Fast End-to-End Simulation and Exploration of Many-RISCV-Core Baseband Transceivers for SDR-Access Networks

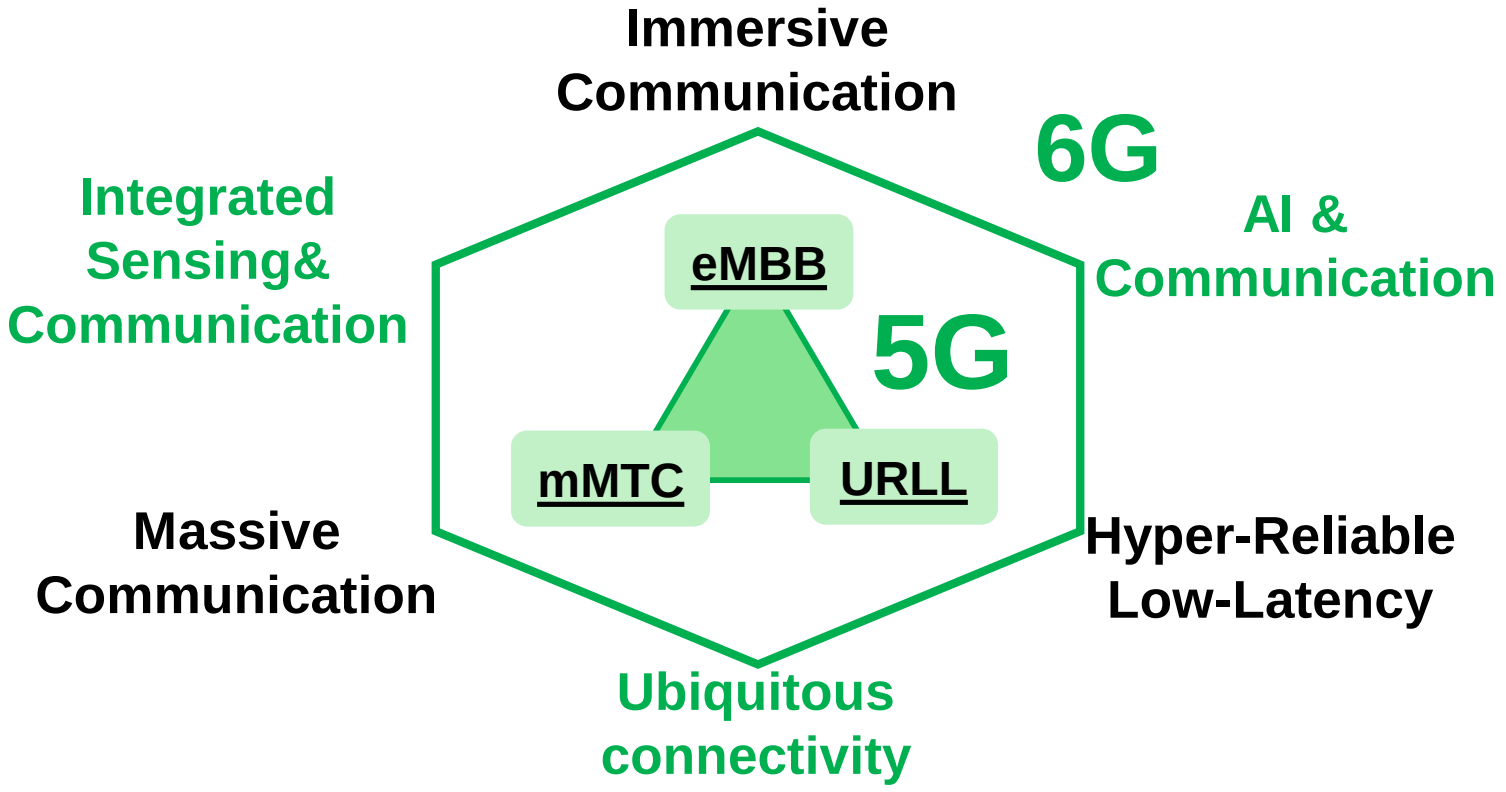
Marco Bertuletti* mbertuletti@iis.ee.ethz.ch, Mahdi Abdollahpour† mahdi.abdollahpour@unibo.it,
Yichao Zhang* yichzhang@iis.ee.ethz.ch, Samuel Riedel* sriedel@iis.ee.ethz.ch,
Alessandro Vanelli-Coralli*† avanelli@iis.ee.ethz.ch, Luca Benini*† lbenini@iis.ee.ethz.ch
ETH Zurich*, Università di Bologna†

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA SIGDA

Software-defined 6G-baseband, why?

1. **Open hardware&software** for inter-operability of **multi-vendor** components
2. **Distributed intelligence** for the densification of network functions at the **edge**
3. **Software-Defined** processing for **diverse scenarios** and **evolving standards**



SW-defined 6G-receivers: Modeling&Sim.

Requirements of SW-defined receiver modeling*:

* Wittig, *Modem Design in the Era of 5G and Beyond: The Need for a Formal Approach*, ICT, 2020

- Flexibility to evolving standard
- Deterministic behaviour
- Awareness of timing (to synchronize with other elements of processing chain)

Simulation must be fast:

- HW-in-the 6G-trasmission loop Monte Carlo simulation for end-end performance of the software-defined receiver

SoA doesn’t scale-up to large core-count!

Only up to 36 cores

	Device	Freq.	Speed	Design-Effort	Multi-Core
RTL [1-2]	-	-	↓	↓	?
TLM [3]	Intel Core2	3.00GHz	↑	↑	?
FPGA [4]	XCZU28DR	128MHz	↑	↓	?
FPGA [5]	ZCU102	120MHz	↑	↑	✓

[1] Rezgui, ICMCS, 2018; **[2]** Park, SMACD, 2021; **[3]** Barreteau, Signal Processing and Systems, 2013; **[4]** Cheng, MWSCAS, 2023; **[5]** Kamaledin, IEEE Access, 2020

SoA doesn't scale-up to large core-count!

	Device	Freq.	Speed	Design-Effort	Multi-Core
RTL [1-2]	-	-	↓	↓	?
TLM [3]	Intel Core2	3.00GHz	↑	↑	?
FPGA [4]	XCZU28DR	128MHz	↑	↓	?
FPGA [5]	ZCU102	120MHz	↑	↑	✓
SBT (ours)	AMD EPYC-7742	3.25GHz	↑	↓	✓

[1] Rezgui, ICMCS, 2018; **[2]** Park, SMACD, 2021; **[3]** Barreteau, Signal Processing and Systems, 2013; **[4]** Cheng, MWSCAS, 2023; **[5]** Kamaledin, IEEE Access, 2020

We need a **fast&streamlined** simulation for **Design-Space Exploration** of **SW-defined processing** on **large clusters**

SBT-based simulation of 6G-transceivers

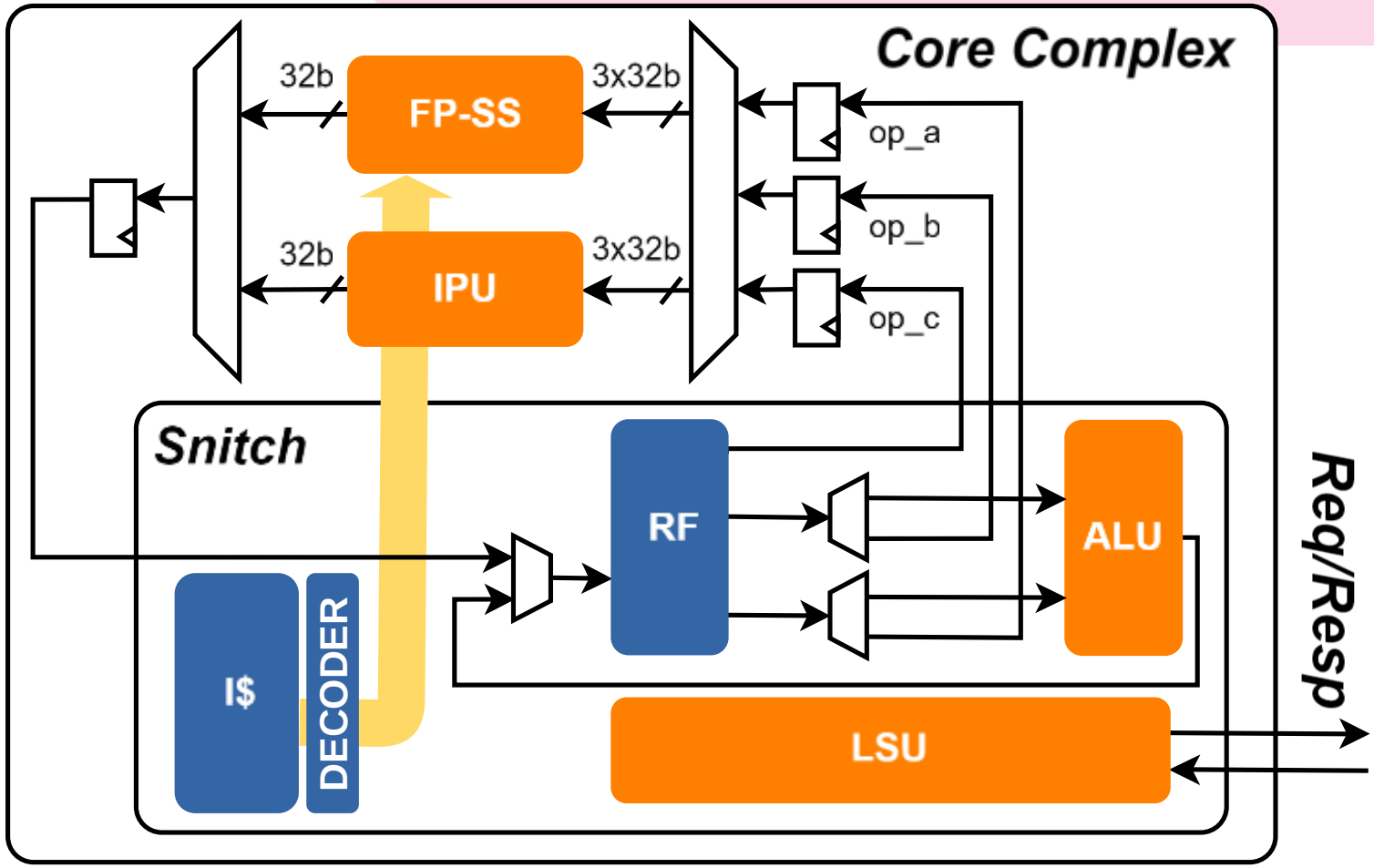
Banshee: open-source **Static Binary Translation** based simulator for **deterministic, instruction-accurate** simulation of **TeraPool**, a **1024-cores** RISC-V 6G BB-receiver

Features:

- Fast Design-Space exploration (< 4min per OFDM symbol)
- **Simple timing model** → 30% average-error on cycle-count

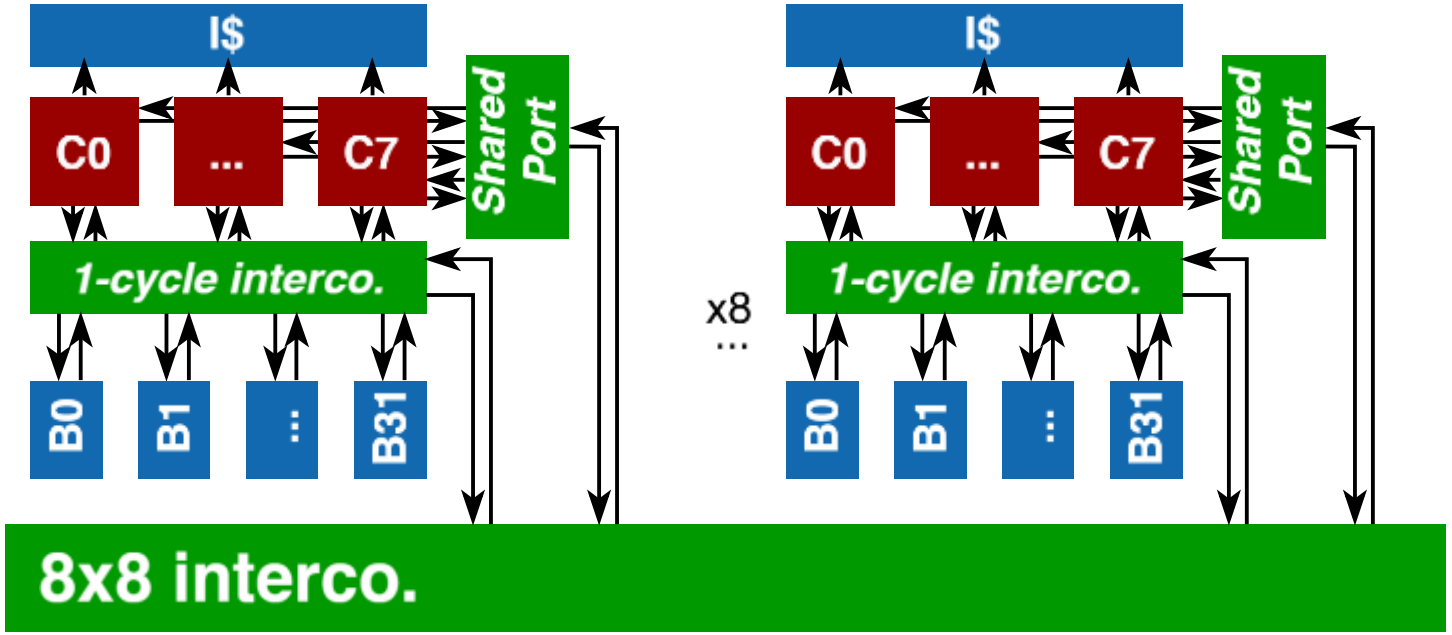
TeraPool: 1024-cores SW-defined receiver

- **1024** lightweigth **Snitch** cores, RISCV32IMA + **BB-processing ISA**
  - *Zfinx, zhinx,*
  - *int. / floating-point in-x SIMDs*
  - *mixed-precision floating-point in-x extensions (16b-32b, 8b-16b)*
  - *complex 16b floating-point dot-product*



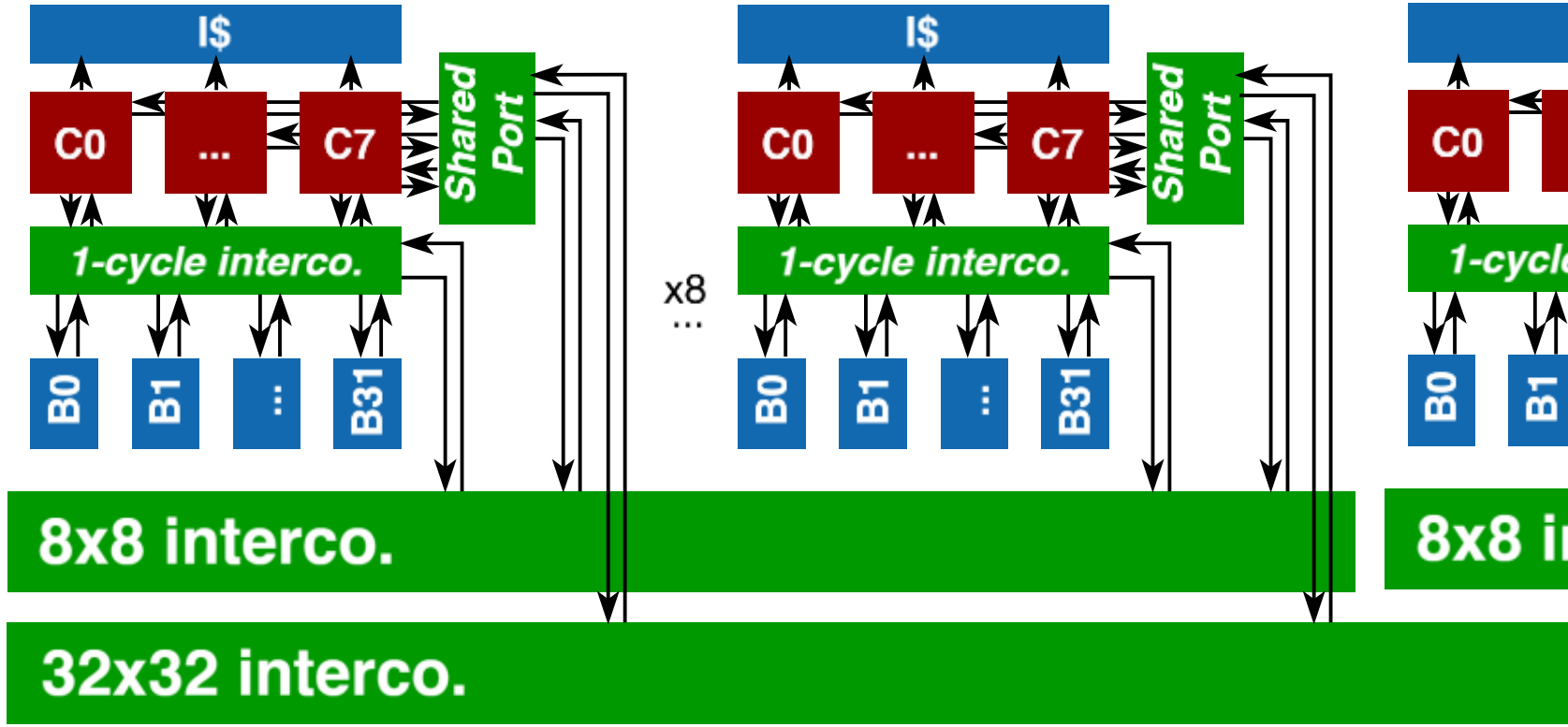
TeraPool: 1024-cores SW-defined receiver

- **1024** lightweigth cores, RISCV32IMA + **BB-processing ISA**
- **Hierarchical low-latency interconnect** to **4096 1KiB-banks scratchpad**



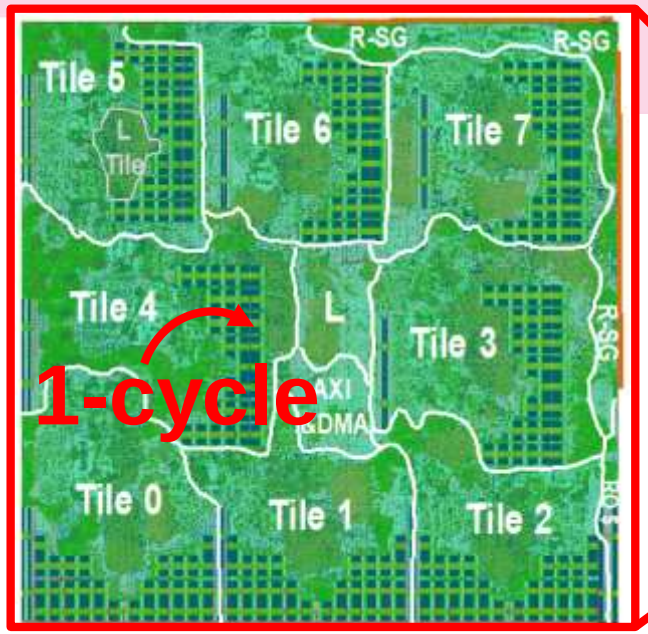
TeraPool: 1024-cores SW-defined receiver

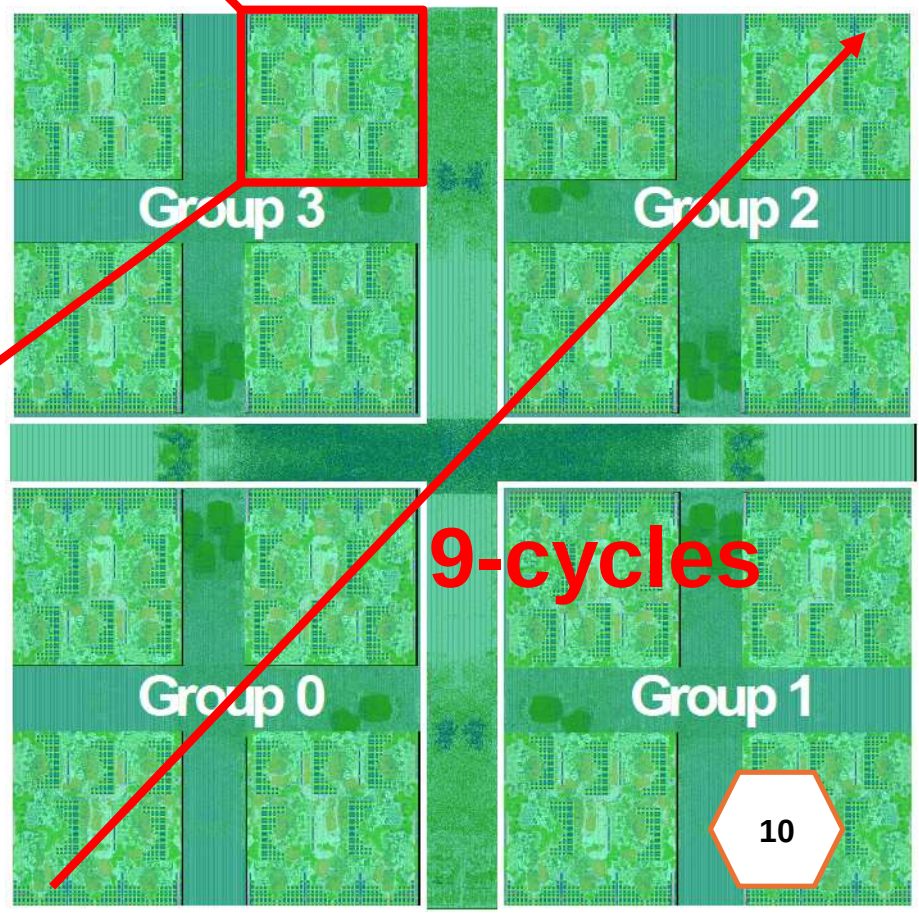
- **1024** lightweigth cores, RISCV32IMA + **BB-processing ISA**
- **Hierarchical low-latency interconnect** to **4096 1KiB-banks scratchpad**



TeraPool: 1024-cores SW-defined receiver

- **1024** lightweigth cores, RISCV32IMA + **BB-processing ISA**
- **Hierarchical low-latency interconnect** to **4096 1KiB-banks scratchpad**
- **Physically-feasible** GF12nm **910MHz** @(0.8V, 0.25C)

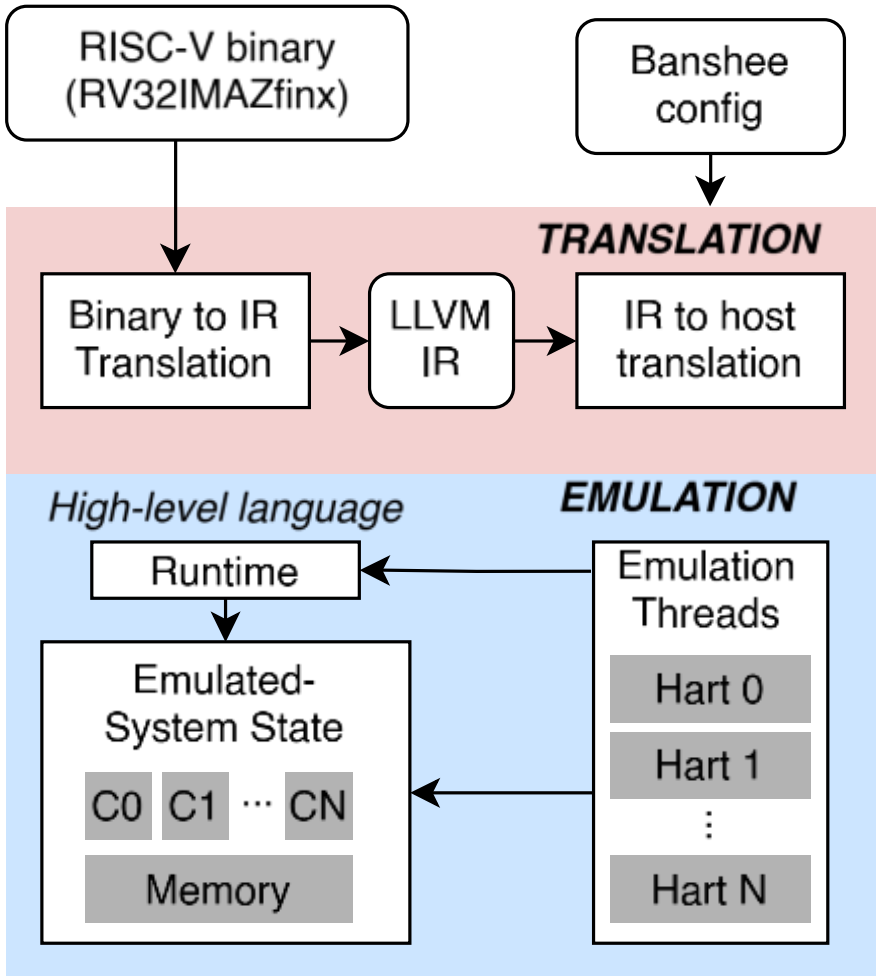




github.com/pulp-platform/mempool

Open Source!

Banshee



github.com/pulp-platform/banshee

- **Inputs:**
  - RISC-V guest binary
  - Configuration (num-cores, mem-size, instr-latencies, …)
- **Translation:** creates the host binary with LLVM infrastructure
- **Emulation:** runs the host binary, simulating the guest architecture

(Parallelized → spawn one core per host thread)

Banshee timing-model

1. Assign **static latency** to each instruction via configuration
2. Bookeep destination registers of in-flight instructions in a **scoreboard**
3. Associate a down-counter to destination registers
4. Add leftover latency to cycle-count for next instruction with a dependency

```
fmul.s t10, t10, t11;
fadd.s s0, s0, t10;
```

✓ Good for arithmetic instructions

⍰ Approximation for memory instructions

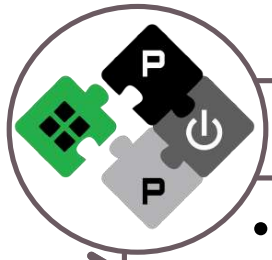
- No contentions in shared interconnects
- No priorities of atomic instructions

HW-in-loop with Banshee

Repeat on randomly generated data for Monte-Carlo simulation of the channel

Python model

- Generate TX-data
- Map bits to QAM-symbol
- Simulate the wireless channel



Banshee

- Translate binary
- Simulate MIMO-MMSE
- Dump results

LLVM 12.0

Compile RISC-V binary

mmse.h
mmse.c
main.c

Python model

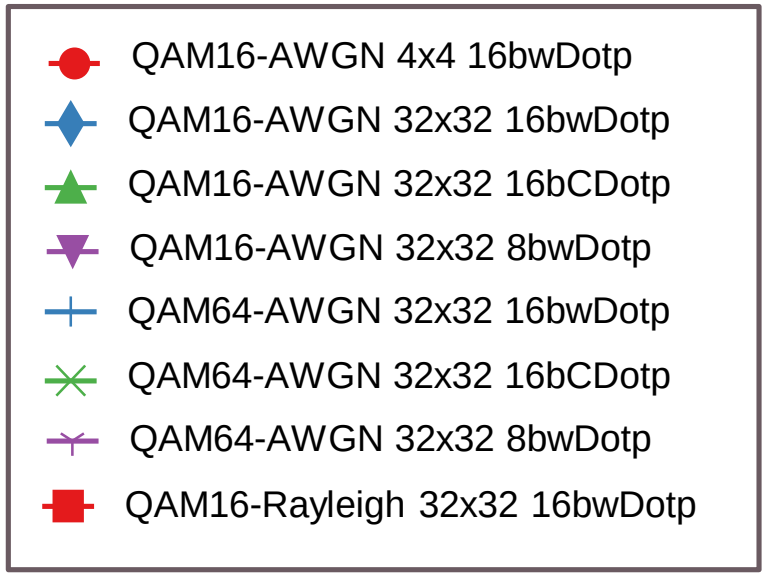
- Map QAM-symbol to bits
- Evaluate metrics (e.g. BER)

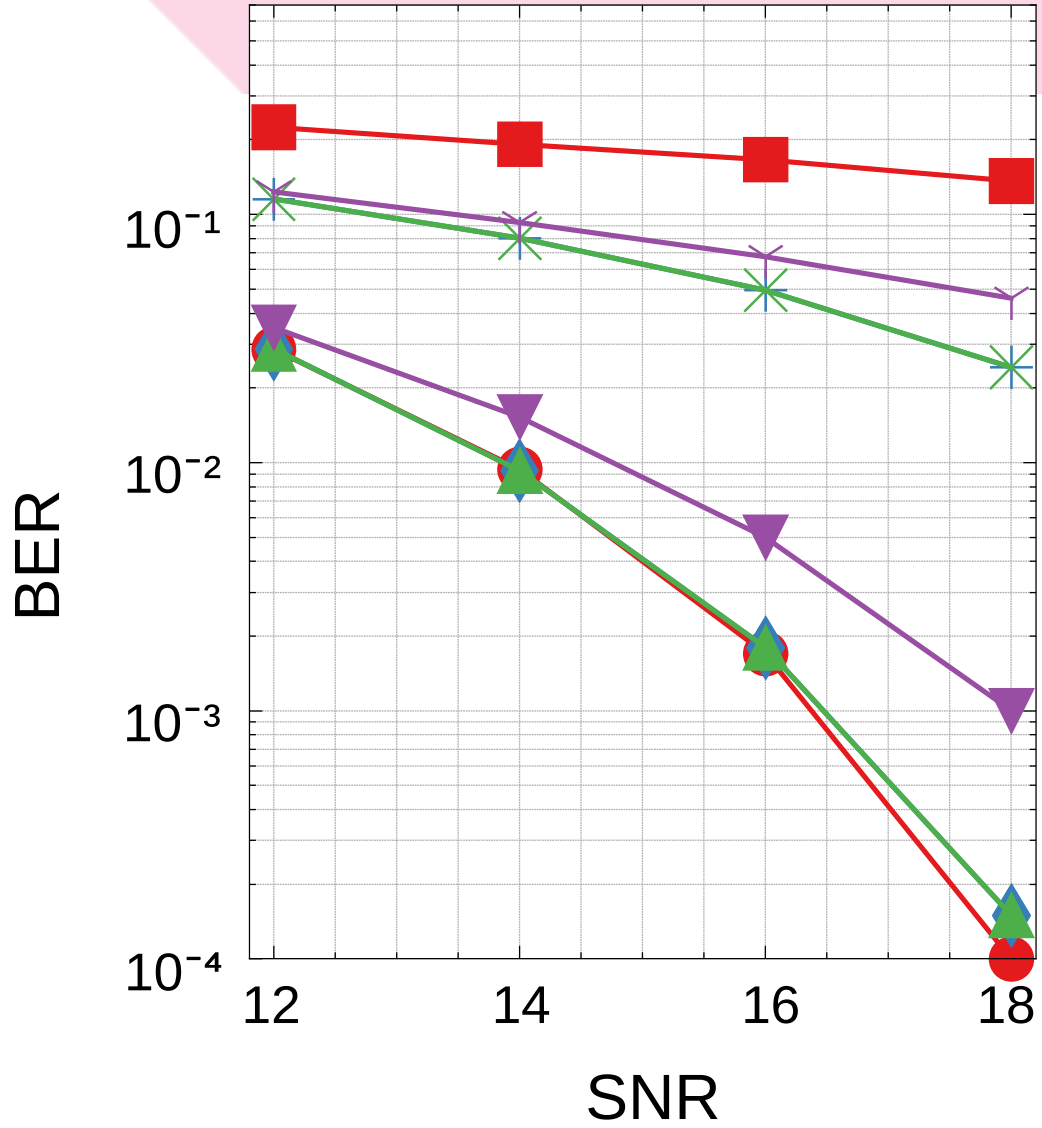
Flexible: MIMO-MMSE HW-SW co-design

Design-space exploration:

MIMO size, Arithmetic precision,

Modulation, Channel



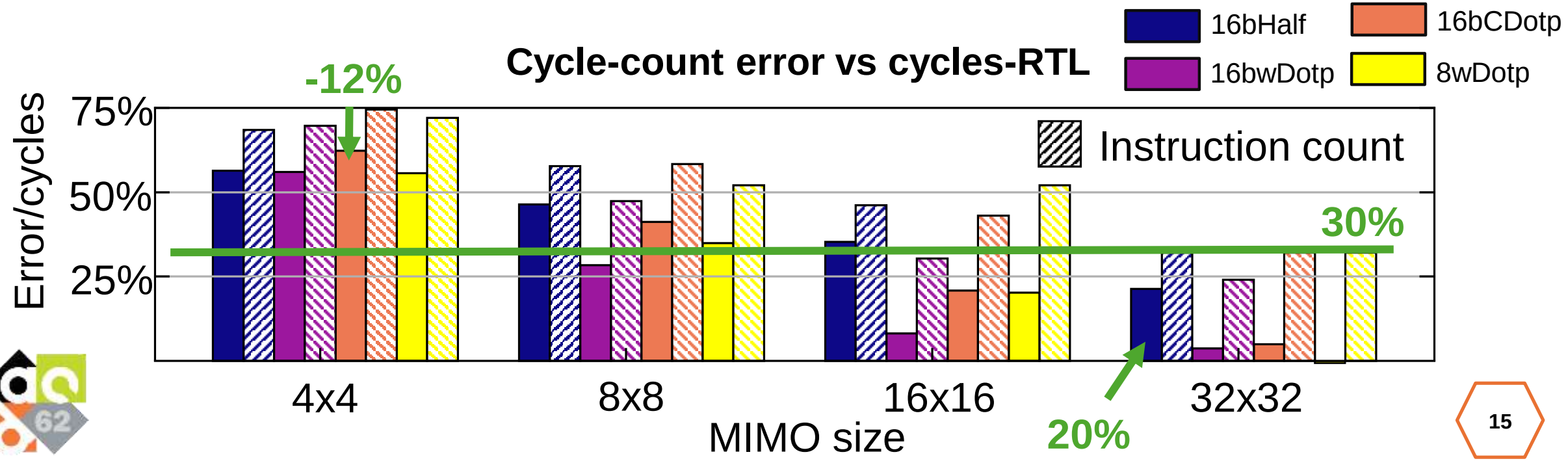


Timing: trade-off cycle-count error/speed

Assigned static-latency does not model priorities of atomics (used in synchronization)

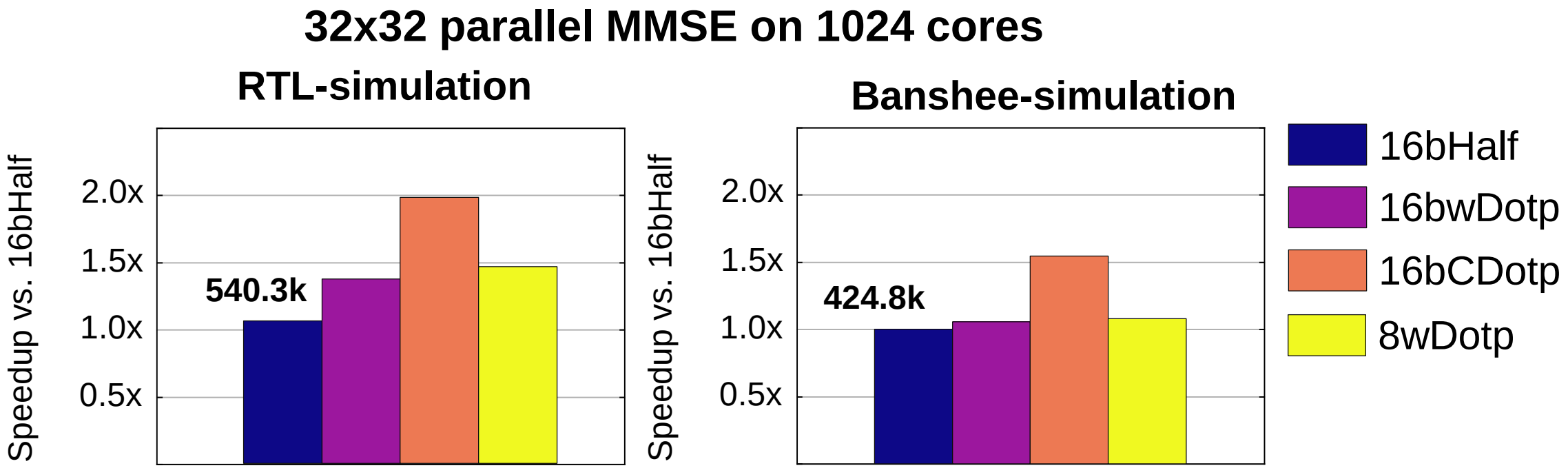
Parallel MMSE on 1024 cores

- Average **30%** error (0.6%-62%, for 32x32 MIMO **0.6%-20%**)
- In worst-case (4x4 MIMO) **12%** improvement wrt instruction-count



Timing: relative runtime preserved

Gives information for design-space exploration on different arithmetic precisions:

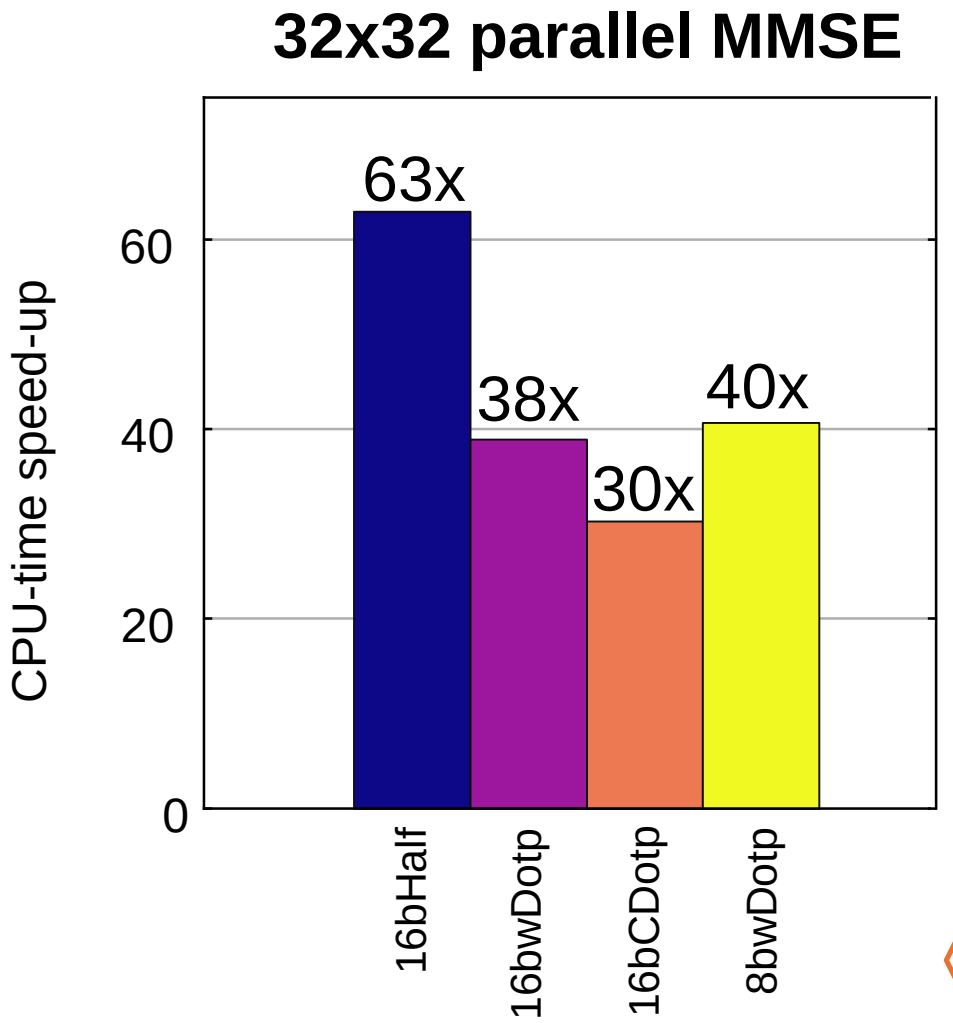


16bCdotp < 8bwDtop < 16bwDotp < 16bHalf

Fast: Banshee is 63X faster than RTL-sim.

Each Snitch runs an MMSE problem

- **CPU-time** on 128-core **AMD EPYC-7742**
  - Questasim-2022.3 (single-core) **36h**
  - Banshee (parallel) **< 35min**
  - **63x speedup**
- **Wall-clock time** is **< 2min:45s**



Fast: Monte-Carlo @ 9.44s/iteration

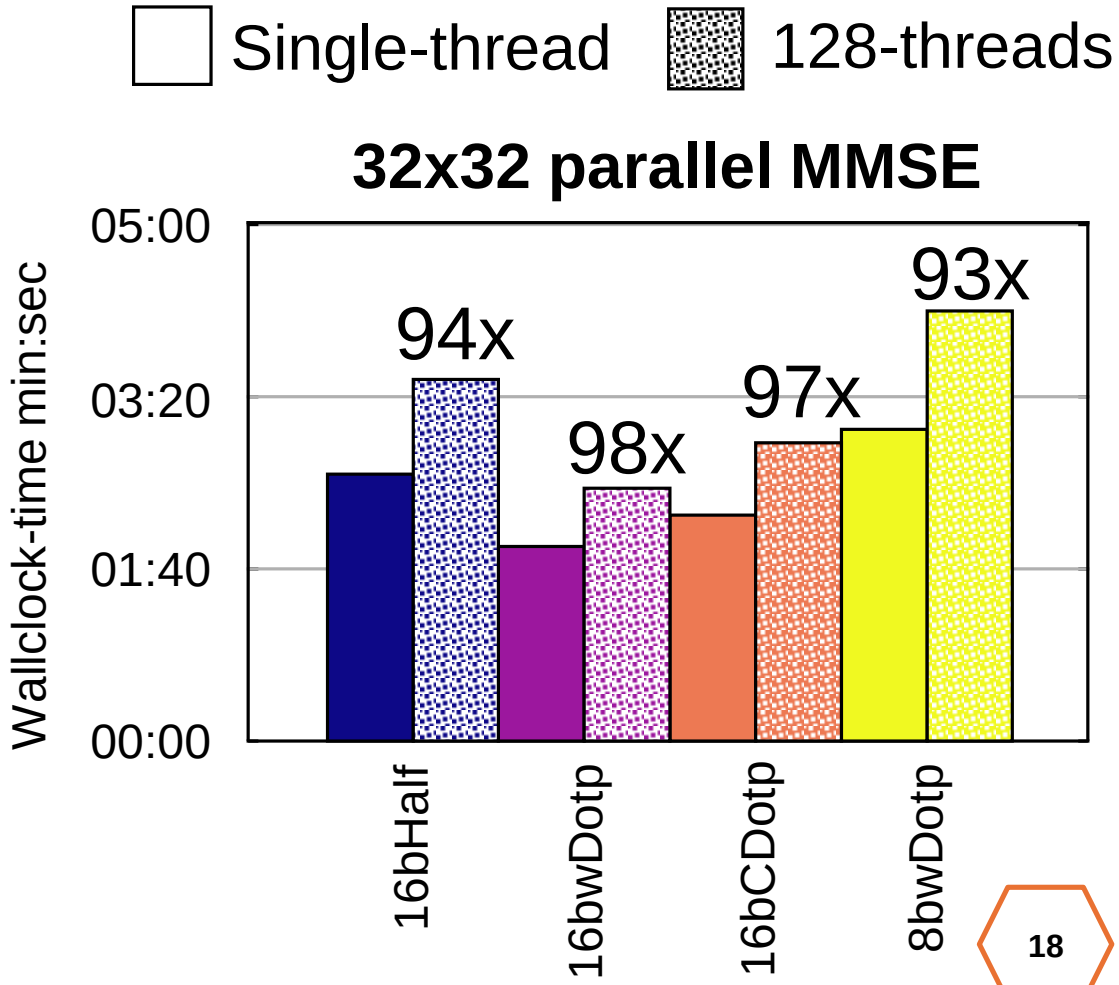
Snitch runs 1638 MMSE / OFDM-symbol

Single-thread Banshee simulation

- Wallclock-time per symbol **< 3min**
  (**9.44s** for 4x4 MMSE)

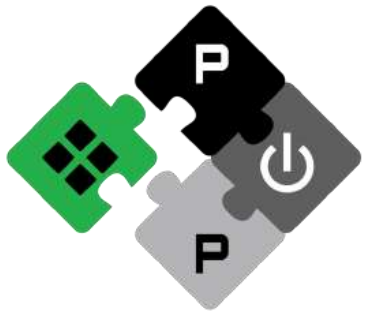
Different symbols parallelized on different host threads

- Parallelization speedup up to **98x**
  (**122x** for 16x16 MMSE)



Fast SBT-based simulation of 6G-trasceivers

- Functionally correct SBT-based simulation of programmable 6G-transceivers
- A simple timing model yields **30% average error** and up to **12% improvement** wrt instruction count in **estimating the runtime**
- The lightweight simulator is suitable for fast **Monte-Carlo simulation @9.44s per OFDM-symbol iteration**



github.com/pulp-platform/banshee

github.com/pulp-platform/mempool