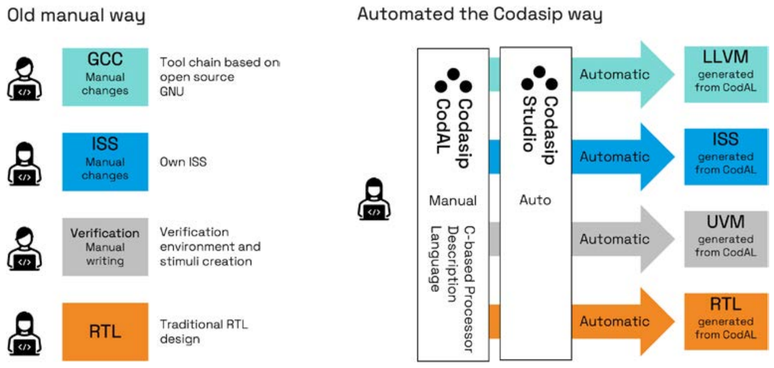
Re-targetable C/C++ LLVM compiler for RISC-V

Zdenek Prikryl, CTO

→ Scaling is failing

For about 50 years the semiconductor industry has relied on shrinking silicon geometries to achieve greater design complexity and processor performance for an acceptable cost. This shrinking has been most famously described by Moore's Law and the less wellknown Dennard Scaling. This virtuous and predictable scaling is broken – so how can we achieve improvements in performance in the future?

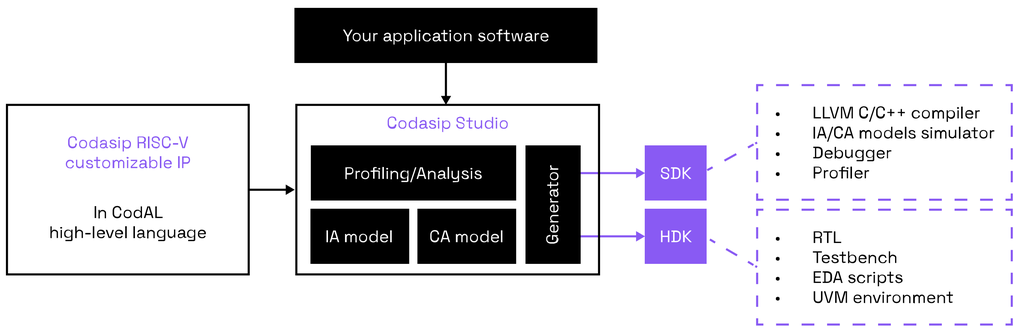


→ Customizable RISC-V processors

Differentiate with Codasip Studio

Configure / Modify

Using CodAL architecture description language



→ Re-targetable C/C++ LLVM

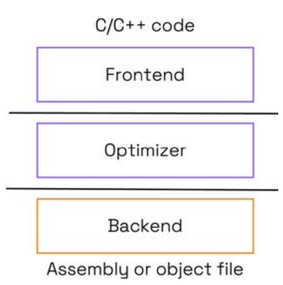
Codasip uses LLVM as a base line.

LLVM is the re-targeted based on the CodAL processor description.

Beside the C/C++ compiler, Codasip generates

- LLVM assembler, disassembler
- LLVM linker, binutils
- LLVM debugger (LLDB)

Complete SDK/toolchain is generated.



→ Benefits and features

CodAL processor description serves as an input to the generator

Generator extracts:

- Behavior of every single instruction in a form of a graph
- Architectural and microarchitectural features for a scheduler or register allocation
- Application binary interface
- Peephole and other optimizations

Informative report is generated

Designer may see which instructions are recognized and how they will be used

C/C++ compiler uses the instructions automatically

No need to change the C/C++ code

If an instruction is too complex, then:

- Intrinsic is automatically generated
- User may use the instruction via the intrinsic or inline assembly

→ Performance and code size

LLVM Vanilla as a base line

- GCC as well as Codasip LLVM is then compared relatively to it

Coremark compiled for performance

Dhrystone is compiled using legal arguments only

Codasip outperforms both, GCC as well as LLVM Vanilla

- Can be improved further using custom compute with Codasip Studio

LLVM Vanilla as a base line

- GCC as well as Codasip LLVM is them compared relatively to it

Embench-iot used as a benchmark

The same optimization options are used for the compilation across compilers

Codasip produces smaller applications

- Can be improved further using custom compute with Codasip Studio