Charting CIRCT

The present and future landscape

John Demme | Fabian Schuiki | Mike Urbach | Andrew Young

Microsoft SiFive Alloy Computing SiFive

2021 LLVM Developer Meeting

Talk waypoints

1. What is CIRCT? (Quick rehash of the keynote)
2. What's so different about hardware? (vs software or firmware)
3. Selected subprojects
   - FIRRTL: supporting Chisel
   - HLS: lower software into hardware
   - ESI: linking hardware components together and tying in software
4. Selected subsystems
   - Core dialects
   - Simulation
5. CIRCT subproject summary

CIRCT IR for Compilers and Tools … or is it spelled “circuit”?

Hardware compiler tech is ripe for innovation and disruption!

- **Expensive** commercial products suffer from **quality** issues, **interoperability**, and lack of **innovation velocity**.
- New generation of developers fed up with status quo → many new, **innovative open source** technologies.
- Open source products suffer from **startup costs** and **interoperability** problems.

MLIR-based compiler infrastructure for **hardware design and verification**.

- Aims to do for hardware compilers and tools what LLVM did for software.
- **Modular library** allows HW compiler devs to bring their sliver of innovation and let CIRCT do the rest.
- **Interoperability** provided by **standard dialects and APIs**.
- Engineered for **quality** -- solid base to build upon.

“CIRCT: Lifting hardware development out of the 20th century” (Andrew Lenharth and Chris Lattner)

Keynote at this meeting further motivates CIRCT.

What's so different about hardware? (Why isn't the LLVM IR sufficient?)

Massively parallel: **everything** runs concurrently… every. single. op.

- Imperative programming models clearly don't apply.
- Parallel/concurrent SW models don't express ultra-fine-grained parallelism efficiently.
- Converting traditional software models into hardware (via **HLS**) is possible…
  and occasionally even has good results!
- Hardware-specific **languages** necessary.

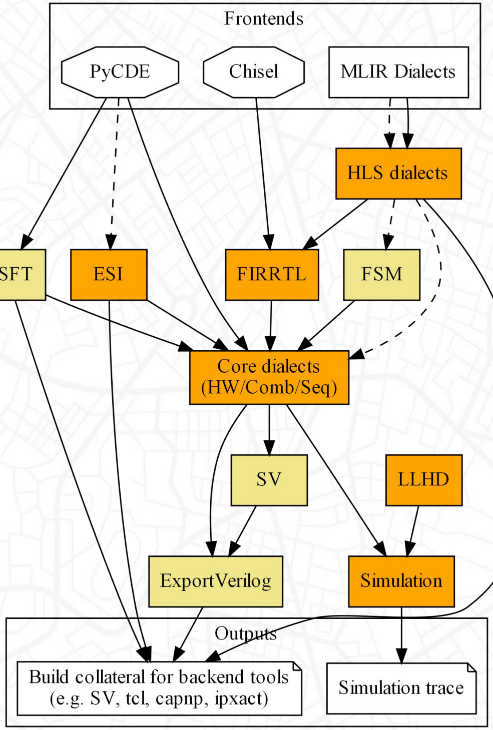
No global, shared memory: designers create "scratchpad"-like local memories.

- **Pro**: no pointers! **Con**: no pointers.
- Must move data **explicitly**, both intra- and inter- chip. No magic remote access.

Zero visibility at runtime: we lack the optical technology to observe wire activity!

- Necessitates **simulation** for debug and verification.
- Wise designers also include some debug/telemetry circuitry.

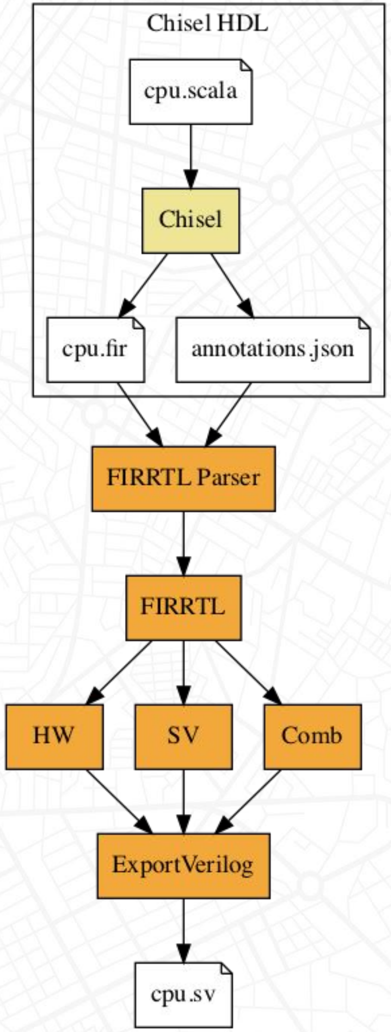
Fodor's list of must-see places (Rick Steves mostly agrees)



[{ Overview photo of CIRCT taken from Mars }]

FIRRTL: Supporting Chisel

- FIRRTL is the name of the compiler IR used by the Chisel hardware description language (HDL)
- HDLs are DSLs used to describe circuit structure
- In CIRCT we are writing a drop-in replacement for the Scala FIRRTL compiler
- FIRRTL IR parser imports to the FIRRTL Dialect
- FIRRTL Dialect lowers to common CIRCT Dialects

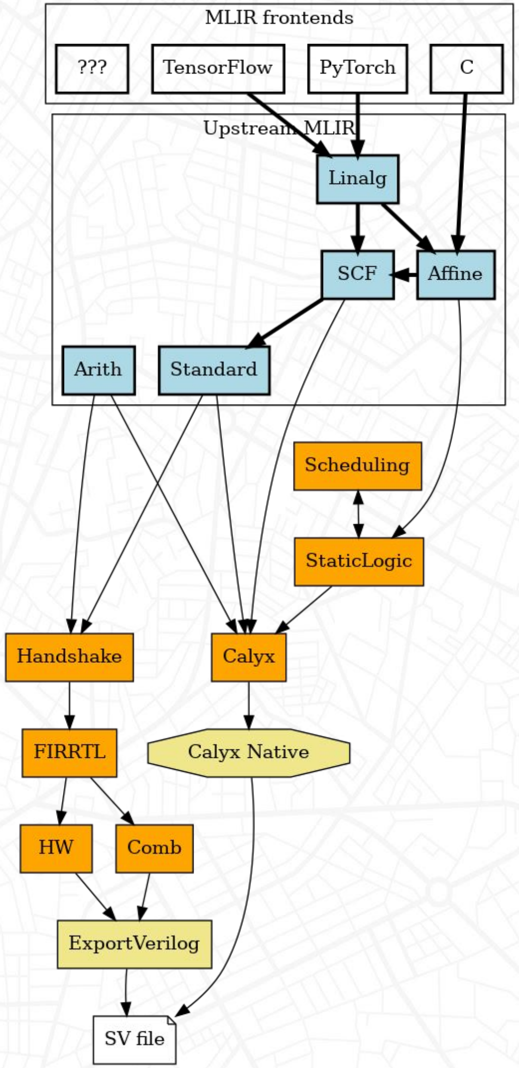


FIRRTL - Why replace the FIRRTL compiler?

- Improve compile times with heavy memory usage
  - Compile times could be 10 minutes, use >64gb ram
  - Multithreaded MLIR based compiler improves performance 10-30x
- Replace bespoke compiler infrastructure
  - Reusable components create a community
  - Leverage shared lower level dialects, transformations, Verilog exporter
- Better Verilog output
  - First class representation of SystemVerilog
- Interoperability with other HDL and HLS

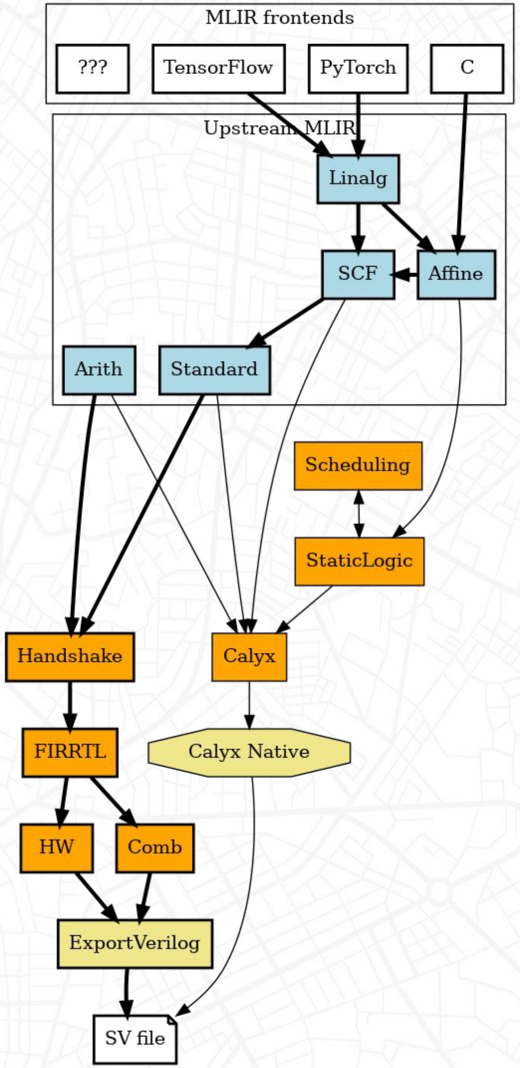
HLS

- HLS compiles a “high level” program into hardware description for FPGAs or ASICs
    - Historically based on C-like languages
- Many challenges translating software IRs to hardware
    - Akin to auto-parallelizing C compilers
- MLIR presents a huge opportunity for HLS
    - Same core IRs for frontends to target
    - New IRs designed for HLS
- CIRCT project is building HLS IRs, analyses, and transformations



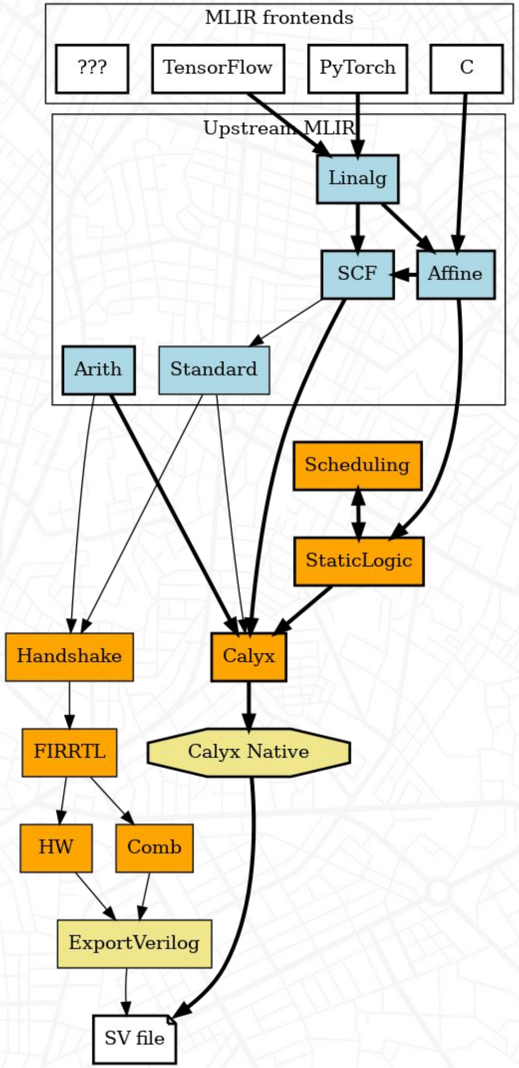
HLS

- Hardware dataflow graph without pre-computed schedule
  - Doesn't require scheduling
  - Extra overhead for control and buffering
- CIRCT IRs capture dataflow semantics
  - Handshake dataflow graph and operators
  - Lowering into hardware implementation
- CIRCT simulators based on LLVM
  - Handshake simulator for dataflow graph
  - Cosimulation of software and hardware



HLS

- Hardware finite-state machine and datapath with pre-computed schedule
  - Can be highly optimized and predictable
  - Requires scheduling, allocation, and binding
- CIRCT IRs capture scheduling semantics
  - Pipeline with static schedule
  - Finite-state machine and datapath with Calyx
- CIRCT scheduling library contains high-quality scheduling algorithms
  - Treats HLS as an optimization problem



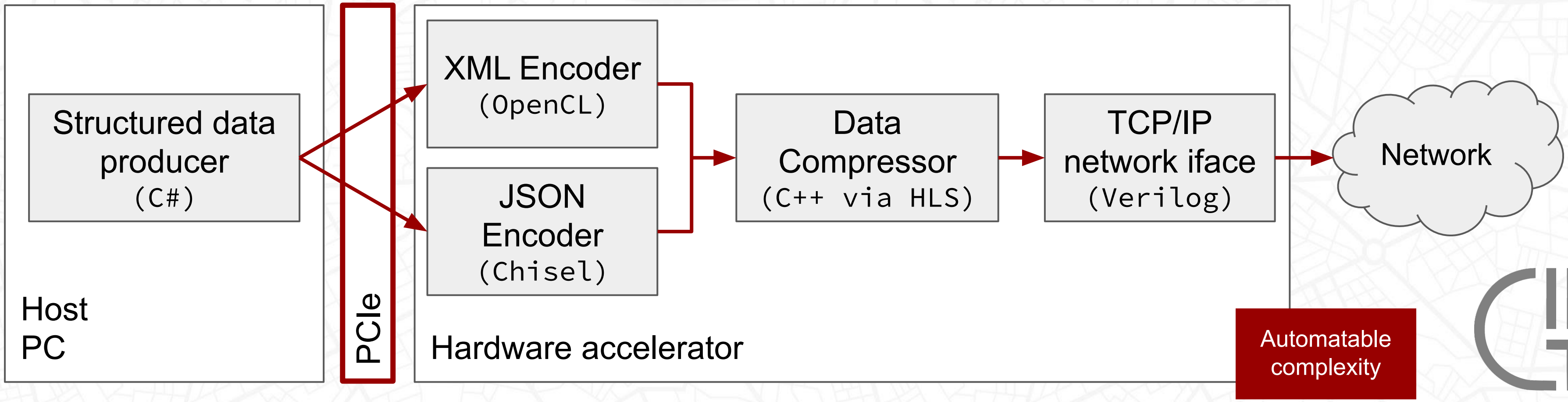
Elastic Silicon Interconnect: it's all about communication

Hardware designs often contain multiple semi-independent subsystems.

- They must communicate with each other and the host.
- In software, main memory is used. HW doesn't have global shared memory.
- Difficult for multiple languages to live on the same device.

An *interconnect* and "*runtime*" must be built to connect the subsystems and the host.

- "Plumbing" is **tedious**, **error-prone**, but **straightforward**. Ripe for automation!



Elastic Silicon Interconnect: it's all about communication

Hardware designs often contain multiple semi-independent subsystems.

- They must communicate with each other and the host.
- In software, main memory is used. HW doesn't have global shared memory.
- Difficult for multiple languages to communicate.

An *interconnect* and *runtime* must be built to connect the subsystems and the host.

- "Plumbing" is **tedious**, **error-prone**, but **straightforward**. Ripe for automation!

ESI takes a typed specification of the communication graph and builds the interconnect.

- Including a bridge to host software, providing an design-specific, typed API.
- Even bridges to simulation, exposing the **same** API for so-called "co-simulation".

"Elastic Silicon Interconnects: Abstracting Communication in Accelerator Design"

J. Demme, LATTE'21 [paper] [talk]

Status: proof of concept

Core dialects: the common denominator

HW: core abstractions

- Operations like module, instance (of a module)
- Also contains standard data types (int, array, struct, etc.)
- Status: mostly **complete**, mostly **stable**

Combinational: computational ops without a sense of cycles or time

- Operations like add, shift, multiply, etc.
- Status: **complete** and **stable**

Sequential: contains clocked storage elements

- Introduces a sense of time measured in cycles.
- Status: **incomplete** but **stable**

Simulation

- Designs are simulated many times for debugging and verification before **Si** production
- Simulation means:
  - Take the hardware description/model
  - Apply some stimulus to its inputs
  - Check the response on its outputs
- Designs go through multiple levels of simulation
- Existing commercial simulators are:
  - Expensive
  - Slow
  - Have obscure performance cliffs
- Buy-in point for CIRCT

Usual Progression of Simulations

RTL-level functional tests

Run tests against high-level language source code.

Gate-level functional tests

Run tests against the logic gates produced by the logic synthesizer and chip layout tool.

Gate-level **timing** tests

Simulation of logic gates back-annotated with propagation delays extracted from Si layout.

Future buy-in point for CIRCT

Using CIRCT immediately gives you a community-curated simulator

Like buying into LLVM immediately gives you solid codegen and JIT for a large number of processors.

Simulation

- Si verification requires timing-aware simulation
- "Traditional" hardware languages do this with an event queue programming model
- **MLIR is brilliant for this:**
  - Separate dialect to interact with event queue
  - Keep higher-level dialect ops for speed
  - Additional dialect for simulation optimization

MLIR allows us to:

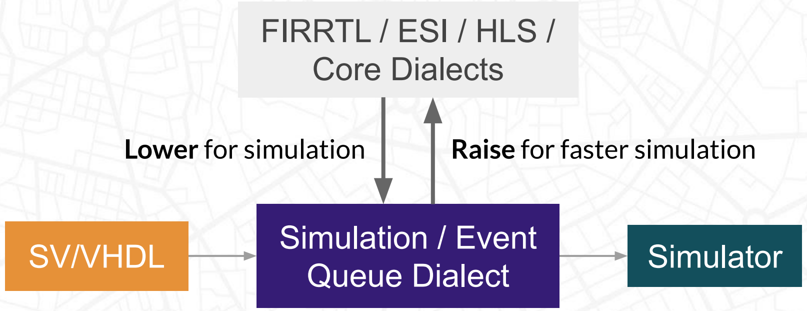
Have a dialect to model event queue and signal timing

Keep high-level IR ops where it allows for fast simulation

99% of a hardware unit test is pure, side-effect-free dataflow; small fraction interacts with event queue.

Add dialects for simulation opt.

Separating hardware into modules is for humans; simulation needs separation into state transfer functions.


FIRRTL / ESI / HLS / Core Dialects
Lower for simulation
Raise for faster simulation
SV/VHDL
Simulation / Event Queue Dialect
Simulator

CIRCT wants YOU!

https://circt.llvm.org/

https://github.com/llvm/circt

Discourse discussion board

Weekly discussions Wed. @ 11am PT

FIRRTL	Core features complete / Missing some annotations
HLS	Limited prototype / In Development
ESI	Proof of concept
Core	Stable, mostly complete
Simulation	SystemVerilog prototype Ongoing integration with core dialects

Credits: all the CIRCT contributors!

Join us in disrupting the hardware world!