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

Philiipp 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 toolchain and 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
- Antiflots 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 (xaw)
  - Visual Wake Words (vuw)
  - Image Classification (reazo)
  - Anomaly Detection (toycars)
- DNN
- CNNS:
  - MACUN, MACHHUN
  - MACSN, MACHHSN (with patches)
- MAC, MSU

**Core-V MAC Instructions:**
- Utilized:
  - MAC, MSU
  - MACSN, MACHHSN (with patches)
- Unutilized:
  - MACUN, MACHHUN
  - MACSN, MACHHSN

**Default LLVM Performance:**
- Layout: NHWC (Channels-first)
- Kernels: Untuned
- Using XCoreVMac:
  - CNNs: 1.7x speedup
  - DNN: 1.8x speedup

**Optimized LLVM Performance:**
- Layout: NCHW (Channels-last)
- Kernels: Tunned with AutoTVM
- Using XCoreVMac:
  - CNNs: 2.3x speedup
  - DNN: 2.6x speedup

**Flow**

1. **Description of XCoreMac Instructions**
   - CoreDSL: Name, Encoding, Assembly and Operation for each instruction
   - CV_MAC
     - encoding: 7'b0001000: rd[4]: {name(rd)}, {name(rs1)}, {name(rs2)}
     - assembly: "CV_MAC", {name(rd)}, {name(rs1)}, {name(rs2)}
   - YAML: Intrinsics, Aliases,...

2. **Processing with M2-ISA-R**
   - CoreDSL 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
   - LLVMGen available later this year

5. **Deployment**
   - Supports handwritten ASM and C/C++ code as well as LLVM generated LLVM-IR

**References**


This work has been developed in the project MANNHEIM-FlexKI funded by the German Federal Ministry of Education and Research (BMBF) under contract no. 16S13093B. The authors are responsible for the content of this publication.