RiTA (RISC-V Trace Analyzer)

Anmol Sahoo, Project Associate, IIT Madras
Neel Gala, CTO, Incore Semiconductors
Outline

- Need for a trace analyzer
- Design and features of RiTA
- Examples
- Getting started with RiTA
Need for a trace analyzer
Need for a trace analyzer

- Hardware designers need to understand software execution characteristics
- Trace analyzers should be able to analyze traces from generic execution contexts - Spike, FPGA, Silicon
- Designers can better use this information to extend or improve their designs at various levels

Architectural Decisions
- Macro-op Fusion
- Pseudo-ops
- ISA Extensions
Need for a trace analyzer

- Hardware designers need to understand software execution characteristics
- Trace analyzers should be able to analyze traces from generic execution contexts - Spike, FPGA, Silicon
- Designers can better use this information to extend or improve their designs at various levels
- Architectural Decisions
  - Macro-op Fusion
  - Pseudo-ops
  - ISA Extensions
Need for a trace analyzer

- Hardware designers need to understand software execution characteristics
- Trace analyzers should be able to analyze traces from generic execution contexts - Spike, FPGA, Silicon
- Designers can better use this information to extend or improve their designs at various levels

- Architectural Decisions
  - Macro-op Fusion
  - Pseudo-ops
  - ISA Extensions
Need for a trace analyzer

- Hardware designers need to understand software execution characteristics
- Trace analyzers should be able to analyze traces from generic execution contexts - Spike, FPGA, Silicon
- Designers can better use this information to extend or improve their designs at various levels
- Architectural Decisions
  - Macro-op Fusion
  - ISA Extensions
  - Hardware Optimization
Design and features of RiTA
Design of RiTA

- RiTA is a command line tool written in Python
- Tiered application
  - Trace File Parser
  - Instruction Parser
  - Post-processing Modules
- Makes it easy to extend!
Features of RiTA

The following post-processing modules are being developed in RiTA,

- Instruction histogram and statistics
- Register histogram
- Memory access histogram
- Branch statistics
- Dependent instruction sequences
Examples
Instruction Histogram

Top 20 Instructions Histogram for Dhrystone
Register Access Statistics

Register Access Stats for Dhrystone
Program Counter Histogram

Top 10 Values of Program Counter Histogram for Dhrystone

- Instruction access address
- Access count

- 0x80003f00: 17500
- 0x80003f04: 17500
- 0x8000a66c: 10000
- 0x8000a670: 10000
- 0x8000a674: 10000
- 0x80001358: 9500
- 0x8000135c: 9500
- 0x80001360: 9500
- 0x80001364: 9500
- 0x80001368: 9500
Getting Started

- The application is available from the PyPI package repository
- Get Started,
- `pip install rita`
- Future work will look at,
  - Parsing new ISA extensions
  - Optimization suggestions
Thank you for your time! :D