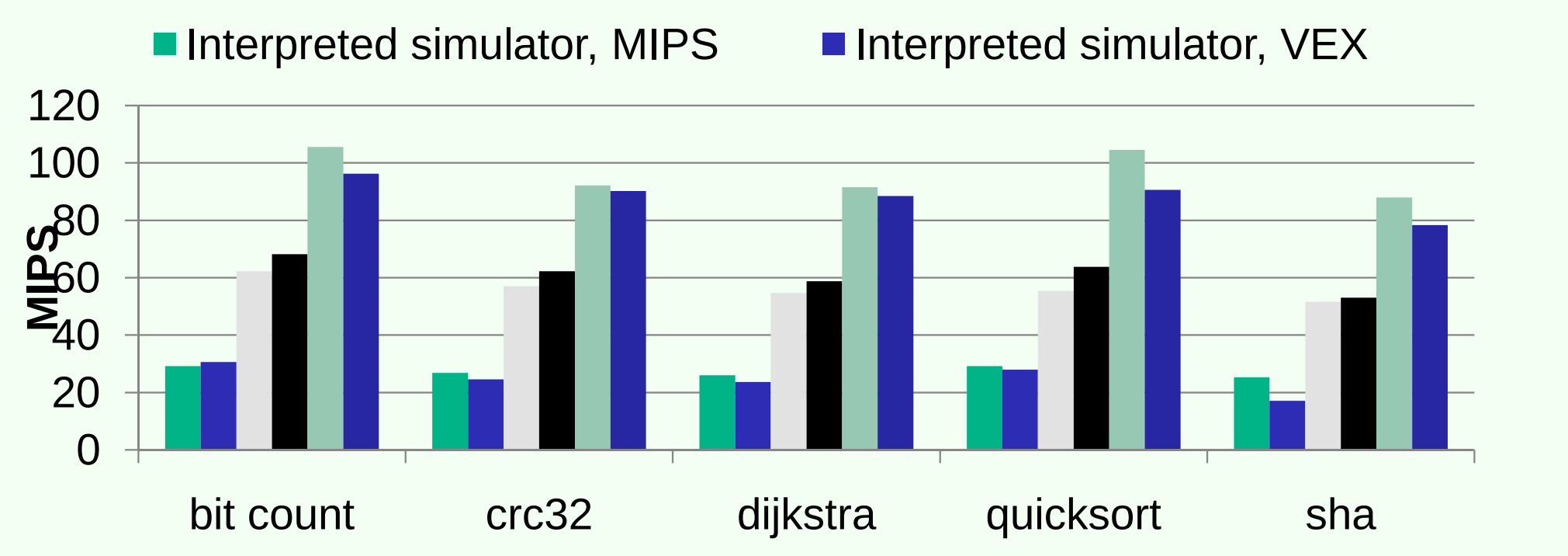

Fig. 3: Simulation speeds of the provided types of simulators of the MIPS and VEX architectures

6 Tools for single- and multi-processor system simulation

- Several types of simulators, which can be used in different phases of DSE, are provided
- The basic type is interpreted simulator, further we provide three types of compiled simulators and an RTL (register-transfer level) simulator
- Simulation of multi-core processors, is also possible, each processor core is represented as an independent simulator
- All types of simulators are based on formal models ([5], [6])
- Source code profiling on assembly and C-code levels is available

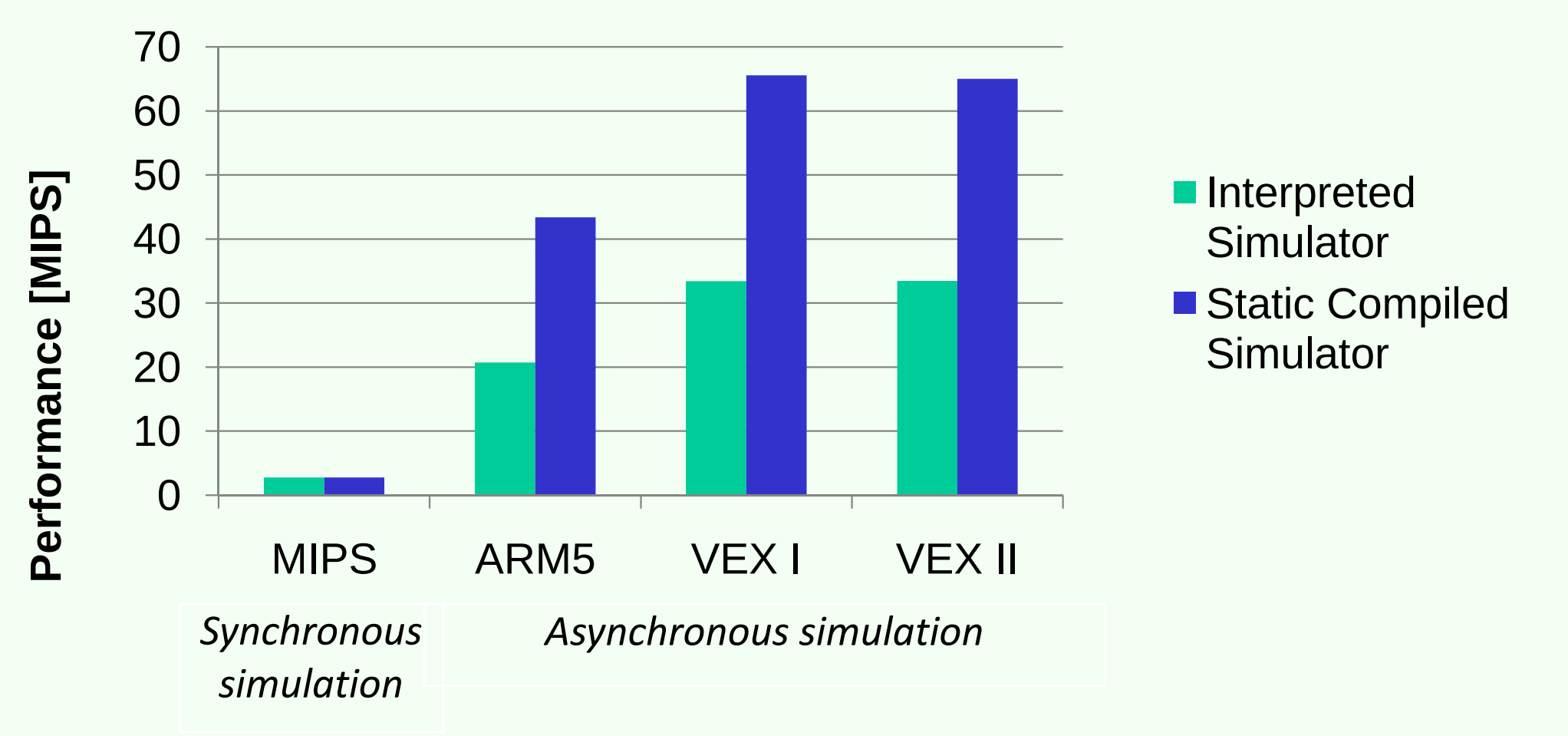

Fig. 4: Multi-core simulation speeds for the interpreted and compiled simulators

7 Conclusion

- Lissom project currently provides development environment that, from a compact processor model in ISAC language, allows to generate automatically toolchain, simulators with profilers, and hardware description
- Also a GUI based on Eclipse platform is provided
- Tool generators were tested multiple on processor models
- Results of this academic research are used in a commercial tool Codasip® from company ApS Brno, Codasip Division [7]

8 Acknowledgements

This research was supported by the doctoral grant GA CR 102/09/H042, by the grants of MPO Czech Republic FR-TI1/038, by the grant FIT-S-10-2, by the research plan no. MSM0021630528 and by the European project SMECY.

9 References

[1] Bailey, B., et al.: *ESL Design and Verification: A Prescription for Electronic System Level Methodology*, Morgan Kauffman Publishers, 2007.
[2] Hoffmann, A., Meyr, H., Leupers, R., *Architecture Exploration for Embedded Processors with LISA*, Kluwer Academic Publishers, 2002.
[3] Husár, A., et al.: Automatic C Compiler Generation from Architecture Description Language ISAC. In: *6th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, Brno, CZ, 2010.
[4] *MiBench* Version 1.0. Available on: http://www.eecs.umich.edu/mibench/.
[5] Hruška, T., Kolář, D., Lukáš, R., Zámečníková, E.: “Two-Way Coupled Finite Automaton and Its Usage in Translators”, *New Aspects of Circuit WSEAS*, Athens, GR, 2008.
[6] Přikryl, Z., Masařík, K., Hruška, T., Husár, A.: “Fast Cycle-Accurate Interpreted Simulation”, *In Tenth International Workshop on Microprocessor Test and Verification: Common Challenges and Solutions*, Austin, US, 2009.
[7] Codasip, Available on: www.codasip.com