Scale4Edge

Published on *Scale4Edge* (https://project.edacentrum.de/scale4edge)

Success Story: Multi-Compiler Support

2023/11/06

[1] [2] [3]

Introduction

The Scale4Edge project aims to create an efficient RISC-V ecosystem for edge application optimization. It focuses on developing a versatile platform to provide cost-effective, application-specific edge devices and services for various market segments. This is achieved through fine-grained adaptation of generic components, covering CPU instructions, user/application software, and operating system/firmware levels. The ecosystem is highly scalable and customizable to individual applications, supporting various hardware architectures and non-functional properties like energy efficiency, fault tolerance, reliability, safety, and security.

The Scale4Edge ecosystem currently contains three different compilers, addressing different use-cases of the ecosystem:

CompCert is a formally verified compiler for safety relevant application of high assurance levels

X-LLVM is an extendible compiler for custom instructions based on Clang/LLVM

A custom configurable GCC compiler

CompCert – a Verified Compiler

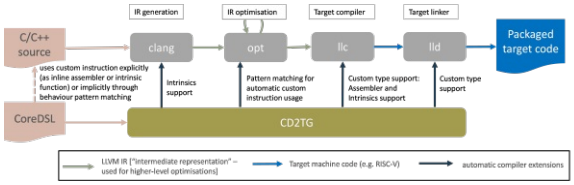
CompCert is a compiler for the C programming language (https://www.absint.com/compcert/ [4]). It accepts most of the ISO-C 99 language, with some minor exceptions and a few useful extensions. Its intended use is the compilation of life-critical and mission-critical software written in C and meeting high levels of assurance.

What sets CompCert apart from any other production compiler is that it is formally verified, using machine-assisted mathematical proofs, to be exempt from miscompilation issues. In other words, the executable code it produces is proved to behave exactly as specified by the semantics of the source C program. This level of confidence in the correctness of the compilation process contributes to meeting the highest levels of software assurance. Using the CompCert C compiler is a natural complement to applying formal verification techniques (static analysis, program proof, model checking) at the source code level: the correctness proof of CompCert guarantees that all safety properties verified on the source code automatically hold as well for the generated executable.

In 2022, the Association for Computing Machinery, ACM, presented the CompCert development team with the prestigious ACM Software System Award (https://www.absint.com/releases/220511.htm [5]) and the ACM SIGPLAN Programming Languages Software Award (https://www.sigplan.org/Awards/Software/ [6]).

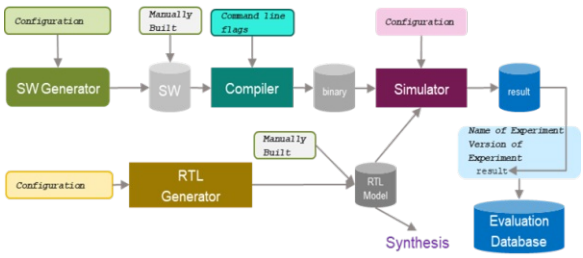
During the Scale4Edge project, CompCert was given a backend for RISC-V. CompCert for RISC-V generates code for the base instruction sets RV32I and RV64I with standard extensions M (Integer Multiplication and Division), F (Single-Precision Floating-Point), and D (Double-Precision Floating-Point). CompCert for RISC-V has been combined with picolibc (https://github.com/picolibc/picolibc [7]), a C library designed for embedded 32- and 64-bit microcontrollers with small memory. After some changes proposed by the CompCert developers (mostly to avoid GCC/Clang specific language extensions), the picolibc source code can now be compiled with CompCert.

X-LLVM – an extendible Compiler for custom instructions



Existing toolchains with RISC-V support cannot yet be flexibly extended to quickly support ISA extensions. In addition, RISC-V extensions in the Scale4Edge project are described using a separate language called CoreDSL. An automated translation of these CoreDSL definitions into corresponding toolchain extensions, as far as possible, is therefore not only desirable for reasons of flexibility, but also almost inevitable for reasons of consistency. For this reason, the described hardware (processor core) and a virtual platform and a compiler with support of custom defined instructions can be generated from CoreDSL. We have successfully automated the modification of Clang/LLVM to support custom instructions throughout the whole software toolchain (compiler, linker debugger), as depicted in the figure. Based on a CoreDSL description of the custom instructions, the extendible compiler toolchain (called X-LLVM) implements a compiler generator (called CD2TG) to provide the defined custom instructions as assembler code or intrinsic function for explicit usage. If possible, custom instructions are also implicitly used by the compiler (e.g., MAC (Multiply Accumulate) or Zero Overhead Loop). If this were done based on a manual translation, there would be a potential for an incorrect or at least inconsistent translation at each individual step. By automating this process, these errors can be avoided. Another benefit of automation is the speed with which the different artifacts can be regenerated. For example, if a new CoreDSL description is created or an existing description is revised, the corresponding tool chains can be triggered directly to generate the different artifacts and thus be able to test the newly defined instructions directly and revise them further if necessary. The current open source release of X-LLVM is available at https://github.com/DLR-SE/extensible-compiler [8].

GCC – the power of optimization



gcc supports plenty of optimization opportunities, most can be controlled by over 120 command line flags.

The Infineon RISC-V RTL generator supports also plenty of configurations (superset of CoreDSL) to generate a huge number of RISC-V variants differing in function, performance in clock cycles and performance in clock period. From the perspective of compiler performance analysis, the number of clock cycles is relevant. The maximum clock period is provided by the synthesis – or better R2G – flow. Performance evaluation is done by RTL simulation but can be moved to FPGAs and Emulation as RISC-V or SoC features (Timers) are used to measure time. For evaluation, classical benchmarks but also generated software has been applied. Of course, the framework is also used to run regression over the complete generation chain and validate the compiler result for a specific code and application.

Results worth mentioning are a substantial impact of compiler optimization, more performance when comparing to other non-RISC-V cores but also a non-neglectable memory footprint overhead.

Cont@ct:

CompCert | Reinhold Heckmann | AbsInt Angewandte Informatik GmbH | heckmann@absint [dot] com

X-LLVM | Kim Grüttner | German Aerospace Center (DLR) | kim [dot] gruettner@dlr [dot] de

GCC | Wolfgang Ecker | Infineon Technologies AG | wolfgang [dot] ecker@infineon [dot] com (wolfgang.ecker@infineon.com)

ARQUIMEA [9] BOSCH Technik fürs Leben [10] concept engineering [11] EBERHARD KARLS UNIVERSITÄT TÜBINGEN

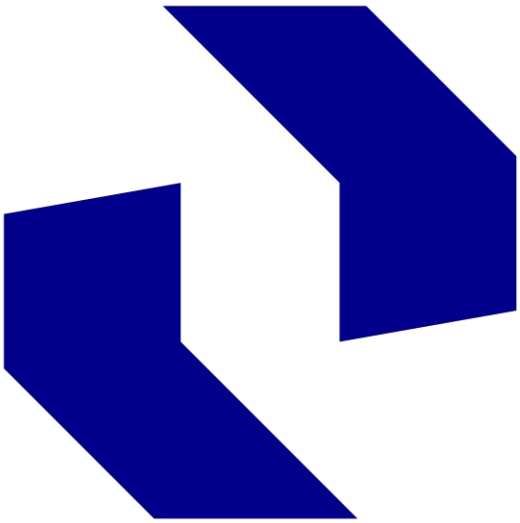
[12] EPOS [13] FZI [14]

[15] LAUTERBACH [16]

MINRES TECHNOLOGIES [17] BOSCH Technik fürs Leben [10]

SIEMENS EYSGO EMBEDDING INNOVATIONS [18] Universität Bremen [19] [20]

TECHNISCHE UNIVERSITÄT DARMSTADT [21]



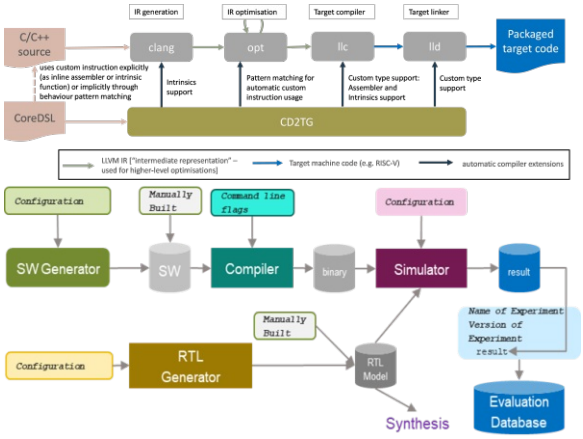
Technisch Universit Dresden

[22] [23] [24]

[25]

UNIVERSITÄT PADERBORN Die Universität der Informationsgesellschaft [26]

[27]



Herunterladen:

Success Story: Multi-Compiler Support [28]

The Scale4Edge project (project label 16ME0122K-140, 16ME0465, 16ME0900, 16ME0901) is supported by the German Federal Ministry of Research, Technology and Space (BMFTR).

Source URL: https://project.edacentrum.de/scale4edge/en/success-story-multi-compiler-support

Links:
[1] https://www.infineon.com
[2] https://www.AbsInt.com
[3] https://www.dlr.de
[4] https://www.absint.com/compcert/
[5] https://www.absint.com/releases/220511.htm
[6] https://www.sigplan.org/Awards/Software/
[7] https://github.com/picolibc/picolibc
[8] https://github.com/DLR-SE/extensible-compiler
[9] https://www.arquimea.com/
[10] https://www.bosch.de
[11] https://www.concept.de
[12] https://uni-tuebingen.de/fakultaeten/mathematisch-naturwissenschaftliche-fakultaet/fachbereiche/informatik/fachbereich/
[13] https://www.epos-d.com
[14] https://www.fzi.de
[15] https://www.ihp-microelectronics.com
[16] https://www.lauterbach.com
[17] https://www.minres.com
[18] https://eda.sw.siemens.com/en-US/
[19] https://www.sysgo.com
[20] https://www.uni-bremen.de
[21] https://www.tu-darmstadt.de
[22] https://tu-dresden.de/
[23] https://uni-freiburg.de
[24] https://rptu.de/
[25] https://www.ei.tum.de/eda/
[26] https://www.hni.uni-paderborn.de/sct/
[27] https://www.edacentrum.de
[28] https://project.edacentrum.de/scale4edge/en/system/files/ct_project_news/scale4edge-success-story-compilers.pdf