MLIR as Hardware Compiler Infrastructure

Schuyler Eldridge, Prithayan Barua, Aliaksei Chapyzhenka, Adam Izraelevitz,
Jack Koenig, Chris Lattner, Andrew Lenharth, George Leontiev, Fabian Schuiki,
Ram Sunder, Andrew Young, Richard Xia

SiFive
schuyler.eldridge@sifive.com

Modern hardware design languages [1, 2, 3, 4] enable agile hardware development. Such development produces faster hardware iteration cycles, more successful hardware projects, and a lower barrier to entry for companies doing hardware development. Necessarily, new hardware design languages must produce Verilog or VHDL to interface with existing proprietary or open source hardware tools. Due to this constraint, many of these languages adopt compiler-like architectures. They often include one or more intermediate representations (IRs) and lower these IRs to Verilog via a sequence of passes. Unfortunately, this has resulted in fragmentation with each language project resolving the same problems:

1) Implementation of common compiler passes like dead code elimination, common subexpression elimination, and constant propagation
2) Generation of human-readable Verilog or VHDL
3) Development of performant compiler infrastructure

Critically, item 3 has proved problematic for large designs based on Chisel which may take an hour or more to finish compilation of their FIRRTL IR.

To alleviate these issues, we are building CIRCT [5]. CIRCT[1] (Circuit IR Compiler and Tools) is built on MLIR [6], a modern, best-in-class compiler infrastructure project that is both a part of the LLVM project and an evolution of it. MLIR is extremely performant, supports SSA-like or graph-like regions, enables representation and intermixing of multiple dialects (different IRs), and lets users define new dialects.

Currently, CIRCT consists of a number of hardware dialects: common hardware dialects for sequential or combinational logic, a Verilog/SystemVerilog dialect, a FIRRTL dialect, and other more exotic, higher-level dialects (e.g., latency-insensitive dialects). While incomplete, CIRCT also provides an order-of-magnitude improvement in FIRRTL compilation times for SiFive designs. Figure 1 shows the collection of dialects in CIRCT and the compilation path from Chisel to Verilog.

In this work-in-progress presentation we will provide an overview of the CIRCT project, discuss its use at SiFive as a replacement FIRRTL compiler, and discuss future directions for the project.

[1] CIRCT is pronounced /sər-kət/.

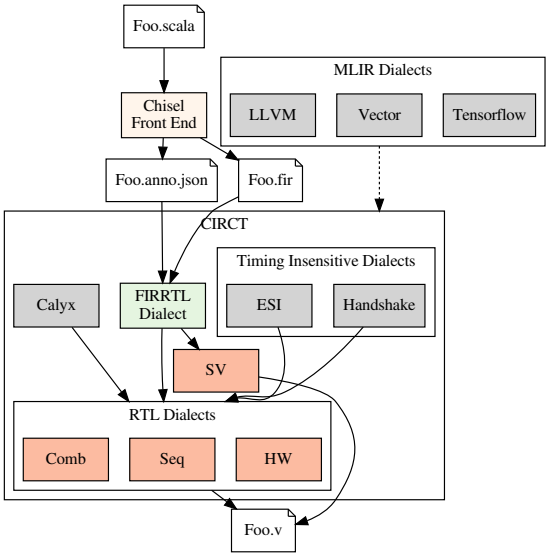

Fig. 1. CIRCT consists of a collection of hardware dialects (IRs) that enable representing circuits and converting them to Verilog. CIRCT can also be used as a faster compiler for Chisel [1] designs.

References

[1] Jonathan Bachrach et al. “Chisel: constructing hardware in a scala embedded language”. In: 2012 Design Automation Conference (DAC). IEEE.

[2] Oron Port et al. “DFiant: A dataflow hardware description language”. In: 2017 International Conference on Field Programmable Logic and Applications (FPL). IEEE.

[3] Rick Bahr et al. “Creating an Agile Hardware Design Flow”. In: 2020 ACM/IEEE Design Automation Conference (DAC).

[4] Rachit Nigam et al. “Predictable accelerator design with time-sensitive affine types”. In: 2020 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI).

[5] LLVM Project. Circuit IR Compilers and Tools (CIRCT). url: https://github.com/llvm/circt.

[6] C. Lattner et al. “MLIR: Scaling Compiler Infrastructure for Domain Specific Computation”. In: 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO).