Chair of Electronic Design Automation
Department of Computer Engineering
School of Computation, Information and Technology
Technical University of Munich

LLVMGen: Automated Generation of a RISC-V LLVM Toolchain for Custom MACs

Philipp van Kempen, Karsten Emrich, Daniel Mueller-Gritschneder, Ulf Schlichtmann

Problem Statement

Motivation

- RISC-V can be extended with special instructions to customize embedded CPUs
- Evaluation of new instructions needs simulator and toolchain (compiler and assembler) support
- Extending Embedded SW compiler suites is very difficult and time intensive

State of the Art

- CoreDSL, a language for describing ISAs, was proposed in [2]
- ETISS[1] is an instruction set simulator (ISS) which can quickly evaluate the benefit of special instructions for a given application
- M2-ISA-R is a metamodel-driven Python tool, generating ETISS-support based on CoreDSL code

Goals

- Introduce a code generation tool for extending existing LLVM implementations with support for custom RISC-V instructions described in the CoreDSL format
- Artifacts should use the Tablegen syntax wherever possible
- Custom MAC instructions of Core-V Extension based on XpulpNN[4] and implemented in CV32E40P core shall be used as reference

Challenges:

- Behavior of instructions with multiple outputs can not be modeled in Tablegen.
- Information which is not part of CoreDSL description (Intrinsics, Aliases, Constraints) needs different format → YAML

Evaluation

TVM ML Compiler Suite [3]

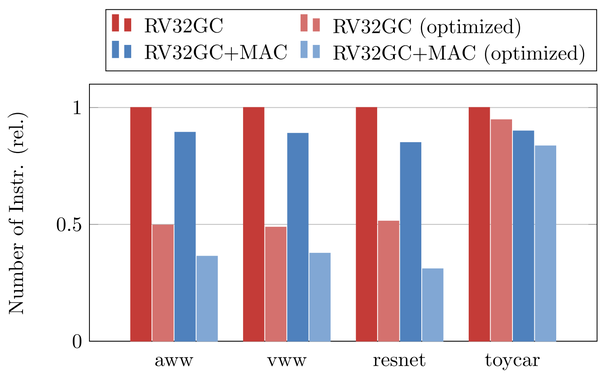
- Applying several optimizations including autotuning
- Runtime: CRT, Executor: Ahead-of-Time (AoT), Memory planning: USMP

MLPerf Tiny Benchmark [5]

- 4 Models:
  - Audio Wake Words (`aww`)
  - Visual Wake Words (`vww`) } CNN
  - Image Classification (`resnet`)
  - Anomaly Detection (`toycar`) } DNN

Core-V MAC Instructions:

- Utilized:
  - MAC, MSU
  - MACSN, MACHHSN (with patches)
- Unutilized:
  - MACUN, MACHHUN
  - MACURN, MACHHURN
  - MASURN, MACHHSRN



Default TVM Performance:

- Layout: NHWC (Channels-last)
- Kernels: Untuned
- Using XCoreVMac:
  - CNNS: 15% speedup
  - DNN: 10% speedup

Optimized TVM Performance:

- Layout: NCHW (Channels-first)
- Kernels: Tuned with AutoTVM
- Using XCoreVMac:
  - CNNS: 35% speedup
  - DNN: 10% speedup

Tuning + Layout Transform:

CNNS: $\approx$ 2x speedup

DNN: $\leq$ 20% speedup

References

[1] Mueller-Gritschneder, D., et al. (2017, October). The extendable translating instruction set simulator (ETISS) interlinked with an MDA framework for fast RISC prototyping. In *Proceedings of the 28th International Symposium on Rapid System Prototyping: Shortening the Path from Specification to Prototype* (pp. 79-84).
[2] Ecker, W., et al. (2022, March). The Scale4Edge RISC-V Ecosystem. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 808-813). IEEE.
[3] Chen, T., et al. (2018). TVM: An automated end-to-end optimizing compiler for deep learning. *arXiv preprint* arXiv:1802.04799.
[4] Garofalo, A, .et al. (2020, March). XpulpNN: Accelerating quantized neural networks on RISC-V processors through ISA extensions. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 186-191). IEEE.
[5] Banbury, C., et al. (2021). Mlperf tiny benchmark. *arXiv preprint* arXiv:2106.07597.

Flow

1. Description of XCorevMac Instructions

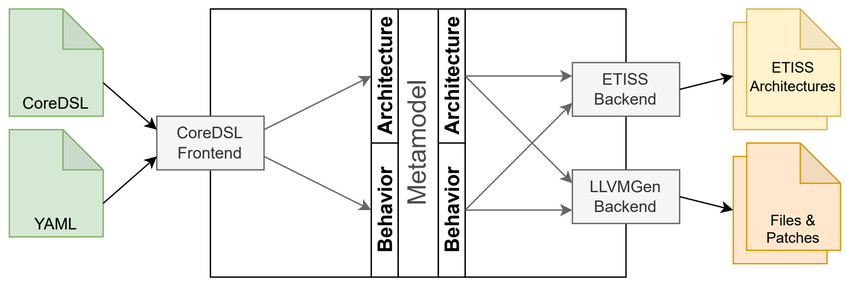
- CoreDSL: Name, Encoding, Assembly and Operation for each instruction

```
CV_MAC {
  encoding: 7'b1001000::rs2[4:0]::rs1[4:0]::3'b011::rd[4:0]::7'b0101011;
  assembly: {"cv.mac", "{name(rd)}, {name(rs1)}, {name(rs2)}"};
  behavior: {
    signed<65> result = (signed)X[rs1] * (signed)X[rs2] + (signed)X[rd];
    if(rd != 0) X[rd] = result[31:0];
  }
}
```

- YAML: Intrinsics, Aliases,...

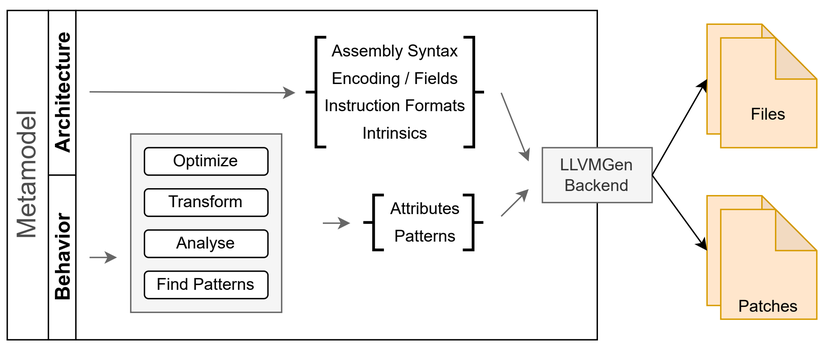
2. Processing with M2-ISA-R

- CoreDL Frontend: Parsing of CoreDSL code and conversion to Metamodel
- Metamodel: Storing information on **Architecture** (Encoding) and **Behavior** (Operation)
- ETISS Backend: Generation of C++ files for our instruction set simulator



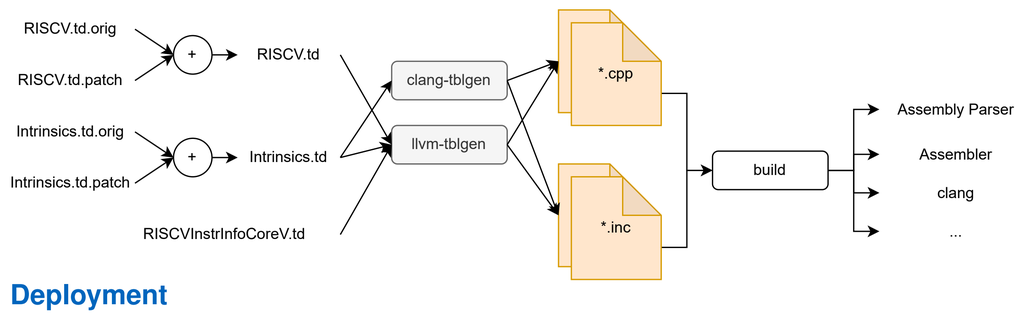
3. Running LLVMGen Backend

- Extract information from metamodel by applying transformation, optimization and analysis passes



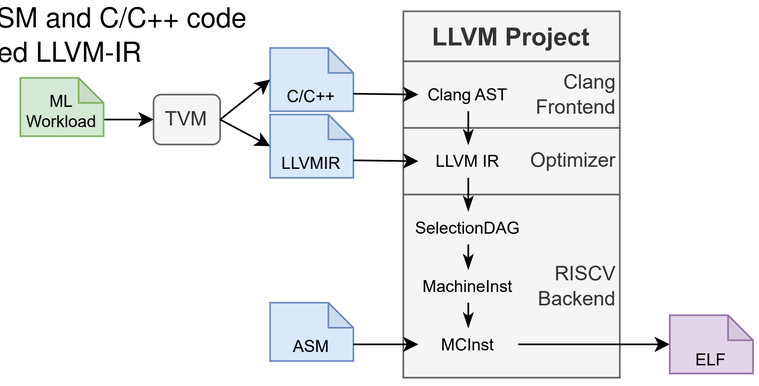
4. Patching of Codebase and Compilation

- Combination of generated patches with upstream LLVM repository
- Tablegen tools generate various files during compilation of LLVM



5. Deployment

- Supports handwritten ASM and C/C++ code as well as TVM-generated LLVM-IR



Future Work

- Support more types of instructions (Memory, SIMD,...)
- Utilize more custom instructions automatically
- Microarchitecture-aware scheduling and cycle-accurate simulation
- Validation on instruction test suite
- Automatic generation of unit-tests

This work has been developed in the project MANNHEIM-FlexKI funded by the German Federal Ministry of Education and Research (BMBF) under contract no.01IS22086L. The authors are responsible for the content of this publication.

SPONSORED BY THE Federal Ministry of Education and Research

Contact:
philipp.van-kempen@tum.de
karsten.emrich@tum.de
daniel.mueller@tum.de

Open Source:
https://github.com/tum-ei-eda/
LLVMGen available later this year