Evaluation of NOEL-V arithmetic performance

Martin Daněk , daiteq

Overview

- Motivation
- NOEL-V configurations
- Benchmarks
- Results
- Discussion

- Packed floating-point types

Motivation

Evaluate the maturity of 64-bit NOEL-V:

How far is its floating point performance from other alternative space-qualified systems?

- AT697, LEON2, LEON3, GR740, LEON5
- PolarFire SoC

NOEL-V configurations

NOEL-V

- 64-bit configurations - HPP, GPP, MIN (w/ FPU)
- Single core x Multi core – RV64IMAFD, RV64IMAFD_SMP

HPP	GPP	MIN
Dual issue	Single issue	Single issue
Large cache	Large cache	Small cache
Large BHT	Large BHT	Small BHT

Sampled NOEL-V systems

1-core system	3-core system	4-core system
HPP / MIN	HPP	GPP
Dual-issue / Single-issue	Dual-issue	Single-issue

daiteq

NOEL-V FPUs

FPU	nanofpunv	daiFPUrv	GRFPUnv
Size	Small	Medium	Medium-big
Computation	Iterative	Parallel blocking	Parallel pipelined
License	Cobham Grislier GPL	daiteq	Cobham Gaisler
Performance	Low	Medium	High

Evolution of the measurements

GRLIB version	FPU	L2 Cache	Performance
2020.4	nanofpunv	No	Baseline
2021.2	nanofpunv daiFPUrv	Yes	No change
2022.2	nanofpunv daiFPUrv GRFPUnv	Yes	**10% improvement**

Alternative systems

System	Technology	Cores	IU stages	FPU	L2 Assoc	L2 Way size	Total L2 size
PolarFire SoC	ASIC	4-core U54	5	Yes	4x 16-way	32KB	2MB
LEON2	ASIC/FPGA	1-core	5	Meiko (AT697) daiFPU			
LEON3	FPGA	4-core	7	GRFPU			
GR740	ASIC	4-core	7	GRFPU	4-way	512KB	2MB
LEON5	FPGA	4-core	8	GRFPU5	4-way	128KB	512KB

daiteq

Benchmarks

Single-core	
Paranoia	Standard compliance test for IEEE 754 Std.
Whetstone	Floating-point - mix of workloads
Linpack	Gaussian elimination - MUL ADD
Stanford	Floating-point/integer mix of workloads
Multi-core	
CoreMark	Integer - standard distribution Floating-point - std distribution incomplete, daiteq completed
CoreMark-Pro	5 integer workloads 4 floating-point workloads
FPMark	10 floating-point workload prototypes Further parameterised: precision, data size

Note: CoreMark, CoreMark-Pro and FPMark are maintained and licensed by **EEMBC** -

EDN Embedded Microprocessor Benchmark Consortium

EEMBC Benchmarking Framework

Key terms:

- Kernels (benchmarks)
- => Workloads

- MITH - Multi-Instance Test Harness
  - Manages execution of a number of contexts using a number of workloads
  - Target-independent, but it requires POSIX threads or other equivalent parallel execution model

  - Our experiments: 1..12 contexts, 1 worker per context
  - LEON and NOEL-V tests: RTEMS (POSIX)
  - PolarFire SoC tests: Linux (POSIX)

EEMBC CoreMark-Pro

- 5 integer workloads:
  - cjpeg
  - coremark
  - parser
  - sha
  - zip
- 4 floating-point workloads
  - Gaussian elimination
  - Livermore loops (basic)
  - Neural network
  - FFT

EEMBC FPMark

10 floating-point tasks

- atan
- Black-Scholes
- Horner
- FFT
- Linpack (enhanced)
- Livermore loops (enhanced)
- LU decomposition
- Neural network
- Ray tracing
- (x+1)^x

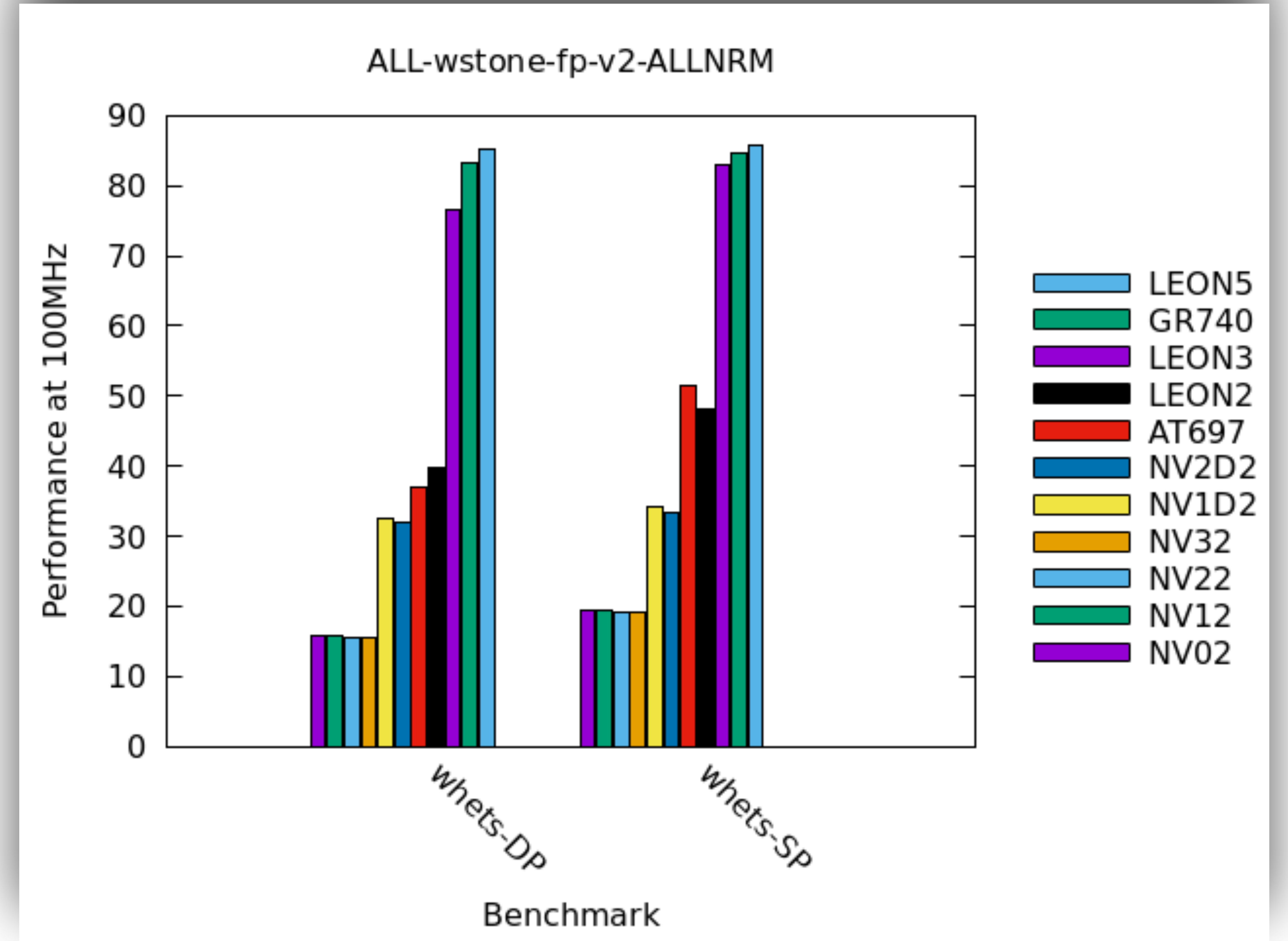
Each task is further parameterised to form a workload:

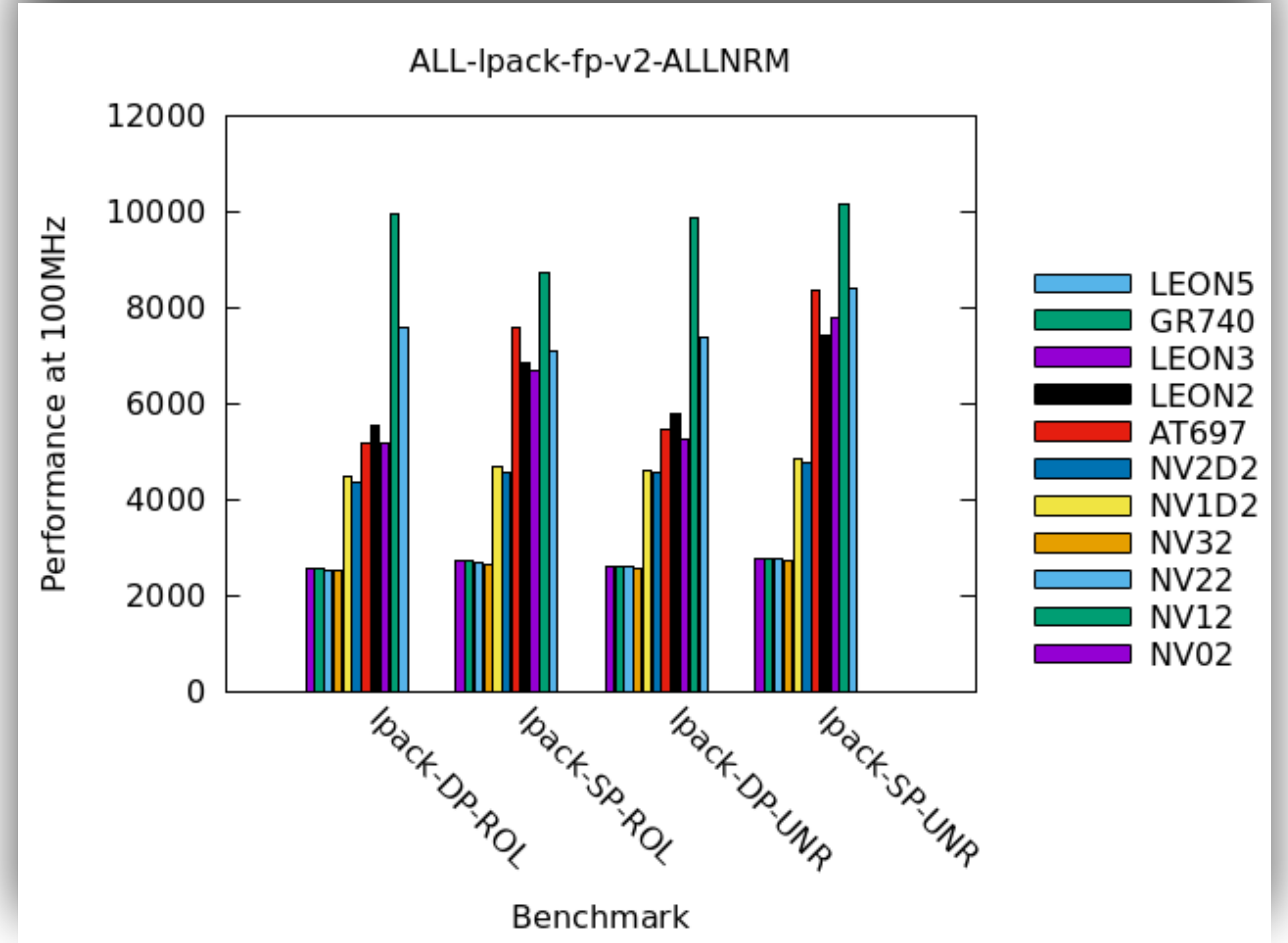
- Precision: SP or DP
- Size: small, middle, big

Theoretically 10x2x3 workloads

Practically 47 workloads

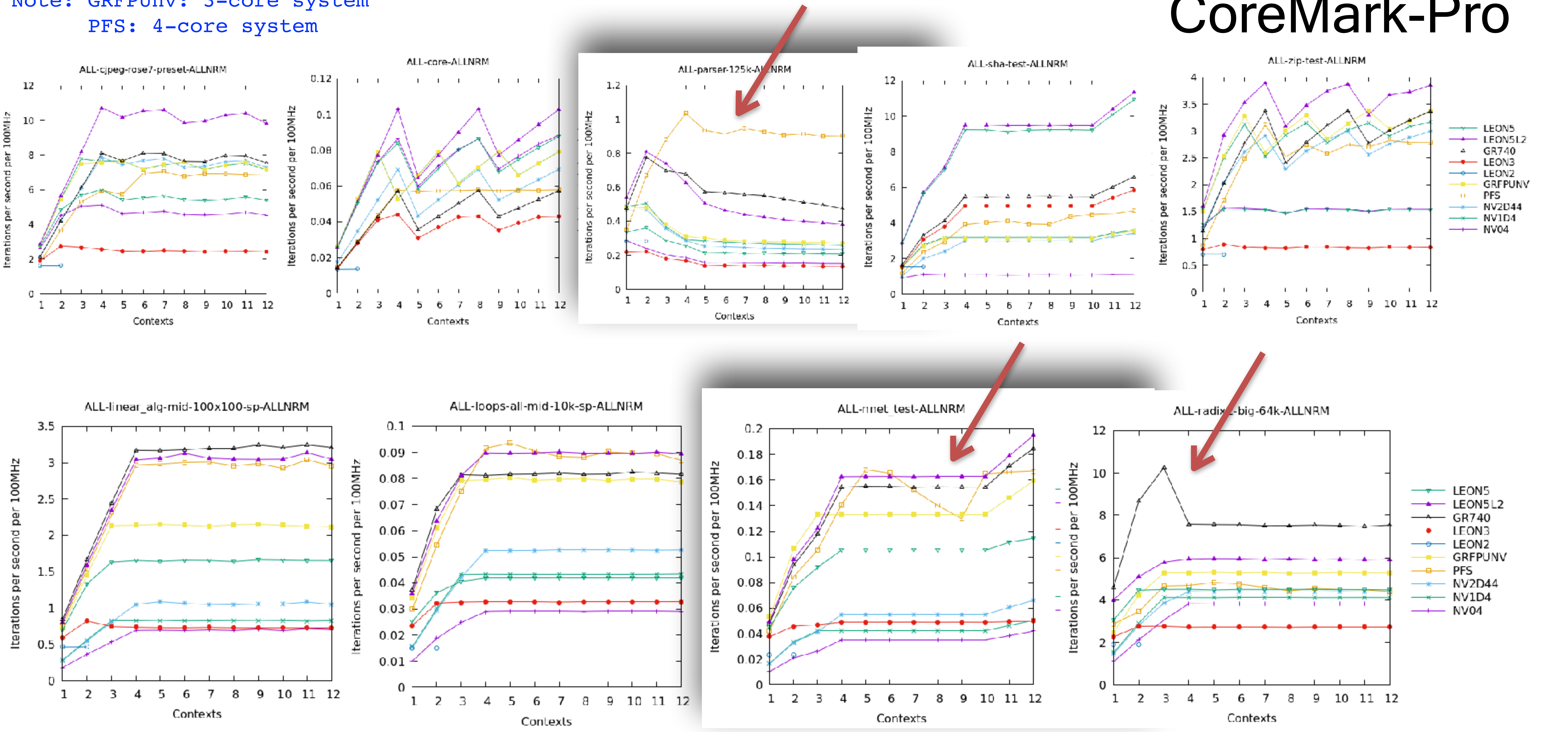
Whetstone, Linpack



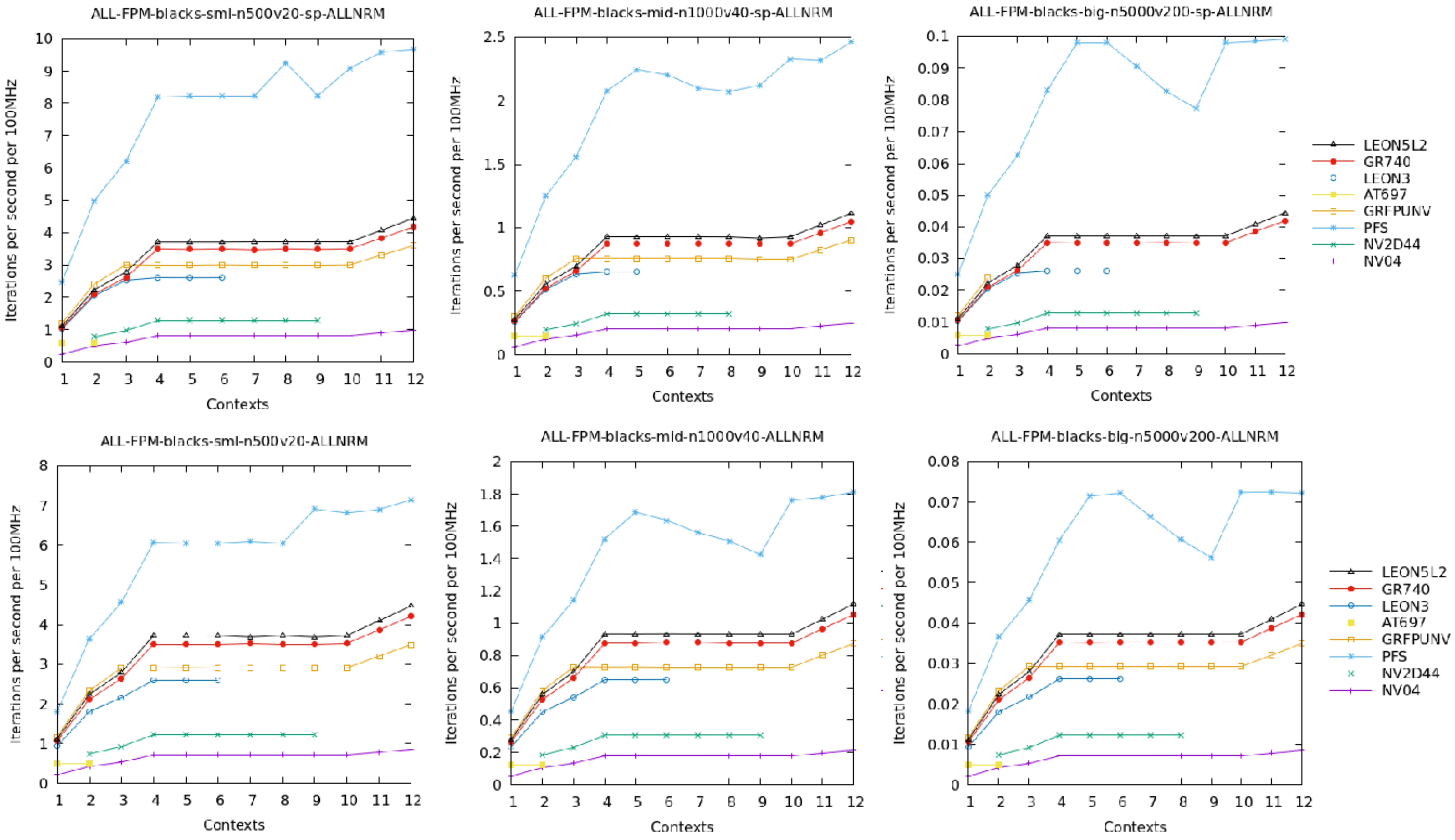


CoreMark-Pro

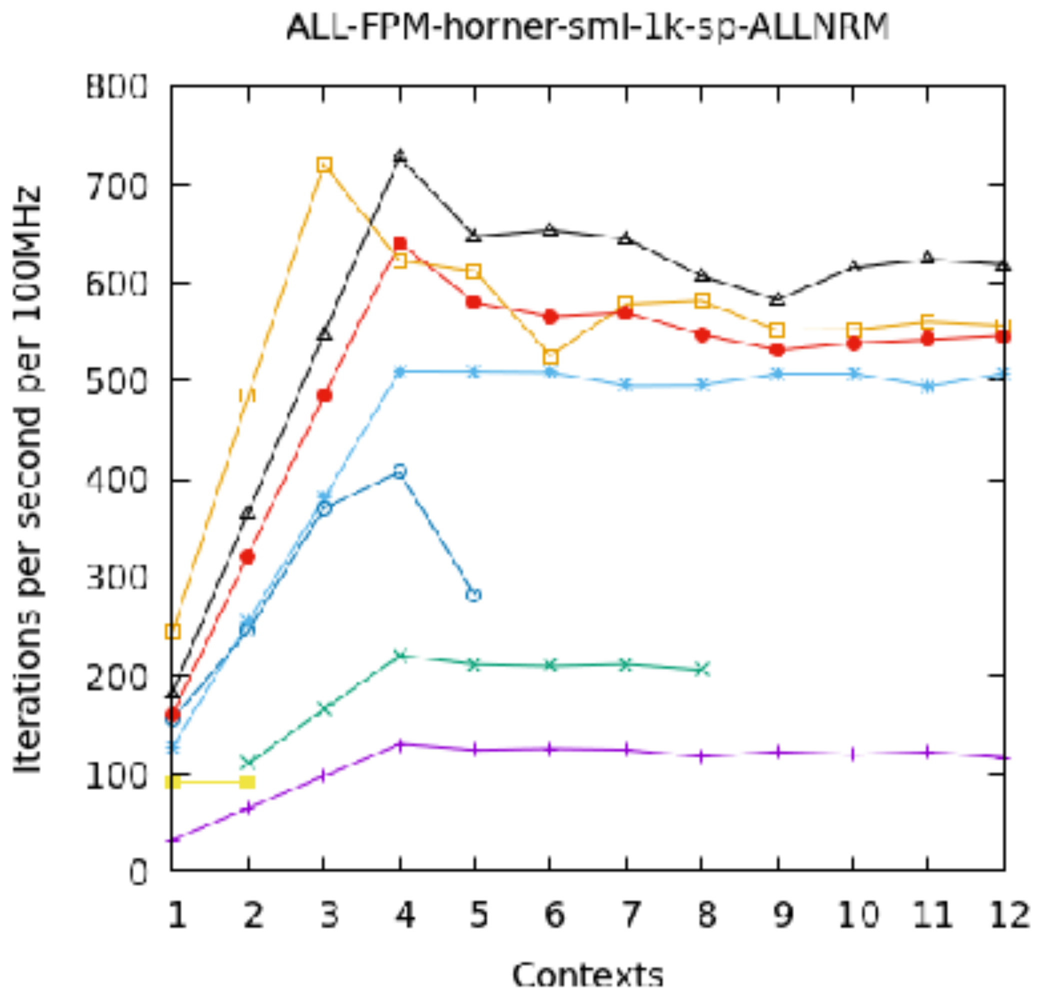
Note: GRFPUnv: 3-core system
PFS: 4-core system

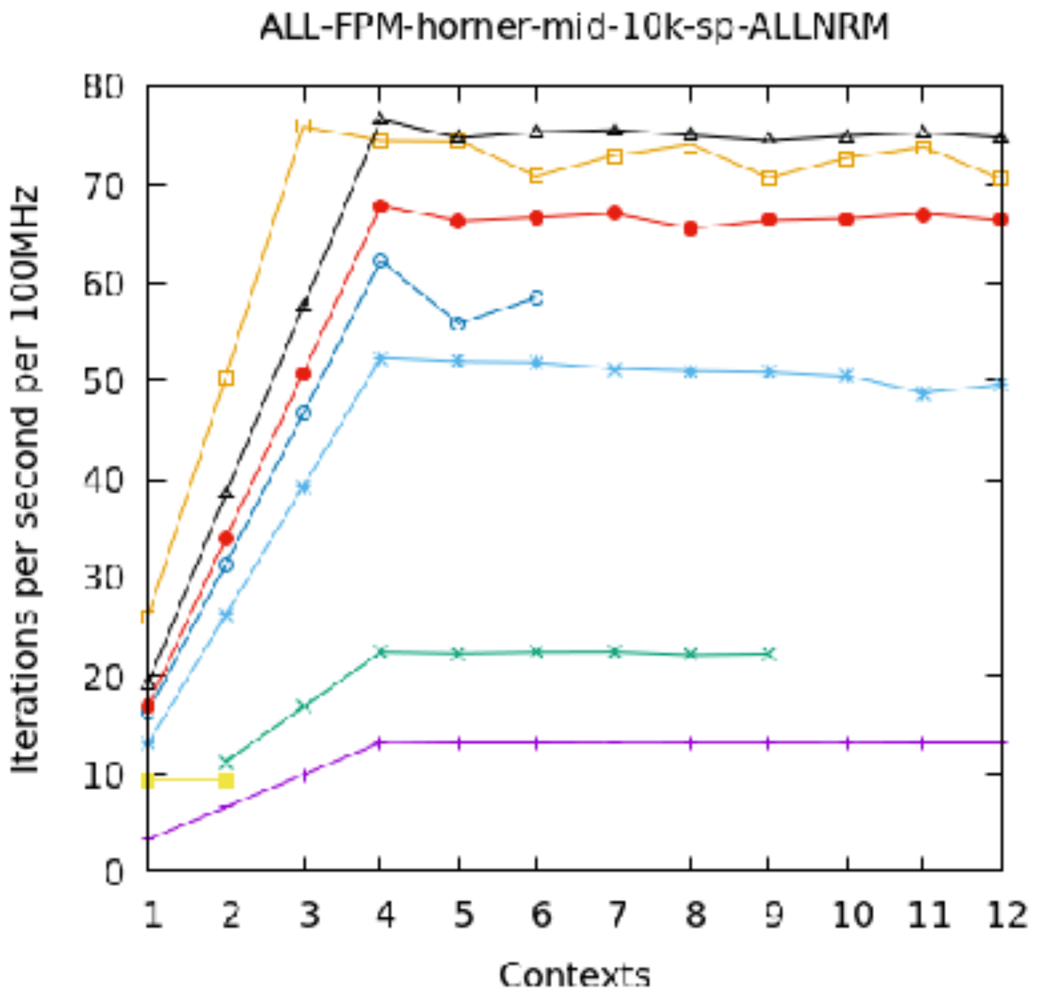


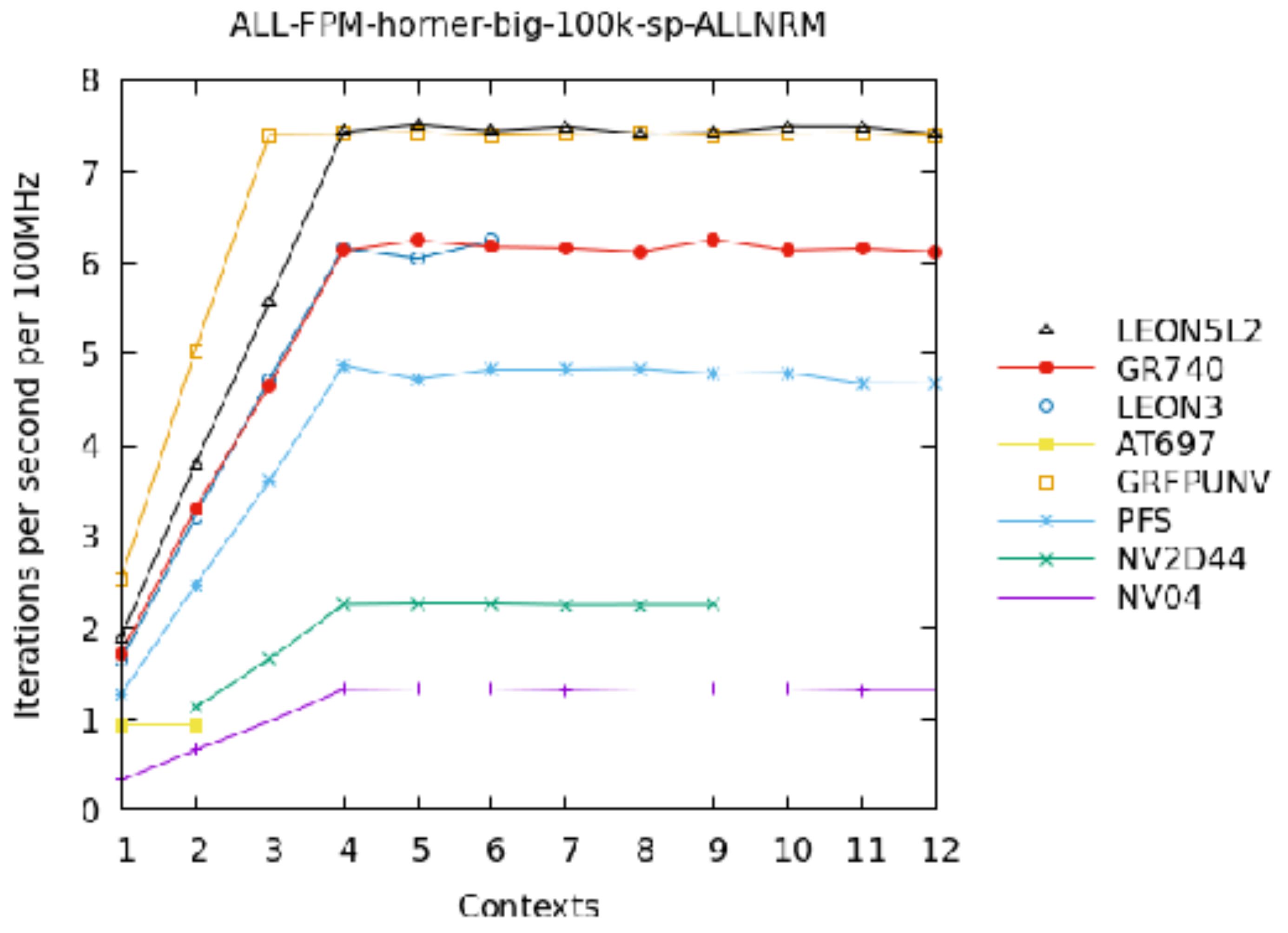
FPMark - Black-Scholes

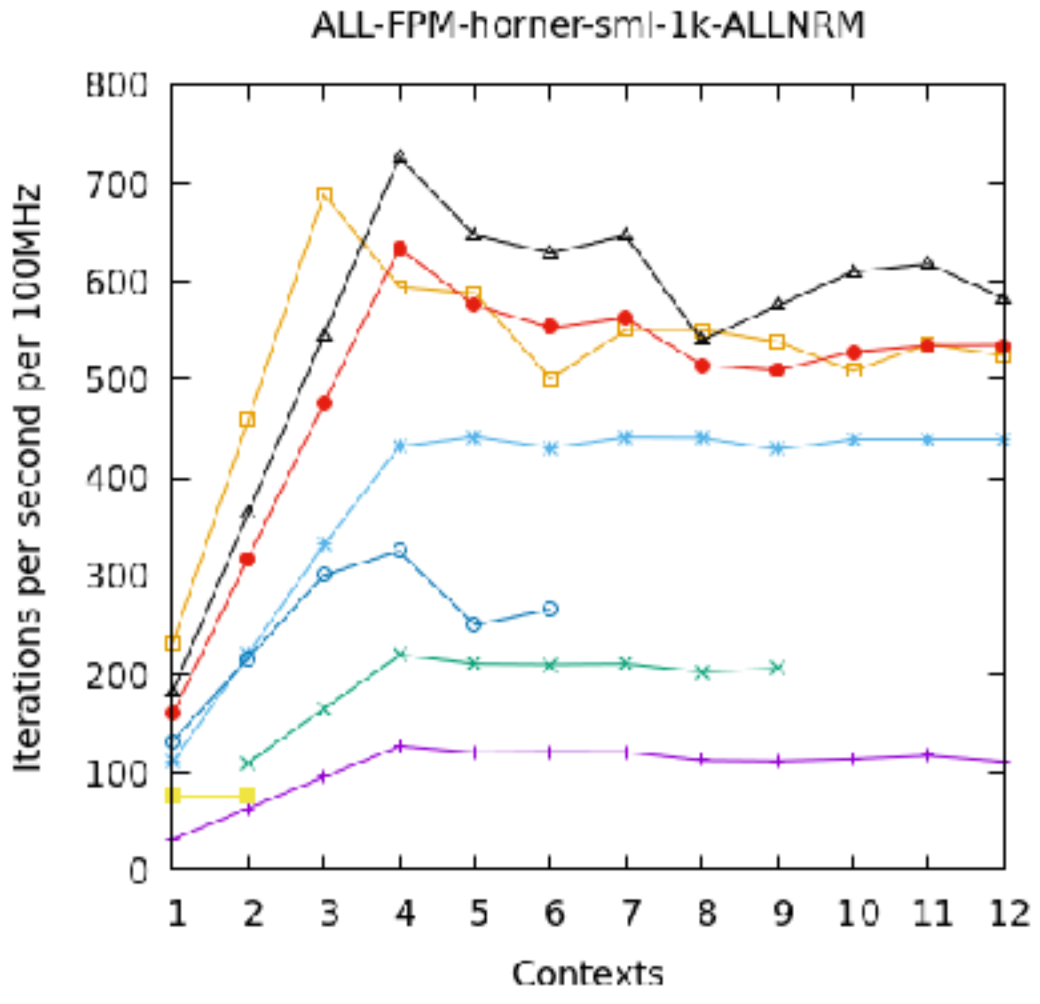


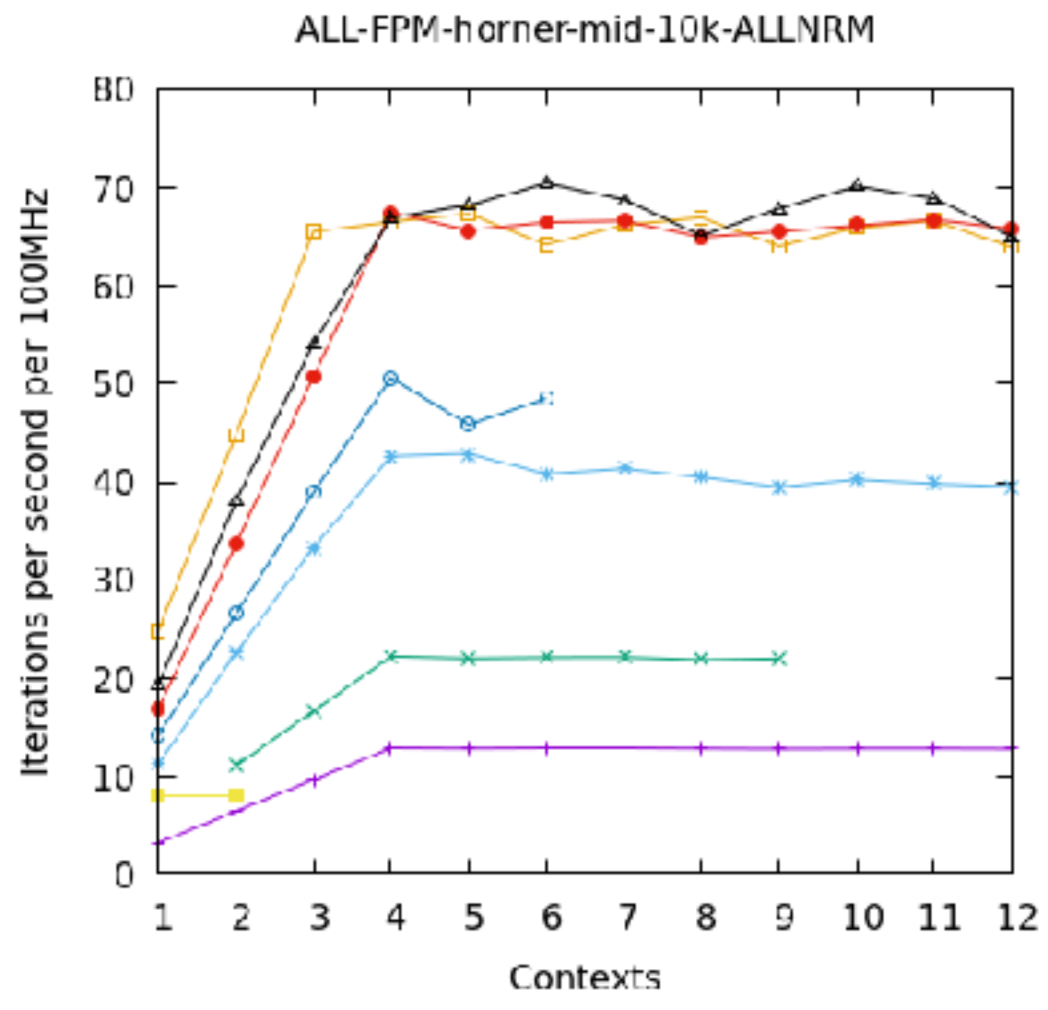
FPMark - horner

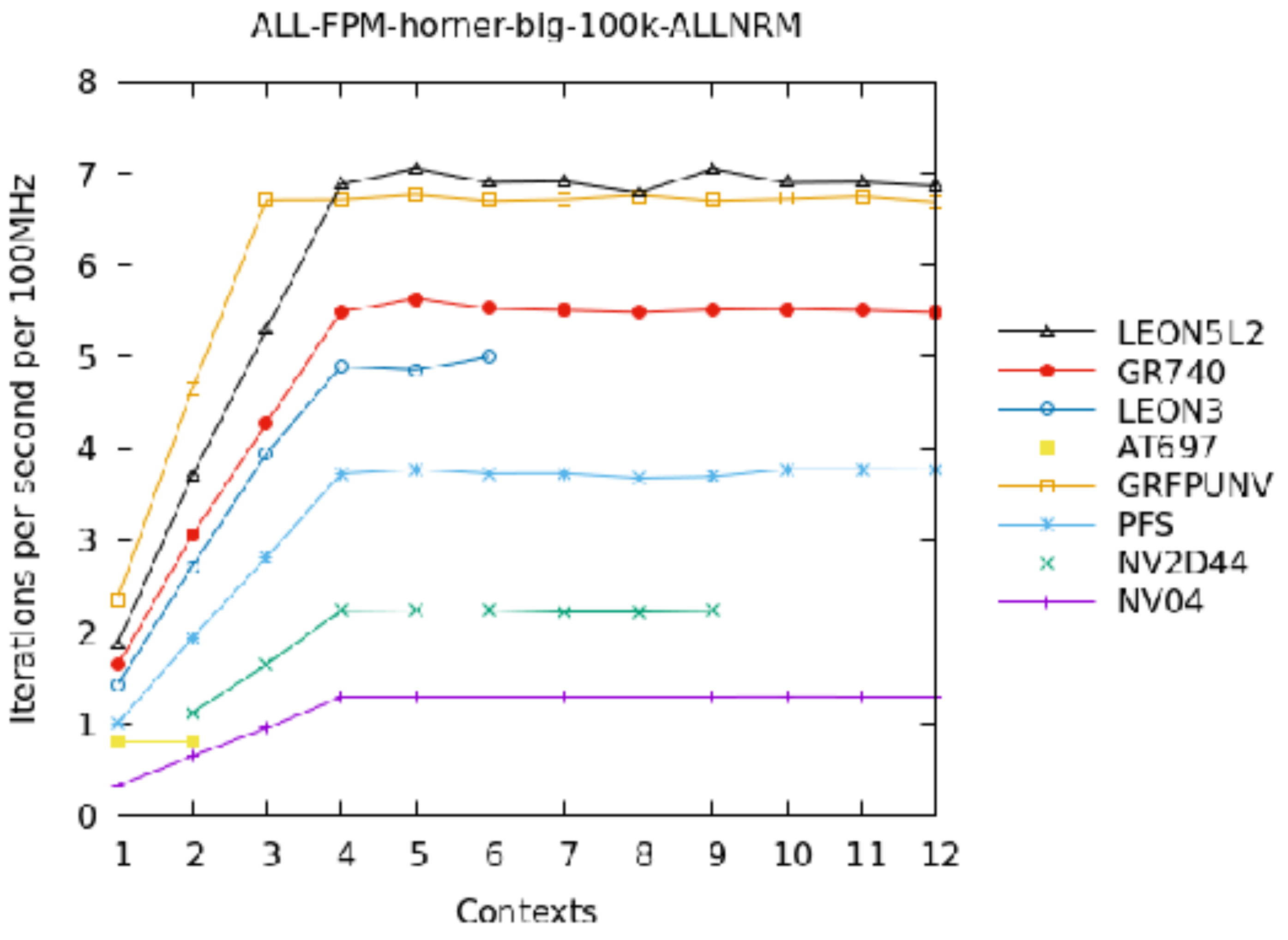




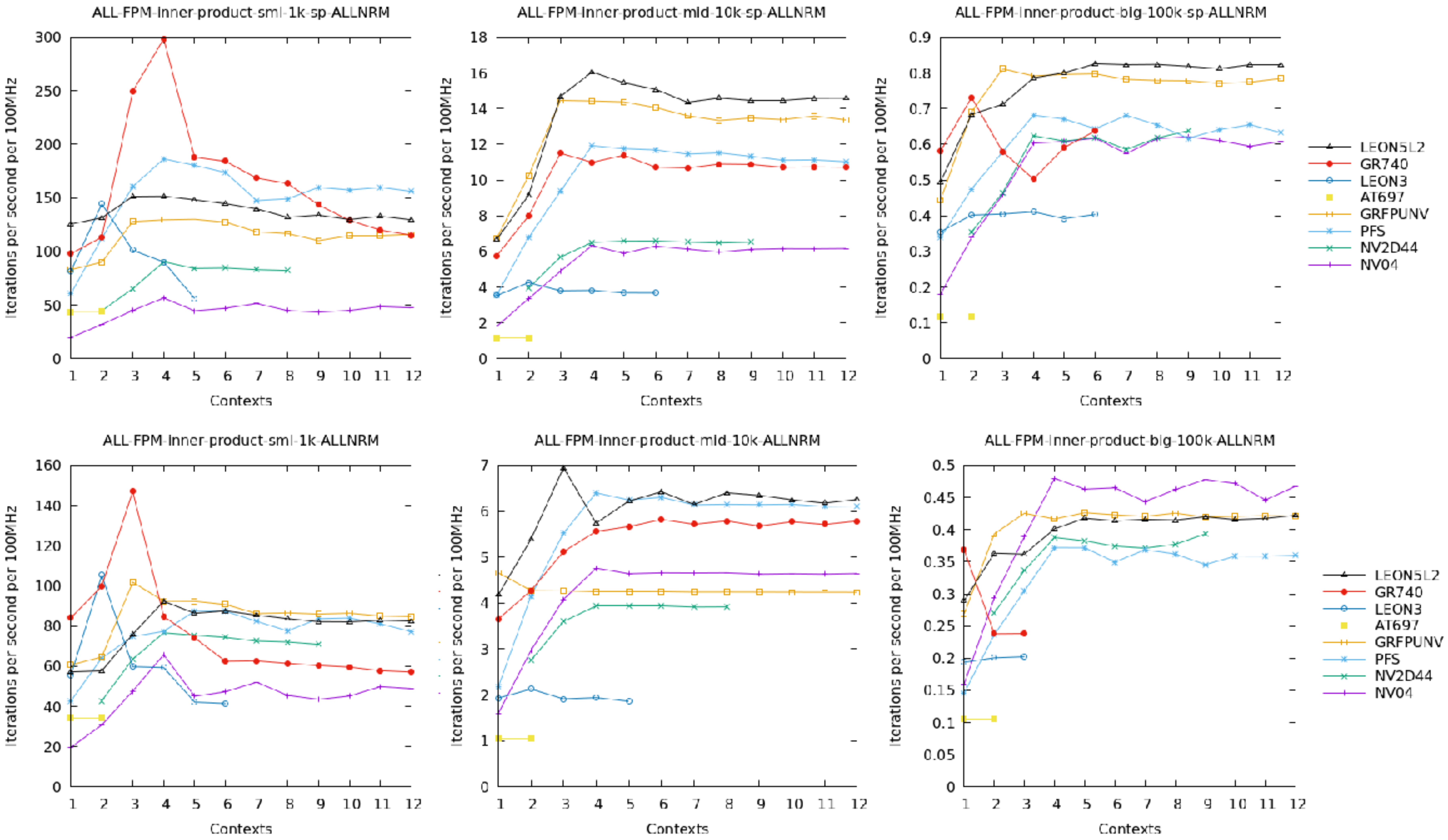






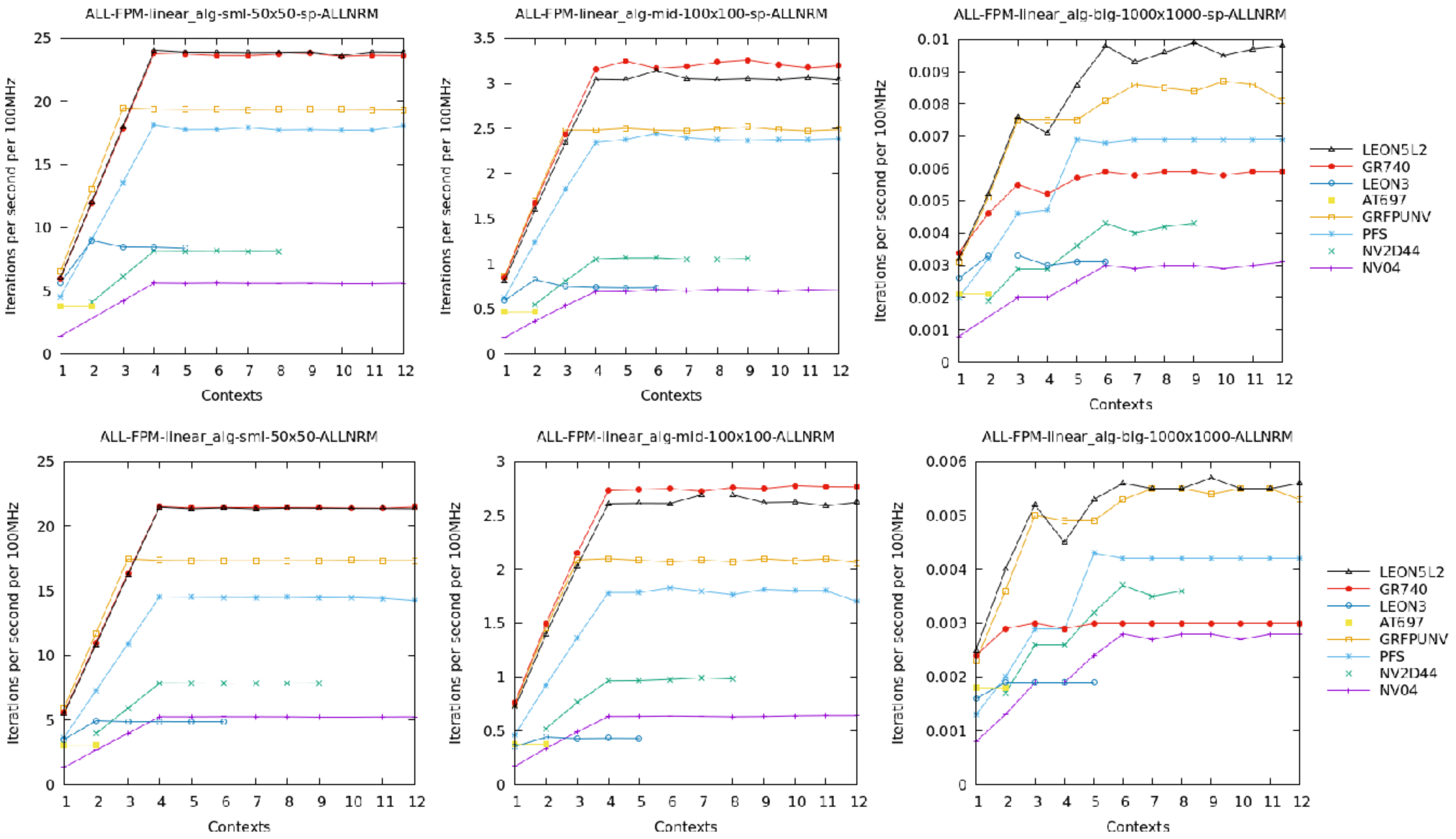


FPMark - inner-product



daiteq

FPMark - linear_alg



Discussion

The best “GRLIB" performance: GRFPU + L2 Cache

CoreMark-Pro:

- Integer workloads: NOEL-V w/ L2 Cache better than PFS.
- Key factors for NOEL-V:
  - With / without L2 Cache
  - Dual-issue / Single-issue pipeline

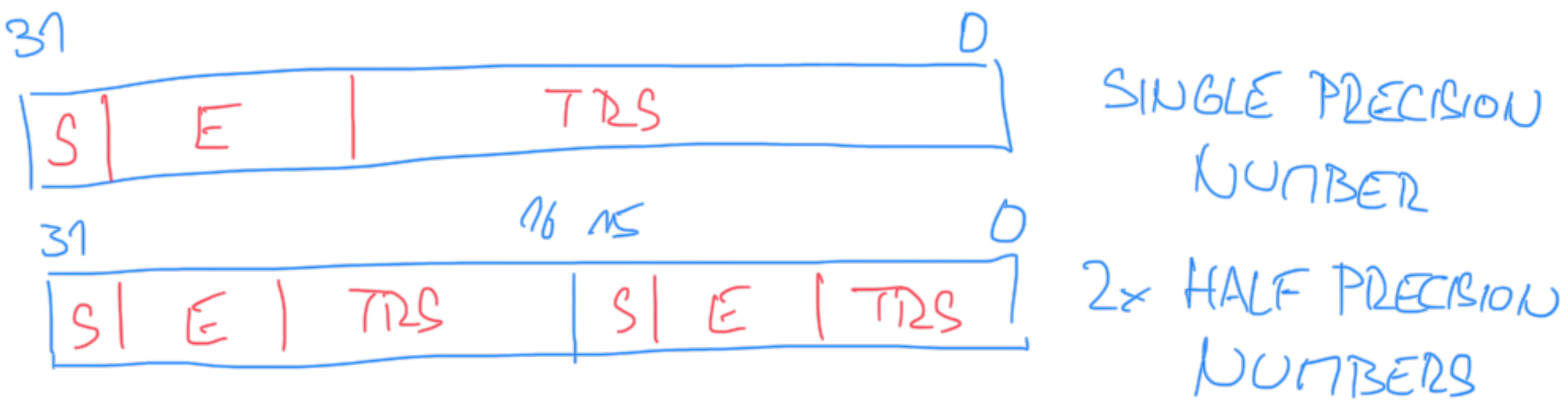
FPMark:

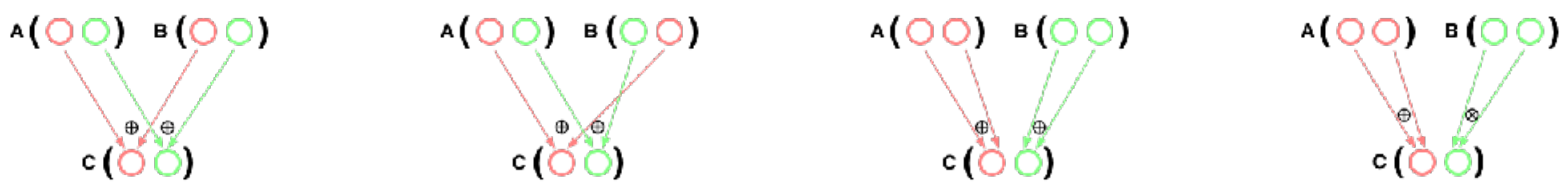
- Floating-point workloads: NOEL-V + GRFPUnv ≈ PolarFire SoC
- Note: Having a dual-issue pipeline does not impact floating-point performance.

Explanation of differences in NOEL-V and PFS performance - RTEMS vs linux:

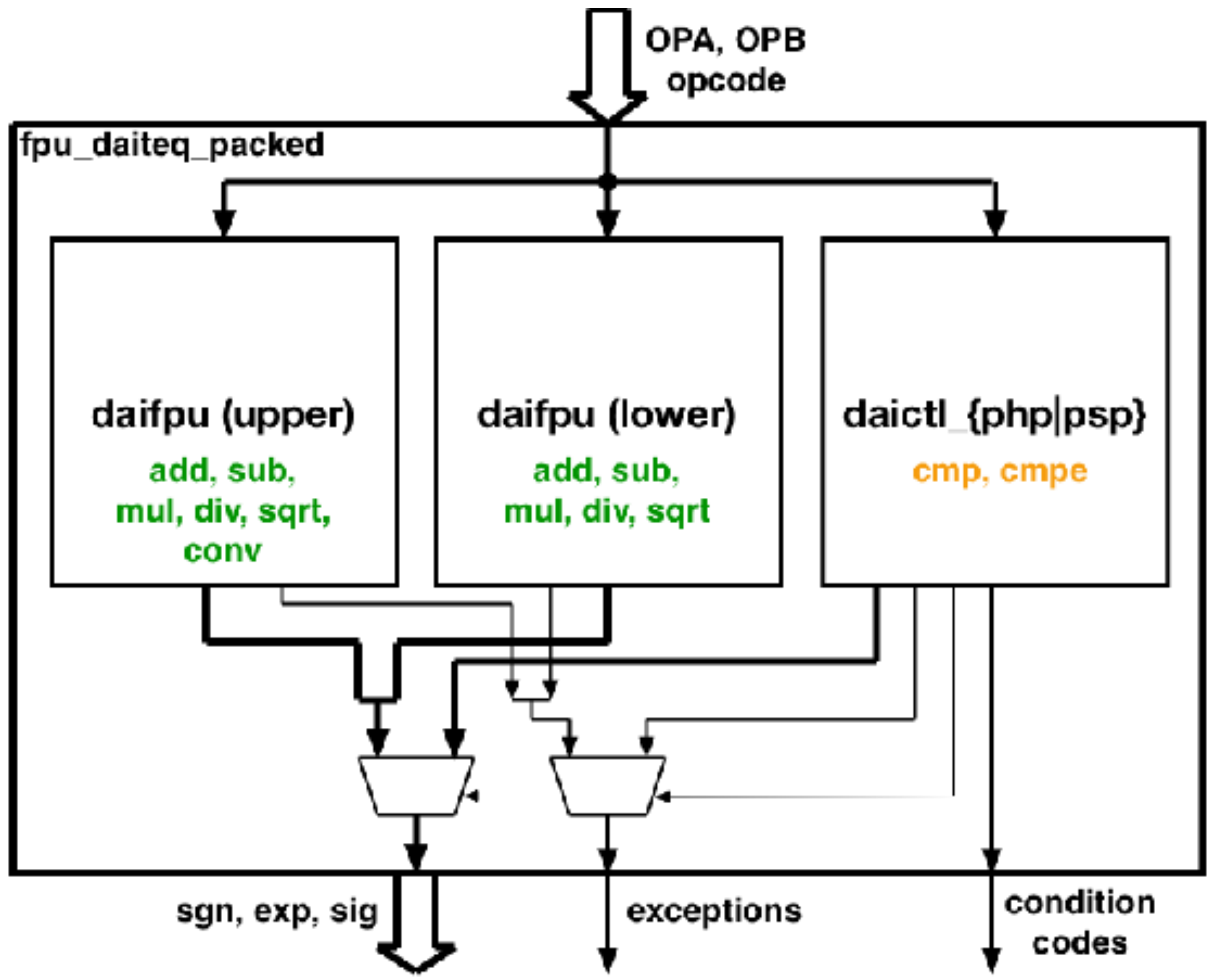
- Different task scheduling
- Different memory management

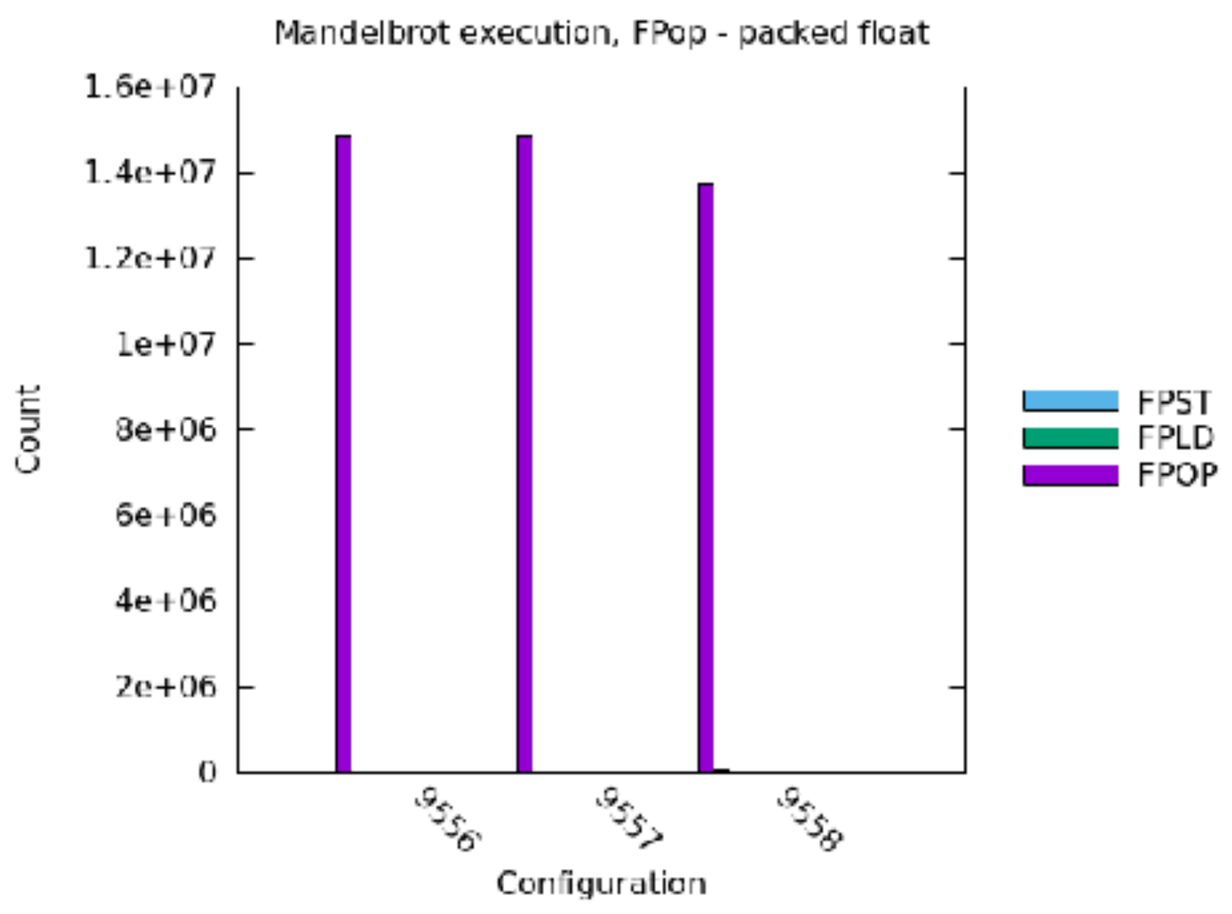
Additional performance - packed floating-point types

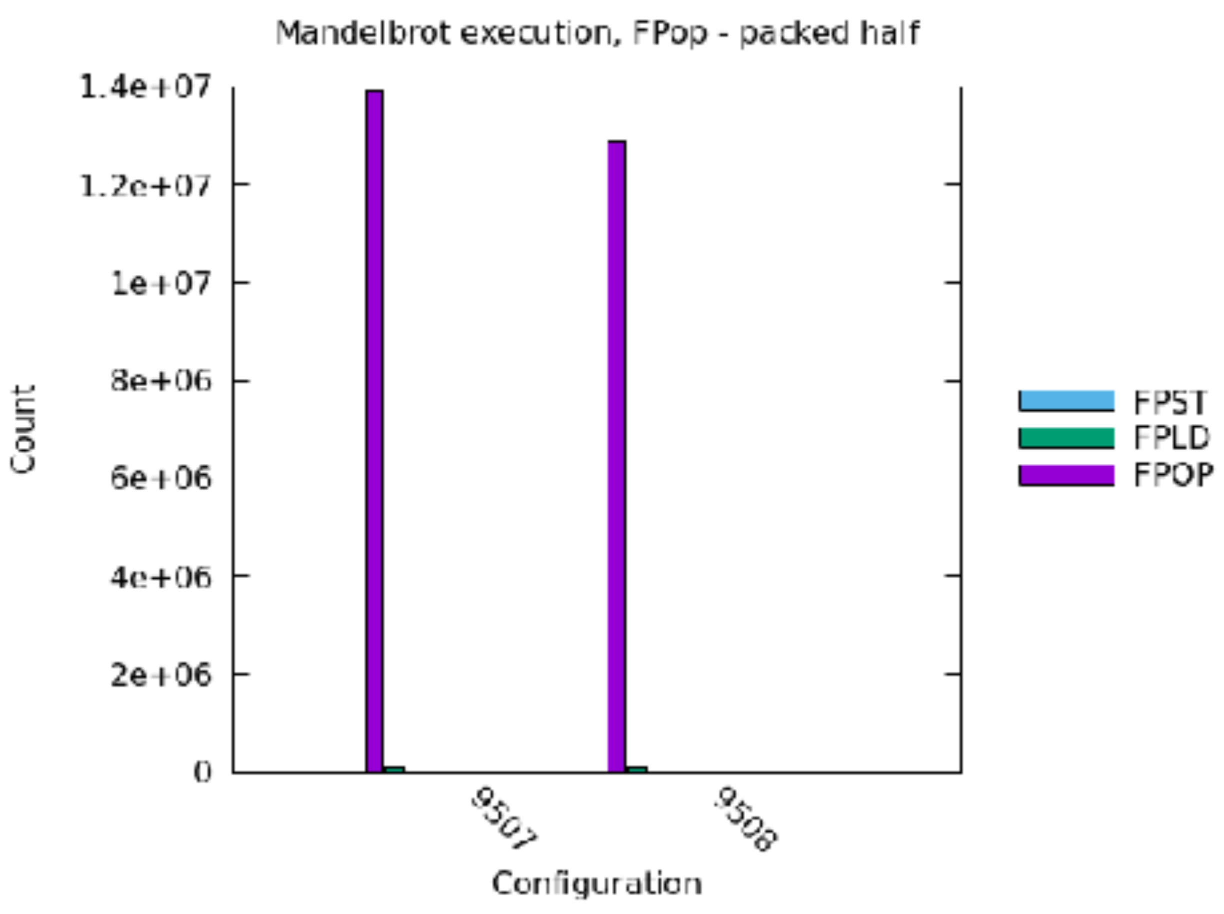


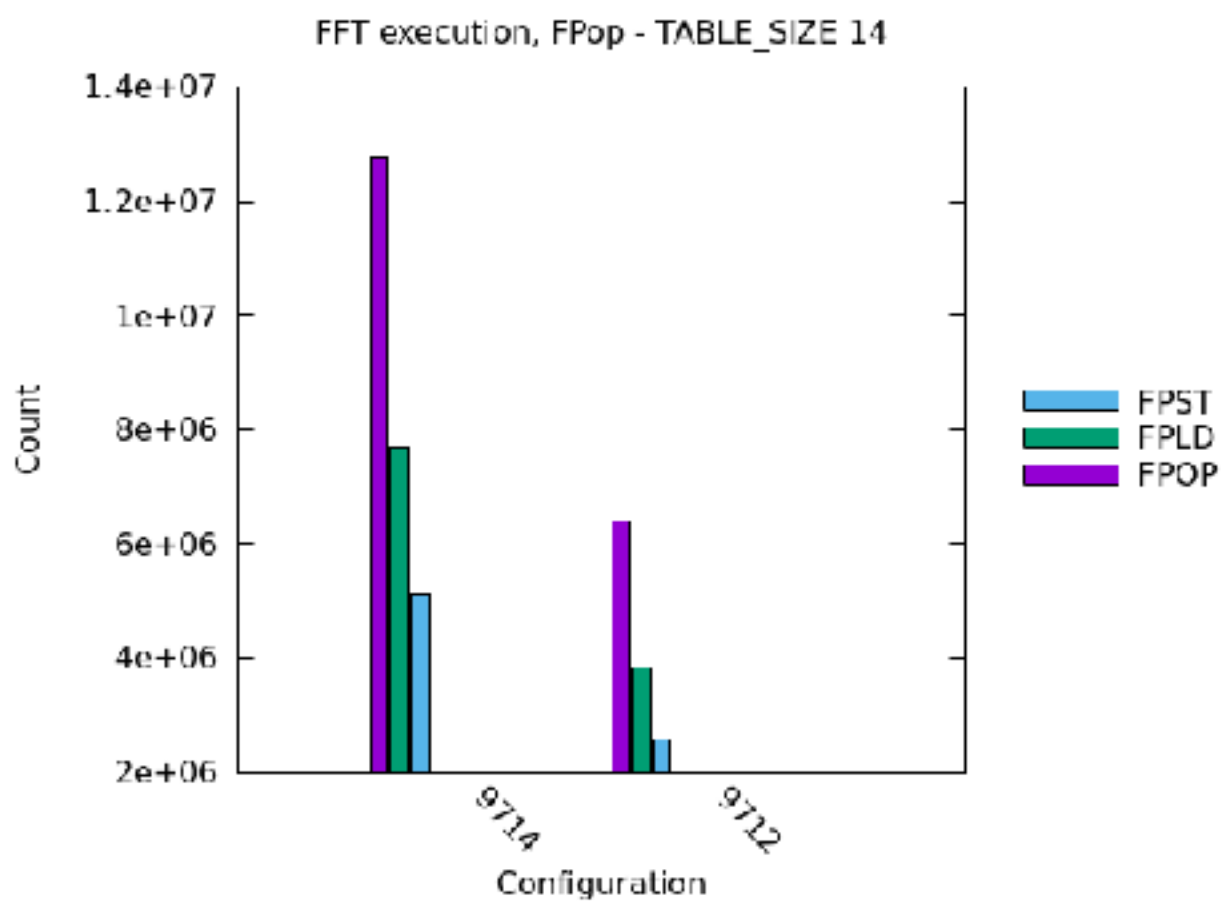


MAP | CROSSED MAP | REDUCE | SUBADD









THANK YOU