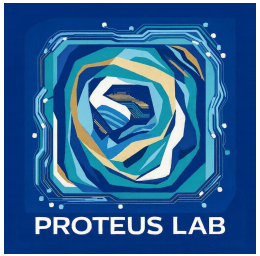
Visualizing gem5 O3CPU Execution Traces with Perfetto

Moscow Center for Advanced Studies

Anton Shurygin, Andrey Derzhavin, Denis Los



Introduction

- The O3CPU model is a complex implementation of a superscalar pipeline with out-of-order execution
- For a detailed analysis of the pipeline, visual analysis tools are required:
  - Debug Flags
  - O3PipeView
  - Third Party …

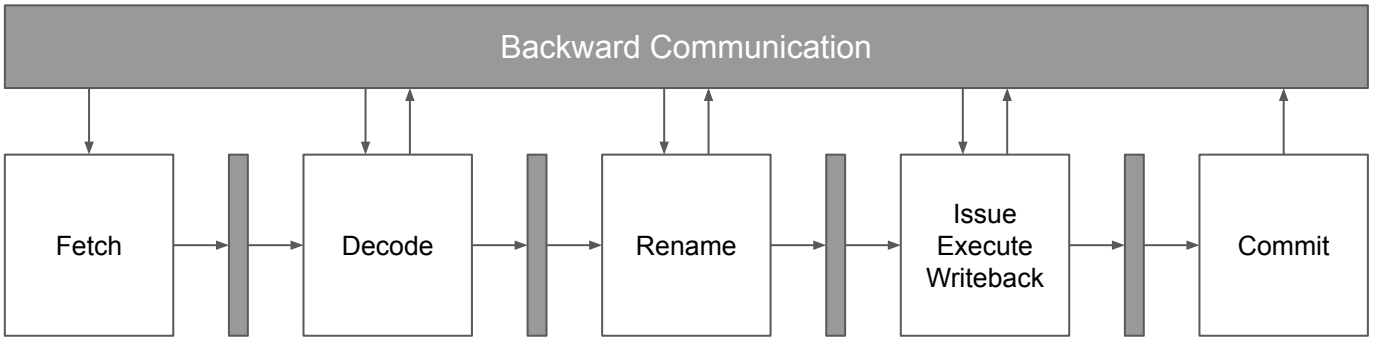
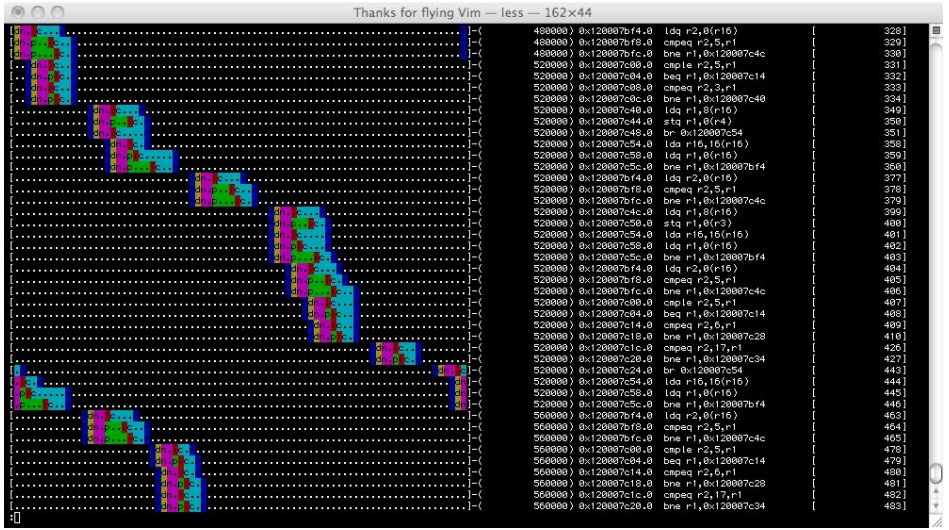


Fig. 1. O3 CPU Model, gem5 bootcamp

Current State

- O3PipeView:
  - ➕ Simple visualization in terminal pseudo graphics
  - ➕ Integrated in gem5
  - ➖ No zooming
  - ➖ Limited interactive exploration



Current State

- O3PipeView:
  - ➕ Simple visualization in terminal pseudo graphics
  - ➕ Integrated in gem5
  - ➖ No zooming
  - ➖ Limited interactive exploration
  - ➖ Timeline display issues

```
// f = fetch, d = decode, n = rename, p = dispatch, i = issue, c = complete, r = retire

                                    timeline                                                                  tick          pc.upc     disasm                    seq_num
[.ic.r.....................................................................fdn]-(         40000) 0x0001010c.0 addi ra, zero,           [         1]
[.pi..c.r..................................................................fdn]-(         40000) 0x00010110.0 mul sp, ra, r            [         2]
[.p...i..c.r...............................................................fdn]-(         40000) 0x00010114.0 mul gp, sp, r            [         3]
[.p.....i..c.r.............................................................fdn]-(         40000) 0x00010118.0 mul tp, gp, r            [         4]
[.p...ic.....r.............................................................fdn]-(         40000) 0x0001011c.0 add t0, ra, s            [         5]
[.p.....ic...r.............................................................fdn]-(         40000) 0x00010120.0 add t0, t0, g            [         6]
[.p.......ic.r.............................................................fdn]-(         40000) 0x00010124.0 add t0, t0, t            [         7]
[.p.........ic.r...........................................................fdn]-(         40000) 0x00010128.0 addi t0, t0,             [         8]
[.p.ic.....r...............................................................fd]-(         40000) 0x0001012c.0 addi a7, zero, 9         [         9]
```

Fig. 2. Visualizing a simple program with O3PipeView

Current State

- Konata:
  - ➕ Graphical interface
  - ➕ Interactive exploration and zooming
  - ➖ No filters by pipeline stages/events
  - ➖ Complexity (~7k lines) → [no support since 2023]

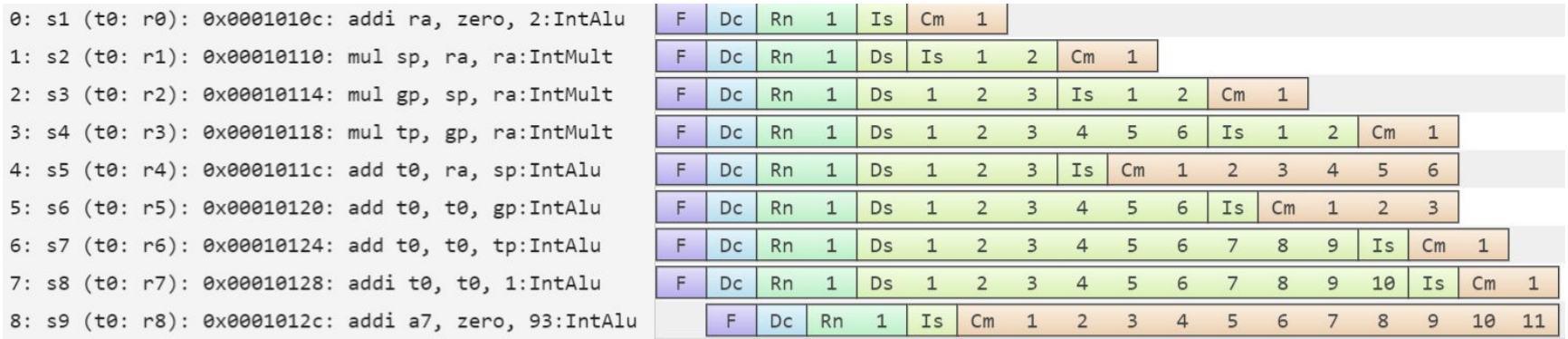


Fig. 3. Visualization of a simple program using Konata

Current State

- These tools allow you to verify the O3 CPU pipeline
- It may help answer the question of why the instruction is executed slowly

Tool	UI	Scalability	Integration	Filtering
O3PipeView	TUI	❌	✅	❌
Konata	GUI	✅	❌	❌

Table 1. Comparison of existing visual analysis tools

Problem

- These tools only provide a pipeline view by stages
- There is no open source solution to help analyze the resource utilization of the gem5 simulator
- But we don't want to develop yet another one visualizer…
- However, there is Perfetto, a platform for performance analysis

Tool	UI	Scalability	Integration	Filtering
Perfetto	WUI	✅	❌	✅

Table 2. Perfetto State

Solution Architecture

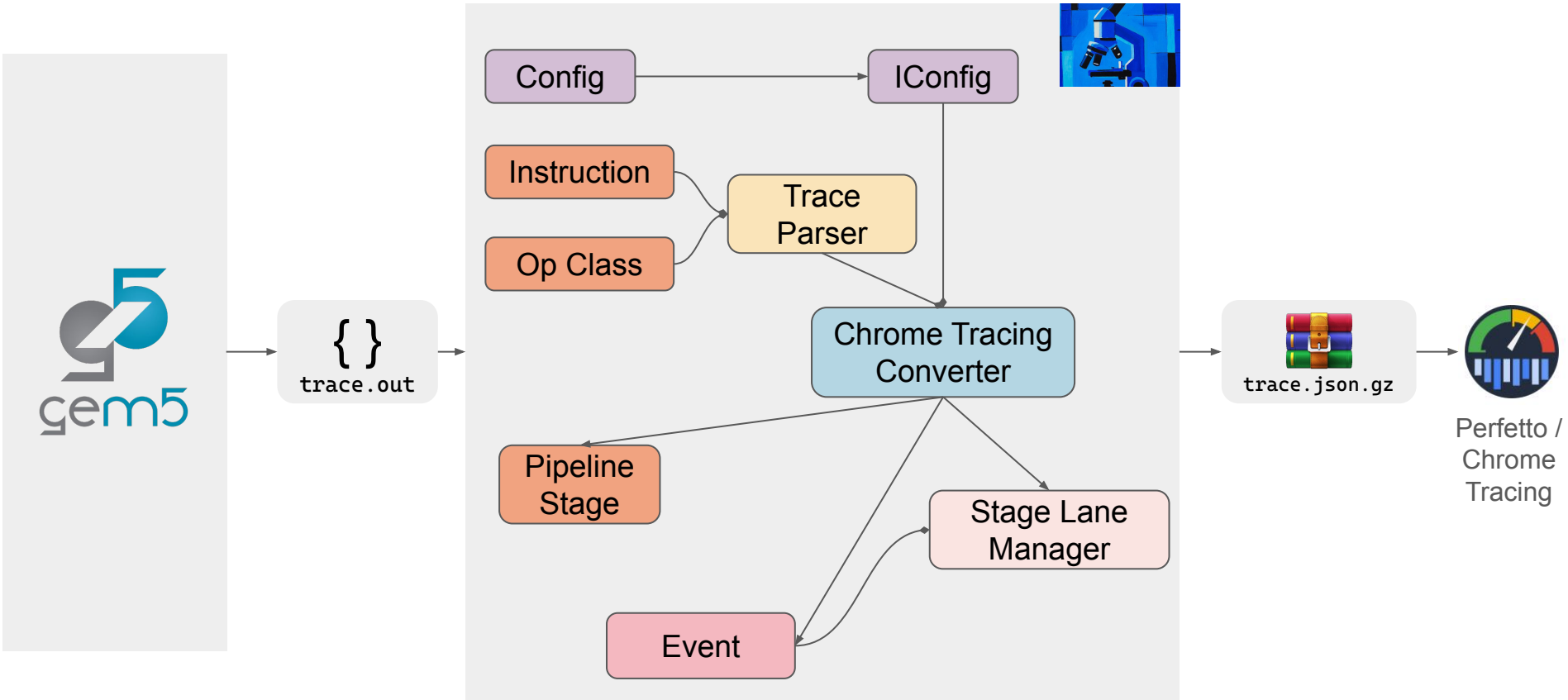
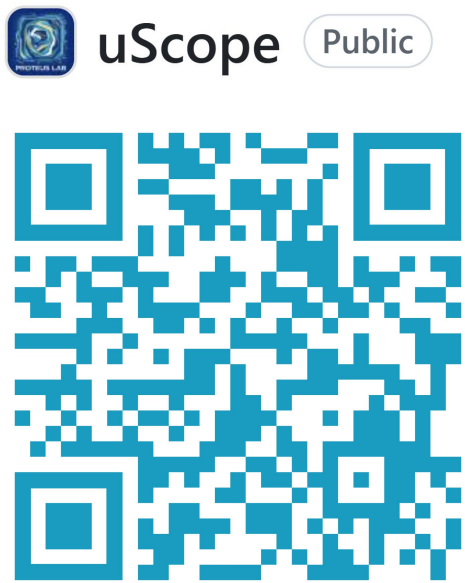


Fig. 5. Workflow of the developed tool

How to get it?

- Open-source repository
- Apache License Version 2.0
- You can pass as input:
  - Vanilla O3PipeView trace
    `--debug-flags=O3PipeView`
  - uScope specific debug trace
    `--debug-flags=O3PipeView,UscopeView`
- The patched gem 5 simulator can be found here



How to use it?

```
.section .data
    .align 4
    array_A: .word 1, 2, 3, 4,\
                   5, 6, 7, 8
    array_B: .word 8, 7, 6, 5,\
                   4, 3, 2, 1

# Initialization
loop_scalar:
    lw x14, 0(x10)
    lw x15, 0(x11)

    add x16, x14, x15
    sw x16, 0(x12)

    addi x10, x10, 4
    addi x11, x11, 4
    addi x12, x12, 4
    addi x13, x13, -1
    bnez x13, loop_scalar
# Exit
```

Listing 1. Element-wise addition of two arrays in RISC-V assembler

- This example [can be found in the uScope repository]
- So, how to use this tool?

```
# Compile sample program

> riscv64-linux-gnu-gcc /path/to/scalar.S -O0 -march=rv64gv -mabi=lp64d -static
-mcmodel=medany -fvisibility=hidden -nostdlib -nostartfiles -o /path/to/scalar
```

```
# Run gem5

> ./build/RISCV/gem5.opt --debug-flags=O3PipeView,UscopeView
--debug-file=/path/to/scalar.out configs/deprecated/example/se.py
--cpu-type=O3CPU --caches --cmd=/path/to/scalar
```

```
# Run uScope

> uScope --input-file /path/to/scalar.out --output-dir
/path/to/scalar_tracings/
```

How to use it?

```
> uScope --input-file /path/to/scalar.out --output-dir
/path/to/scalar_tracings/
```

```
.section .data
    .align 4
    array_A: .word 1, 2, 3, 4,\
                   5, 6, 7, 8
    array_B: .word 8, 7, 6, 5,\
                   4, 3, 2, 1

# Initialization
loop_scalar:
    lw x14, 0(x10)
    lw x15, 0(x11)

    add x16, x14, x15
    sw x16, 0(x12)

    addi x10, x10, 4
    addi x11, x11, 4
    addi x12, x12, 4
    addi x13, x13, -1
    bnez x13, loop_scalar
# Exit
```

Listing 1. Element-wise addition of two arrays in RISC-V assembler

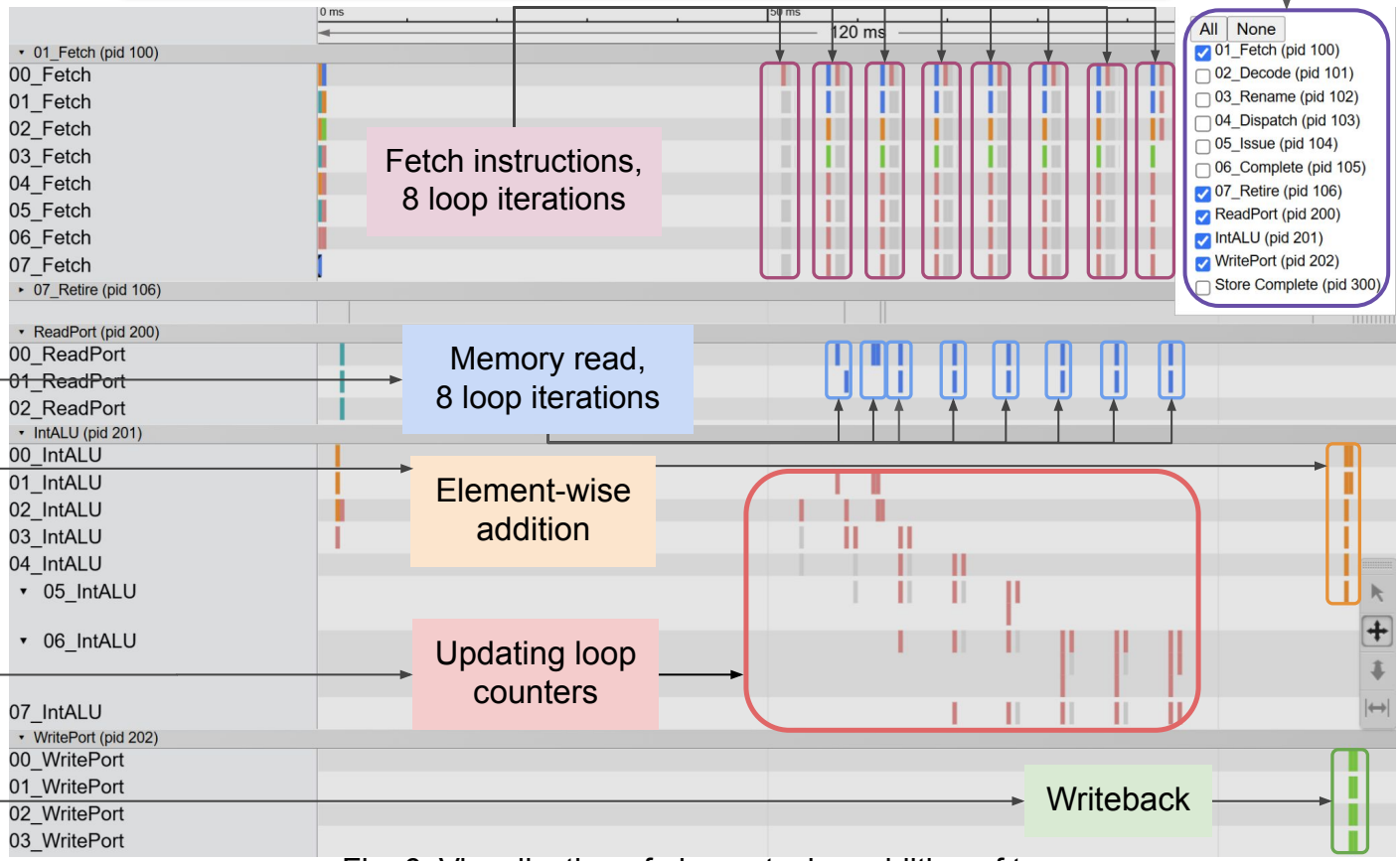


Fig. 6. Visualization of element-wise addition of two arrays

Features

Feature	uScope Option	Affects on gem5?	Status
Trace Compression	`--gzip`	❌	✅
Multicore	Automatically distribute different cores	Execution core id dump	✅
Data flow dependencies	`--data-flow`	Data flow dependencies dump	Experimental, dev-branch

How to use it?

```
> uScope --input-file /path/to/scalar.out --output-dir
/path/to/scalar_tracings/ --data-flow --gzip
```

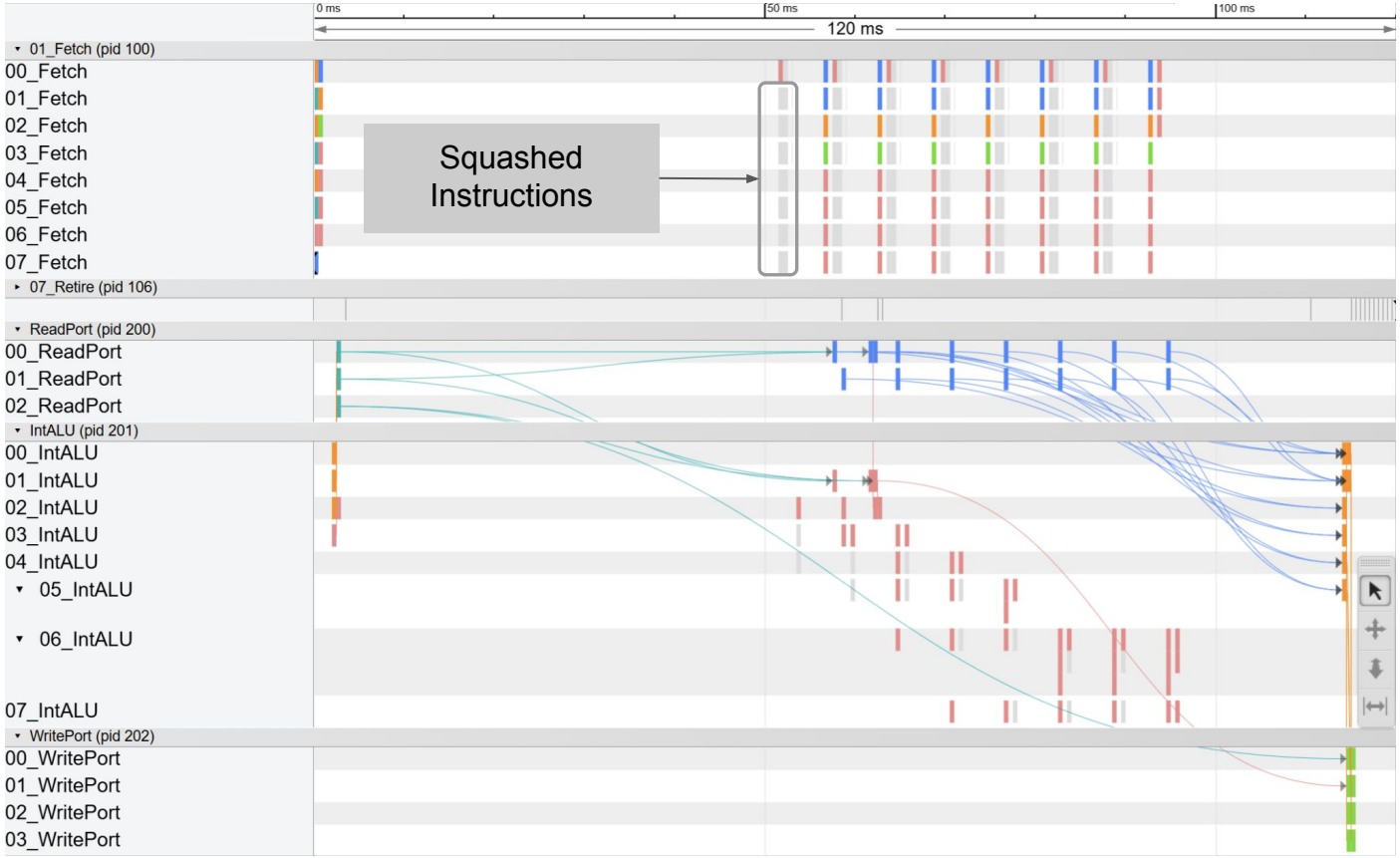
```
.section .data
    .align 4
    array_A: .word 1, 2, 3, 4,\
                   5, 6, 7, 8
    array_B: .word 8, 7, 6, 5,\
                   4, 3, 2, 1

# Initialization
loop_scalar:
    lw x14, 0(x10)
    lw x15, 0(x11)

    add x16, x14, x15
    sw x16, 0(x12)

    addi x10, x10, 4
    addi x11, x11, 4
    addi x12, x12, 4
    addi x13, x13, -1
    bnez x13, loop_scalar
# Exit
```

Listing 1. Element-wise addition of two arrays in RISC-V assembler



Conclusion

- The tool provides a new resource-view approach for analysis
- Unit test coverage > 90%
- It was used to evaluate new extensions to the RISC-V ISA
- As further work:
  - Streaming Mode
  - AI agent + perfetto (SmartPerfetto)
  - Complex compute-bound scenarios

uScope Public

Thank you for your attention

Any questions?

shurygin.aa@phystech.edu