Using Pyrope to Create Transformable RISC-V Architectures

Haven Skinner, Rafael T. Possignolo, Jose Renau

Department of Computer Engineering, University of California, Santa Cruz
http://masc.soe.ucsc.edu
Challenge to Address

- Difficult to design hardware
  - Need multiple Codebases
    - Cycle-accurate version for synthesis
    - High-level simulation
    - Verification reference model
  - Need to adjust pipeline stages
  - Need to verify the hardware
- New HDL (Pyrope) can help
My Hammer to Handle Design Complexity

- Fluid Pipelines design

- Correct by construction Fluid Pipelines transformations
  - Change the number of stages as part of compile flow

- Use Pyrope Language to build Fluid Pipelines designs
Quick Fluid Pipeline Review

- Each pipeline stage has valid/stop signals
- Pipeline stages should tolerate random delays
Quick Fluid Pipeline Review
Quick Fluid Pipeline Review
Quick Fluid Pipeline Review
Quick Fluid Pipeline Review

![Fluid Pipeline Diagram]

RISC-V 2017
Using Pyrope to Create Transformable RISC-V Architectures
Haven Skinner, Rafael T. Possignolo, Jose Renau
Quick Fluid Pipeline Review
Quick Fluid Pipeline Review
Quick Fluid Pipeline Review
New Fluid Pipeline Collapse

We can reduce designs to a single pipeline stage
Fluid Collapse Usages

- Build an emulator out of a RISC-V Fluid Core
- Change number of pipeline stages automatically
- Perform formal verification
Build a Fast Simulator

![Graph showing performance comparison between Baseline, Fluid Emulator, and Spike with Fluid Sampling and Spike Sampling]
Build a Fast Simulator

![Graph showing MIPS comparison between Baseline and Fluid Emulator]
Build a Fast Simulator

Bar chart comparing MIPS performance of Baseline, Fluid Emulator, and Spike.

- Baseline
- Fluid Emulator
- Spike

MIPS

Baseline: Fluid Emulator: Spike:
Build a Fast Simulator

![Graph comparing MIPS for different methods](image)

- Baseline
- Fluid Emulator
- Spike
- Fluid Sampling
- Spike Sampling

MIPS vs. Method Comparison
Fluid Synthesis

![Diagram showing the relationship between Area and Delay for different RISC-V architectures: PICORV32, VScale, Zero-riscy, and RI5CY.](image)
Fluid Synthesis

![Fluid Synthesis Diagram](image)

- Non-Fluid
- Fluid, no transform
- PICORV32
- c4
- c4+ fwd
- c5+ fwd
- VScale
- Zero-riscy
- RI5CY

Area vs. Delay (ns) graph showing performance comparison between different RISC-V architectures during fluid synthesis.
Fluid Synthesis

![Fluid Synthesis Diagram]

The diagram illustrates the relationship between area and delay for various RISC-V architectures. The architectures are categorized into Non-Fluid, Fluid, no transform, and Fluid, transformed. The specific architectures include PICORV32, VScale, Zero-riscy, and several variations of c4 and c5, with options for 2-stage and 3-stage configurations.

The points on the graph represent different instances of these architectures, with markers indicating their performance in terms of area and delay.
Fluid Verification

- Just collapse the RISC-V core
- Verify it against a trivial single stage RISC-V core
- We used this method on our RISC-V cores
What about Pyrope?

• No time to cover a language in 12 minutes but…
  • I created Pyrope 1.0, this work uses it
  • Akash Sridhar (UCSC) is building Pyrope 2.0 parser
    • Tutorial https://masc.soe.ucsc.edu/pyrope.html
  • Working on new backend for fast code generation and hot reload
    • Still not available
  • But we are committed, just look at my advisor’s license plate
Questions?

Haven Skinner, Rafael T. Possignolo, Jose Renau
hskinner@ucsc.edu

Department of Computer Engineering,
University of California, Santa Cruz
http://masc.soe.ucsc.edu