Mirage Cores

The Illusion of many Out-of-Order Cores Using In-order Hardware

Shruti Padmanabha
University of Michigan, Ann Arbor

Andrew Lukefahr
Indiana University

Reetuparna Das
University of Michigan, Ann Arbor

Scott Mahlke
University of Michigan, Ann Arbor

In Proceedings of The 50th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, MA, USA, October 14-18, 2017(MICRO-50).

Presented by: Bernard Pranjic, 20.05.2021

Seminar in Computer Architecture, ETH Zürich

Mentors: Behzad Salami, Kosta Stojiljkovic, Damla Senol Cali

Executive Summary

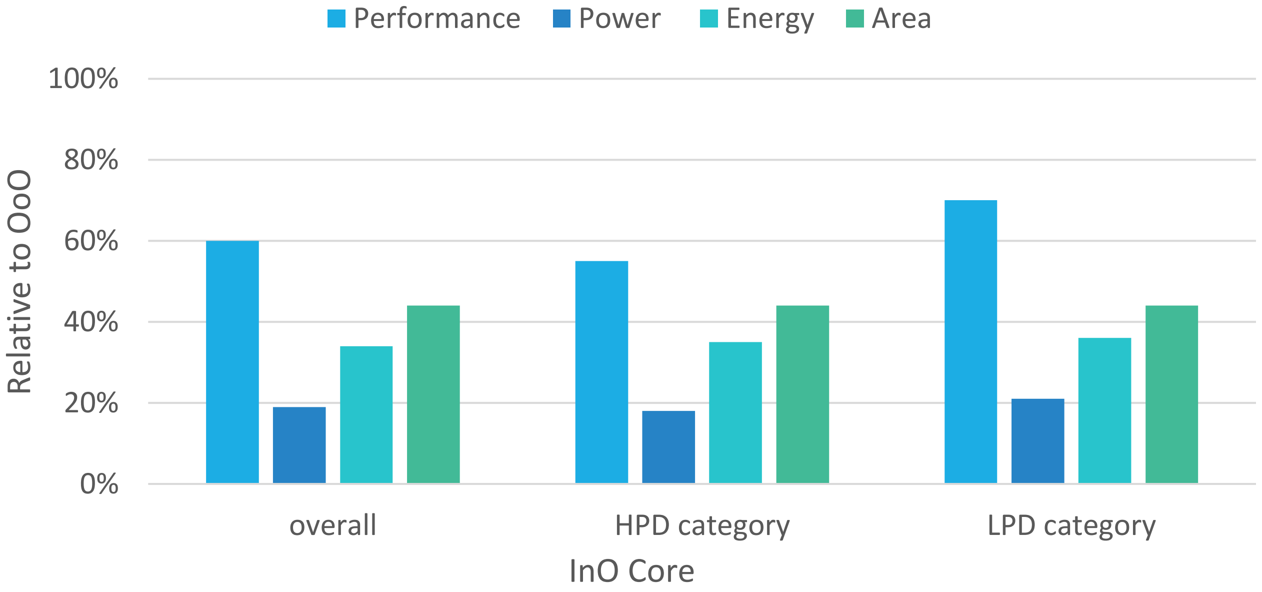
- Problem:
  - Practical power and thermal constraints limit the deployment of homogeneous multicore systems with many big OoO cores
  - Low performance of InO cores limits their widespread usage
- Goal:
  - The goal is to design a Het-CMP with near OoO performance and InO energy consumption
- Idea:
  - The idea is to use clusters of InO cores around one OoO core
  - The OoO core is used as a «scheduler» and the InO cores as «workers»
- Evaluation:
  - The Mirage Core can achieve on average 84% performance of a Homo-CMP, while conserving 55% of energy and 25% of area costs

Overview

- Background, Problem and Goal
- Novelty, Key Approach and Ideas
- Mechanisms (in some detail)
- Key results, Methodology and Evaluation
- Summary
- Strengths and Weaknesses
- Thoughts and Ideas
- Key Takeaways
- Open Discussion

Out-of-Order cores

- Improve latency of programs
- Contain additional HW to reorder instructions to minimize stalls (ROB, RS, LSQ, etc.)
- This increased performance comes at the cost of increased power consumption



Heterogeneous Computing

- Systems contain mixed processor types (e.g. CPUs and GPUs on the same chip)
- Built in logic for interfacing with additional HW
- Hardware accelerators

Goal

Design a processor that...

- has high throughput and single-threaded performance…
- and is very energy-efficient

Overview

- Background, Problem and Goal
- Novelty, Key Approach and Ideas
- Mechanisms (in some detail)
- Key results, Methodology and Evaluation
- Summary
- Strengths and Weaknesses
- Thoughts and Ideas
- Key Takeaways
- Open Discussion

ARM big.LITTLE Architecture

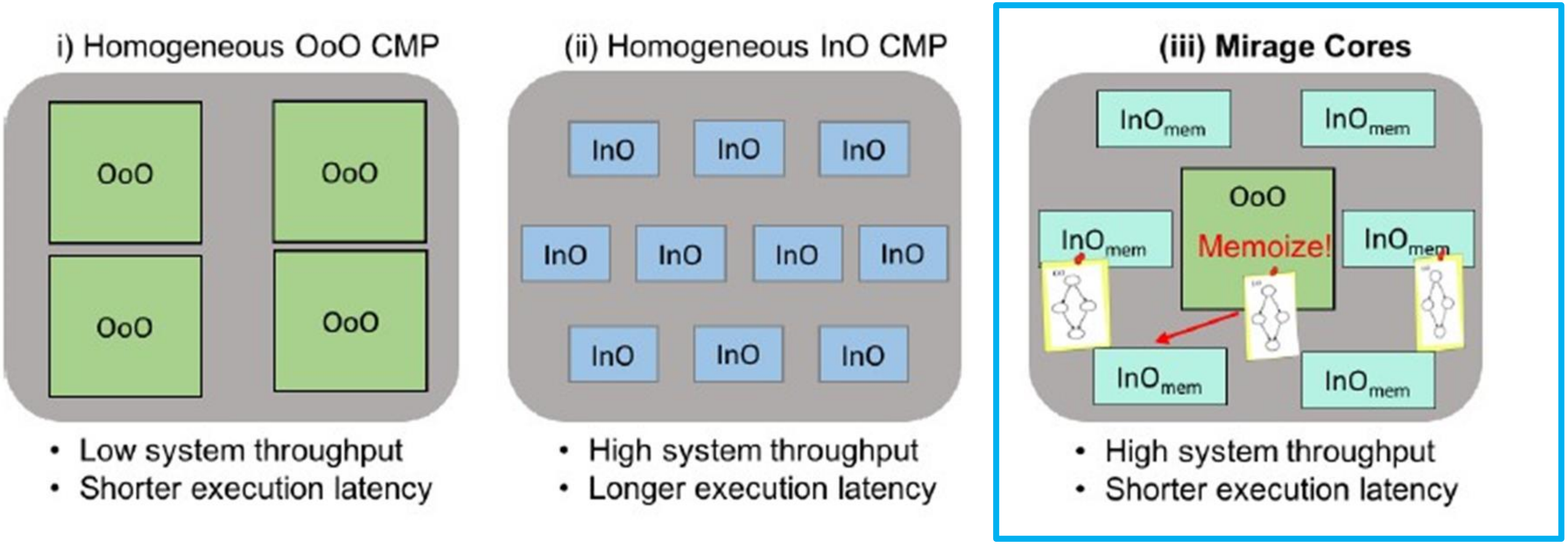
- **Released in 2011**

- Many Android Smartphones

- Apple A series (A14 used in iPhone 12s)

- Nintendo Switch using Nvidia Tegra XI

Mirage Core Architecture


i) Homogeneous OoO CMP
OoO
OoO
OoO
OoO
(ii) Homogeneous InO CMP
InO
InO
InO
InO
InO
InO
InO
InO
InO
InO
(iii) Mirage Cores
InO_{mem}
InO_{mem}
OoO
Memoize!
InO_{mem}
InO_{mem}
InO_{mem}
InO_{mem}

i) Homogeneous OoO CMP

- Low system throughput
- Shorter execution latency

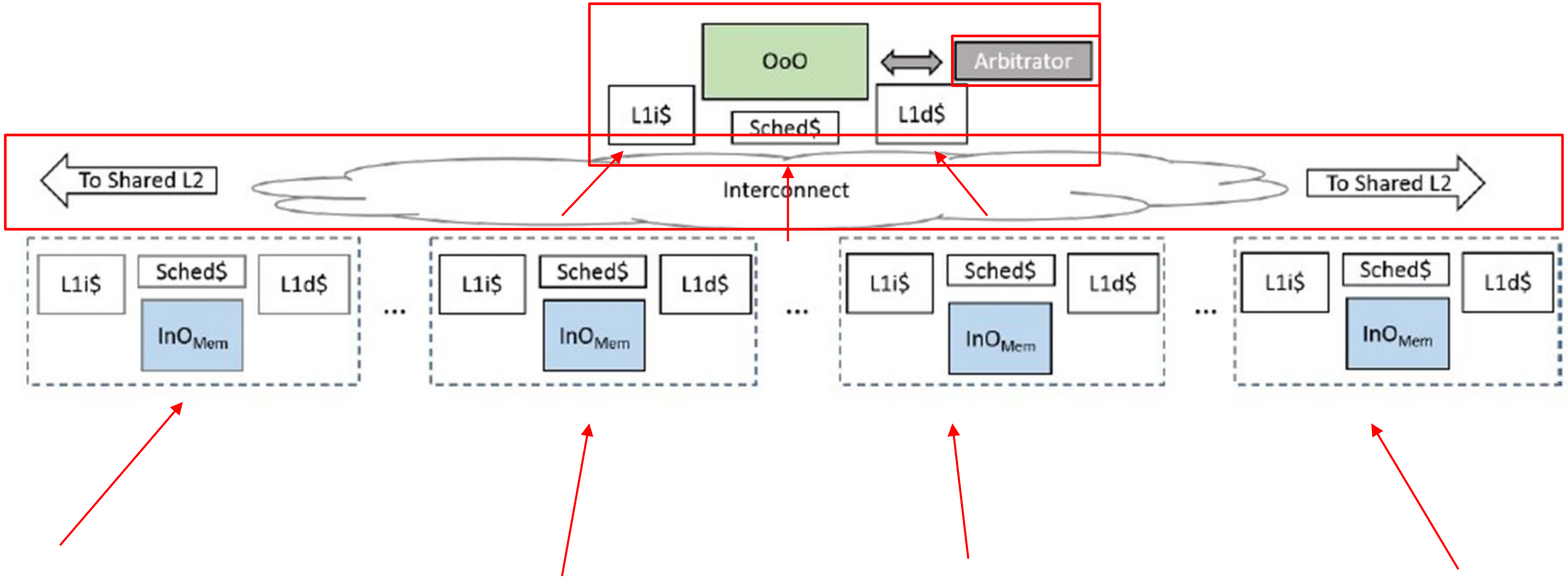
(ii) Homogeneous InO CMP

- High system throughput
- Longer execution latency

(iii) Mirage Cores

- High system throughput
- Shorter execution latency

Mirage Core Architecture

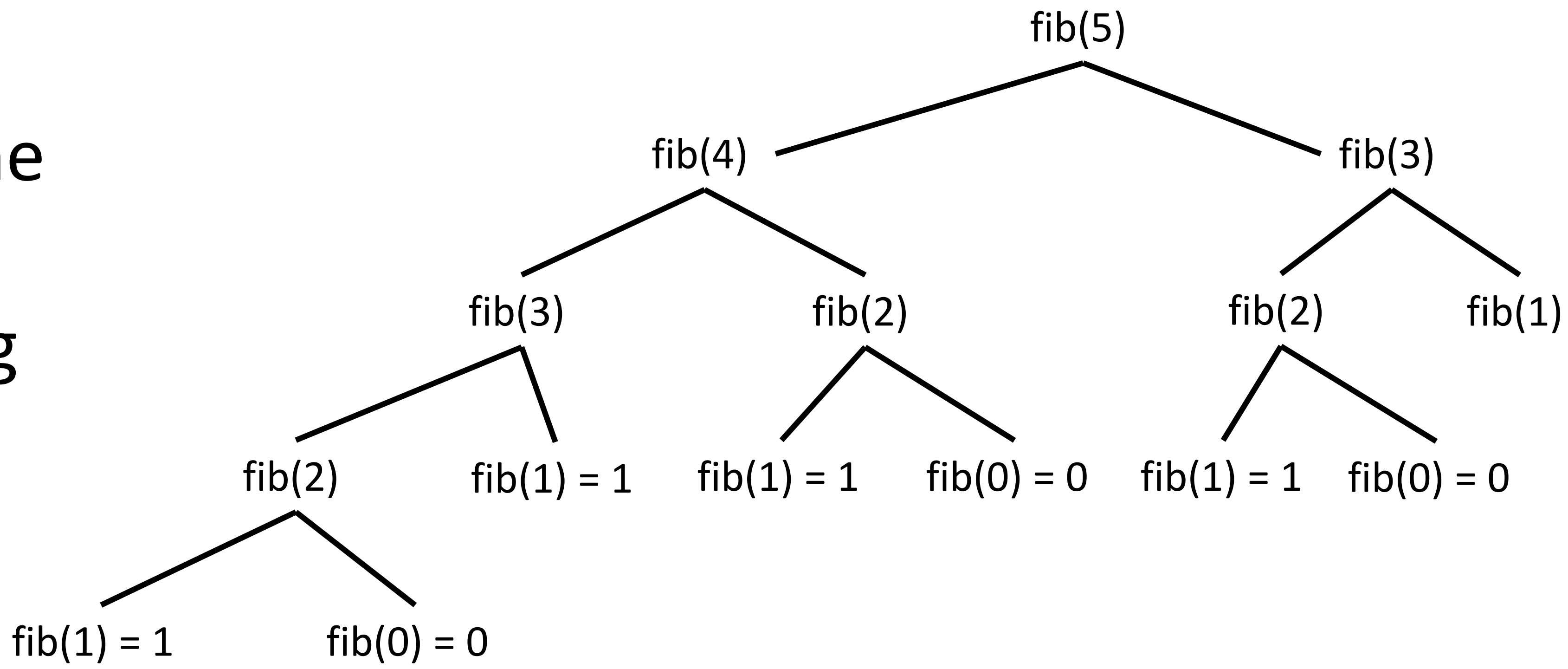


Overview

- Background, Problem and Goal
- Novelty, Key Approach and Ideas
- Mechanisms (in some detail)
- Key results, Methodology and Evaluation
- Summary
- Strengths and Weaknesses
- Thoughts and Ideas
- Key Takeaways
- Open Discussion

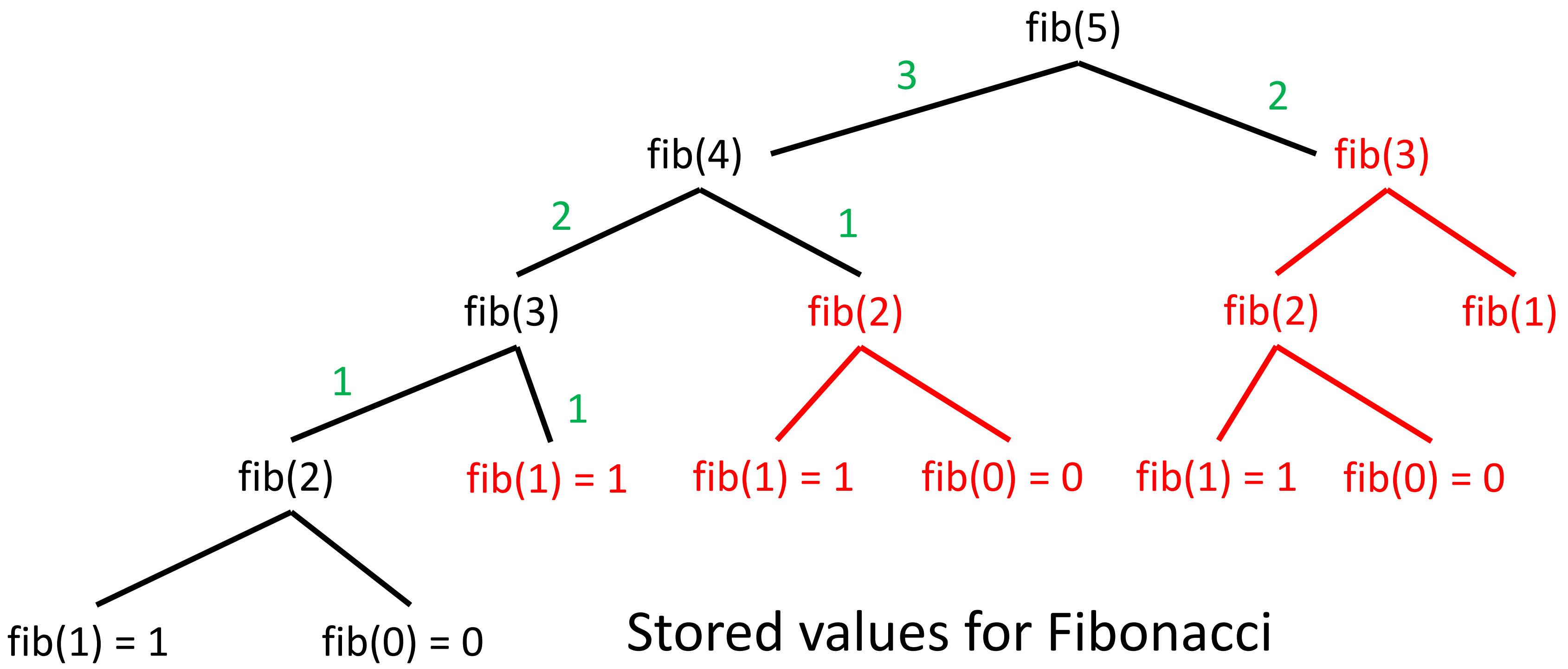
Memoization

- Calculating the 5th Fibonacci Number using recursion



Memoization

- Calculating the 5th Fibonacci Number with Memoization, by storing intermediate values in an array

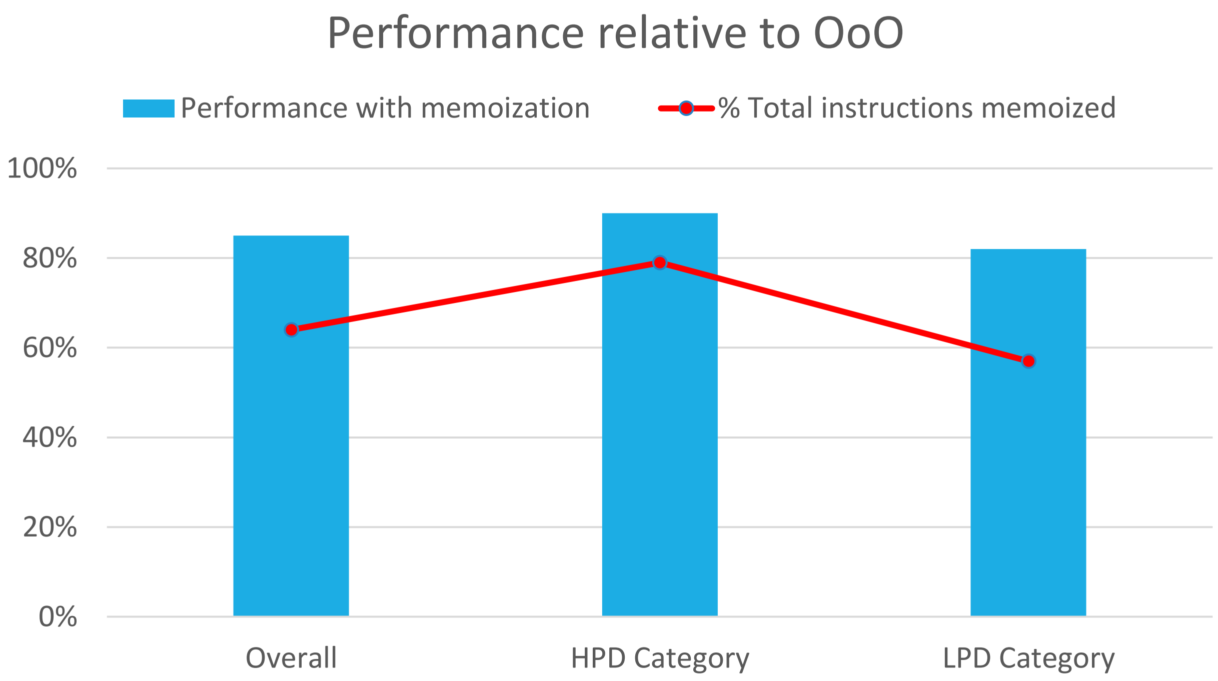


Stored values for Fibonacci

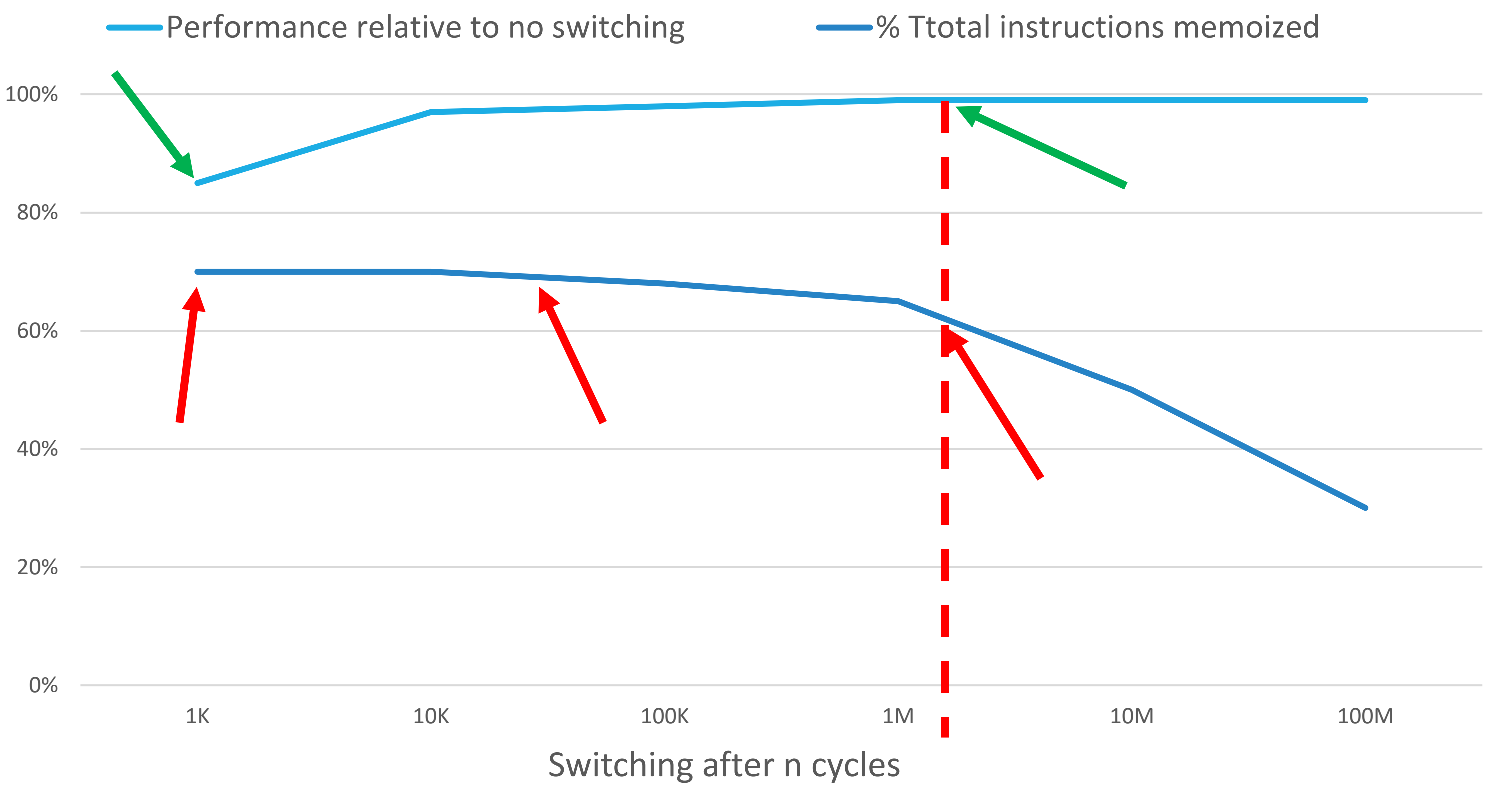
fib(n)	**0**	**1**	**1**	**2**	**3**	**5**
n	0	1	2	3	4	5

Memoization

- Reordering of long latency events only accounts for 19% of the performance advantage of OoO's.
- Most applications spend most of their time in loops
- This means that scheduling usually holds the same pattern in similar contexts



Memoizability



Designing the Arbitrator

- Energy-Efficiency Oriented Arbitration
- System Throughput Oriented Arbitration
- Fairness Oriented Arbitration

Energy-Efficiency Oriented Arbitration

- Schedule Cache Misses per Kilo Instructions (SC-MPKI) quantify the usefulness of memoization
- Picks the application with the highest SC-MPKI above a certain threshold
- If none are above the threshold, OoO is turned off to conserve energy

$$\Delta SC\text{-}MPKI = \frac{SC\text{-}MPKI_{InO} - SC\text{-}MPKI_{OoO}}{SC\text{-}MPKI_{OoO}}$$

Energy-Efficiency Oriented Arbitration

- Application 1
  - Has high SC-MPKI$_{InO}$
  - Has low SC-MPKI$_{OoO}$
  - InO-OoO is high
  - -> good candidate for memoization, as it performs well on OoO, but bad on InO

- Application 2
  - Has low SC-MPKI$_{InO}$
  - Has low SC-MPKI$_{OoO}$
  - InO-OoO is near 0
  - -> bad candidate for memoization, as it already performs near OoO

- Application 3
  - Has high SC-MPKI$_{InO}$
  - Has high SC-MPKI$_{OoO}$
  - InO-OoO is near 0
  - -> bad candidate for memoization, because the code probably has unpredictable control flow

System Throughput Oriented Arbitration

- Overall system throughput (STP) as metric for the scheduler
- Migrates the slowest application to the OoO
- Traditional design on heterogeneous chips

$$speedup_i = (\frac{IPC_{InO(i)}}{IPC_{OoO(i)}})$$

Fairness Oriented Arbitration

- Arbitrator migrates application in round robin order
- Util(i) metric to determine each application’s timeshare
- Application will be migrated only if either Util(i) is less than 1/(#apps) or if ΔSC-MPKI falls below the threshold

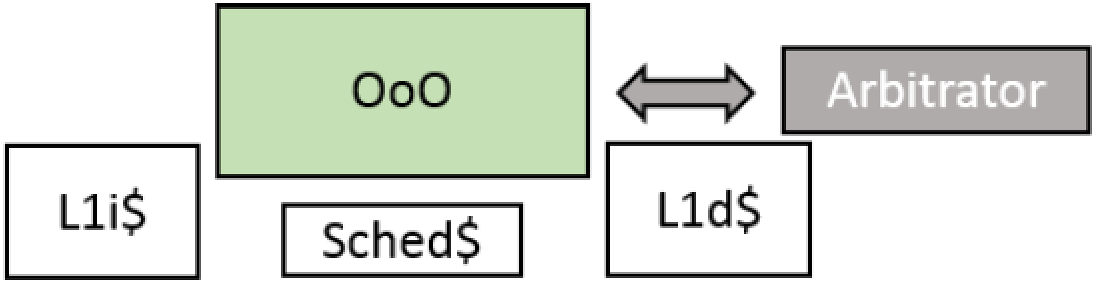
$$Util_{(i)} = \left(\frac{t_{OoO(i)} + t_{InOmemoize(i)} * speedup_i}{t_{overall}}\right)$$

Designing the Core Architecture

- Designing the OoO core
- Designing the InO core
- Migration between the cores

Designing the OoO Core

- In order to memoize schedules, the OoO must be able to recognize
  - (a) when a trace is repetitive
  - (b) if its instructions are scheduled in the same order
- Traces that are deemed memoizable are stored in the schedule cache
- Metrics used to compare two traces are execution time, IPC, memory characteristics, branch misses and reordered instructions



DynaMOS: dynamic schedule migration for heterogeneous cores

Shruti Padmanabha, Andrew Lukefahr, Reetuparna Das, and Scott Mahlke

Advanced Computer Architecture Laboratory

University of Michigan, Ann Arbor, MI

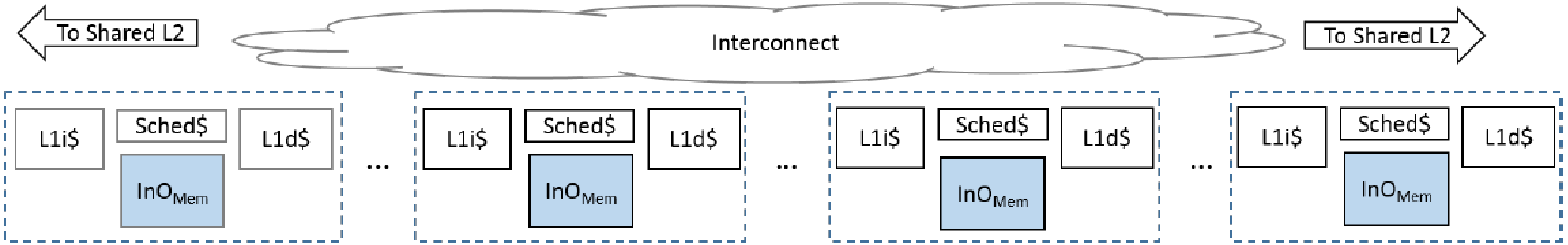
Micro-48: Proceedings of the 48th International Symposium on Microarchitecture, December 2015

https://dl.acm.org/doi/pdf/10.1145/2830772.2830791

Designing the InO Core

Introduces the OinO mode with following modifications

- Atomic Execution
- Physical Register File
- Load/Store Queue
- Schedule Cache

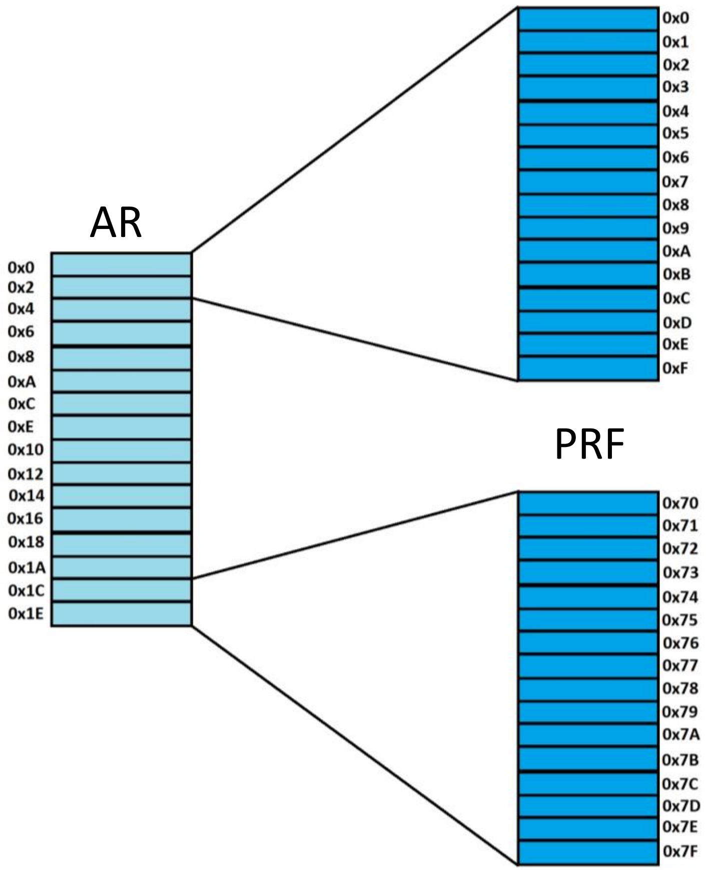


Atomic Execution

- InO cores cannot detect unexpected events like branch mispredictions or memory aliases
- Forces the OinO to execute schedules atomically
- On misprediction, resets the whole execution and executes in original, non-memoized program order

Physical Register File

- OinO is supplemented with expanded register file that maps every architectural register to at most 4 physical registers (PR), resulting in a 128 entry PRF
- Bookkeeping adds an additional 28 bytes of storage
- A bigger PRF and tables adds 14% dynamic energy to the InO



Load-Store Queue

- Implemented to circumvent memory alias errors for load and store operations
- Is added to every recorded schedule as a fixed-size meta-data block and adds 20B
- 32 entry LSQ contributes 5.5% overhead to the dynamic energy of OinO

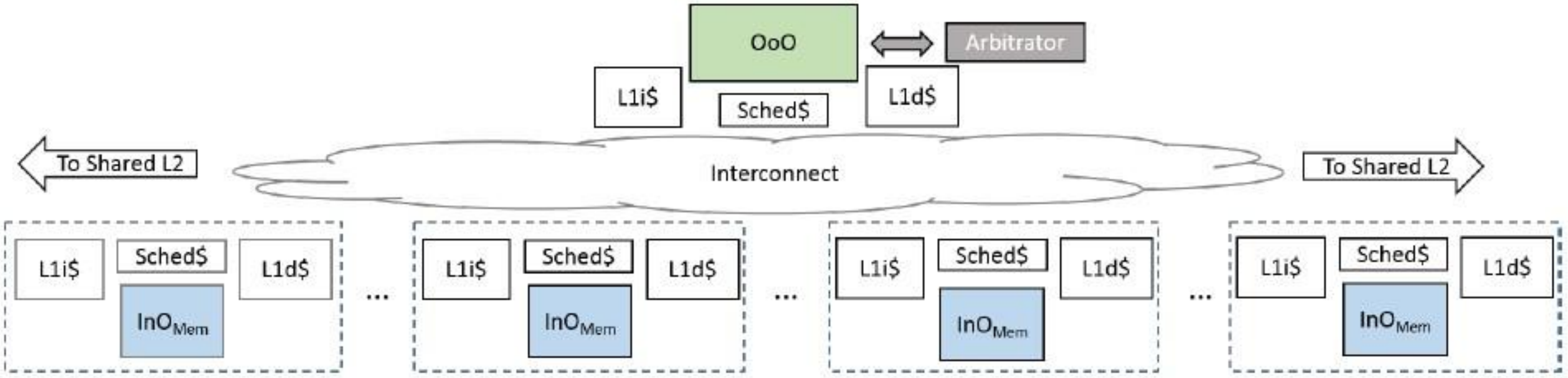
Load/Store Queue		
Load/Store	Addr	Value
L	0x1234	1790
S	0x2468	-532
S	0x3579	1234
L	0x6729	82394
L	0x8923	-3659
S	0x1234	58329
L	0x3333	-2342
L	0x4444	93094

Schedule Cache

- 8KB cache that stores schedules memoized and transferred from the OoO
- Trace mis-speculations and SC writes are very expensive
- Employ an algorithm that is heavily biased against traces that mis-speculate
- Eviction policy: unmemoizable traces -> least recently used
- Contributes 10% towards leakage energy but reduces L1 iCache access energy

Migration between cores

- Must store all of the active core’s state, including the RF, PC, control bits, store buffer entries, etc. into memory on migration and its pipeline must be flushed



Overview

- Background, Problem and Goal
- Novelty, Key Approach and Ideas
- Mechanisms (in some detail)
- Key results, Methodology and Evaluation
- Summary
- Strengths and Weaknesses
- Thoughts and Ideas
- Key Takeaways
- Open Discussion

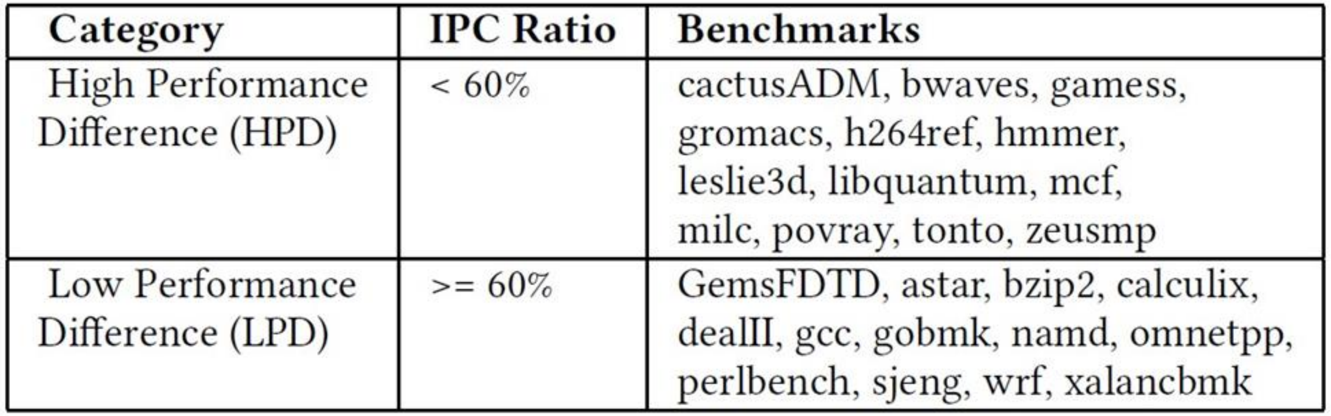
Methodology

- OoO:
  - 3 wide superscalar @ 2 GHz
  - 12 stage pipeline
  - 128 entry ROB
  - 128 entry integer register file
  - 256 entry floating-point register file
  - 8KB Schedule Cache

- InO:
  - 3 wide superscalar @ 2 GHz
  - 8 stage pipeline
  - 128 entry integer register file
  - 128 entry floating-point register file
  - 8KB Schedule Cache

- Memory System:
  - 32 KB L1 iCache @ 2 cycles
  - 32 KB L1 dCache @ 2 cycles
  - 2 MB shared L2 Cache with stride prefetcher @ 15 cycles
  - 8192 MB Main Memory @ 120 cycles
  - 32 B L1-L2 bus @ 2 GHz

Methodology

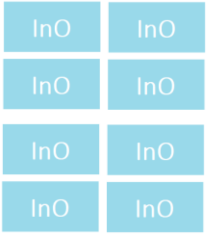


Category	IPC Ratio	Benchmarks
High Performance Difference (HPD)	< 60%	cactusADM, bwaves, gamess, gromacs, h264ref, hmmer, leslie3d, libquantum, mcf, milc, povray, tonto, zeusmp
Low Performance Difference (LPD)	>= 60%	GemsFDTD, astar, bzip2, calculix, dealII, gcc, gobmk, namd, omnetpp, perlbench, sjeng, wrf, xalancbmk

- 27 applicatitons from SPEC 2006 benchmark suite
- Gem5 simulator to model Mirage Cores
- McPAT modeling framework to estimate area, static and dynamic energy consumption for the core and L1 caches

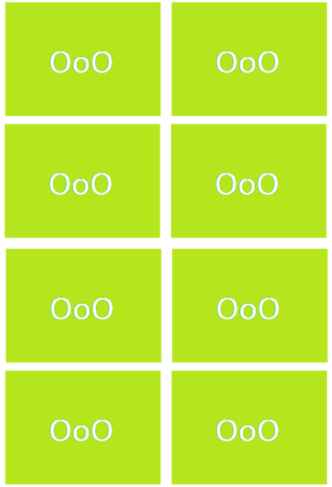
Evaluation

8:0 Homo-InO



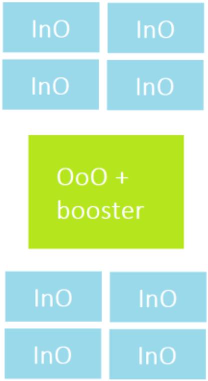
InO InO
InO InO
InO InO
InO InO

0:8 Homo-OoO



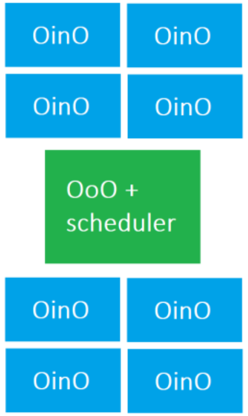
OoO OoO
OoO OoO
OoO OoO
OoO OoO

8:1 Het-Traditional



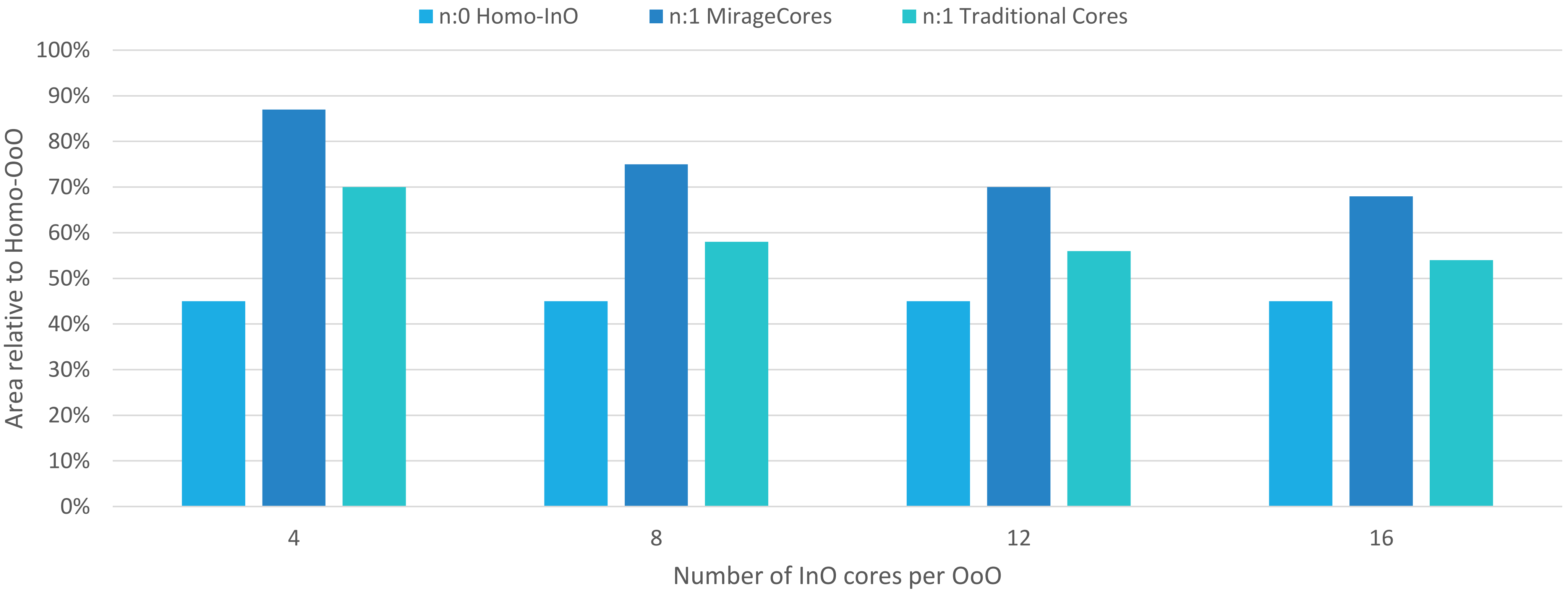
InO InO
InO InO
OoO + booster
InO InO
InO InO

8:1 Mirage

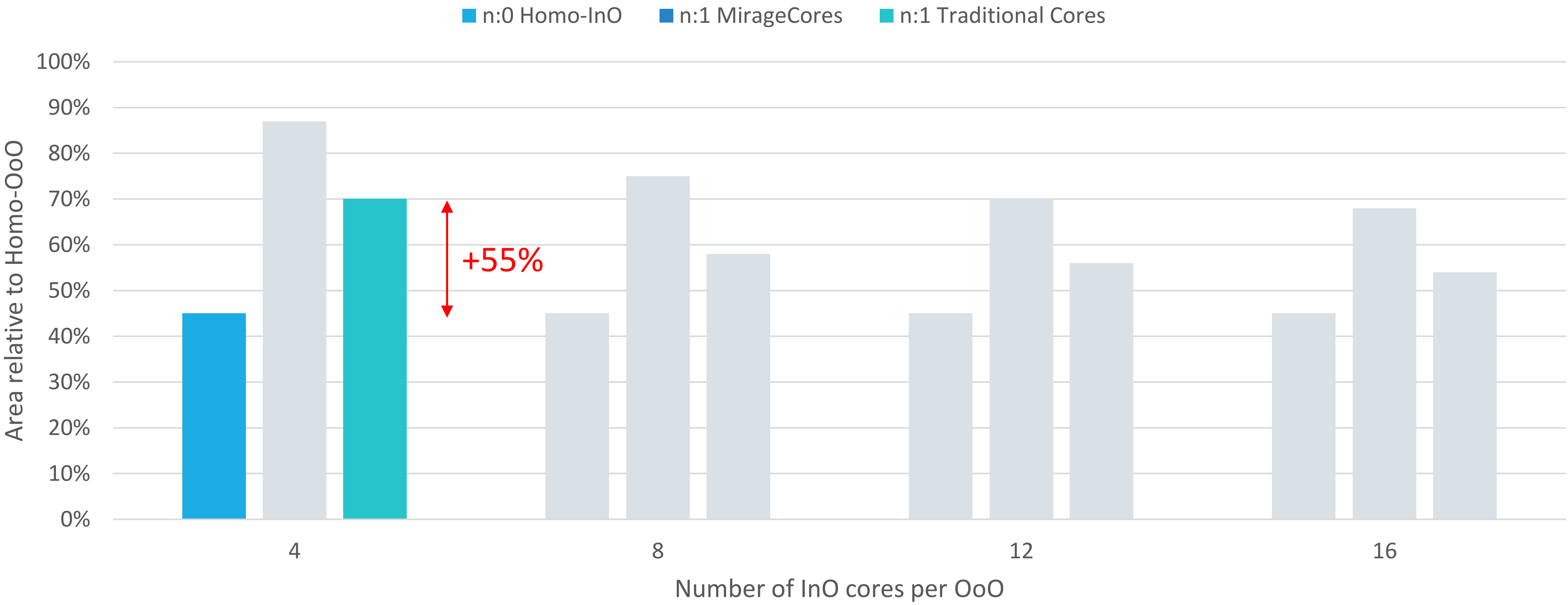


OinO OinO
OinO OinO
OoO + scheduler
OinO OinO
OinO OinO

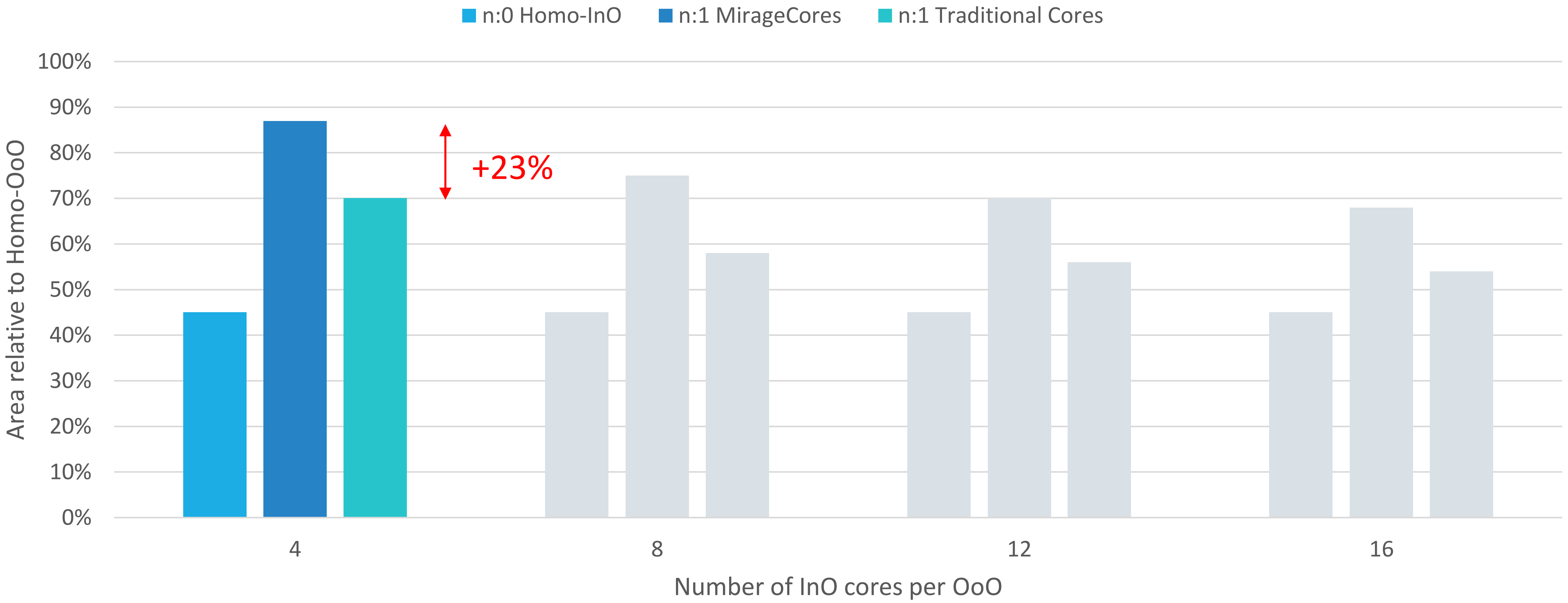
Architecture Configuration



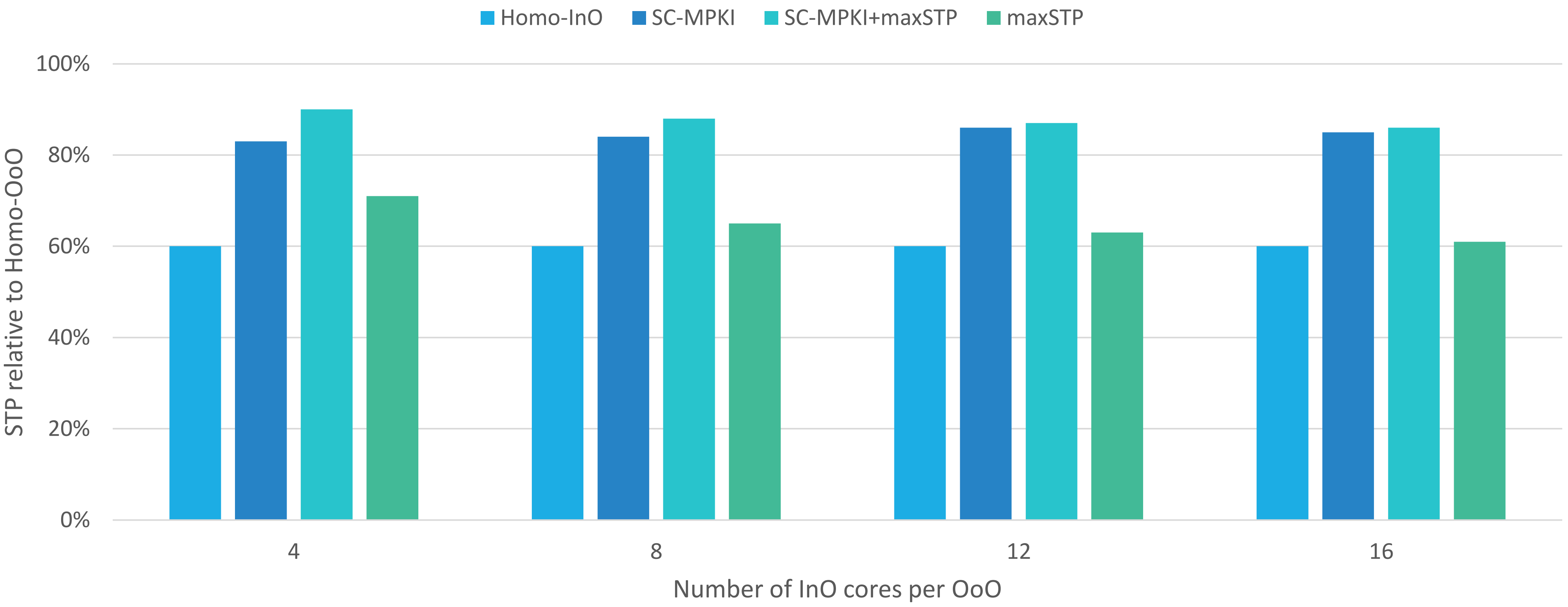
Architecture Configuration



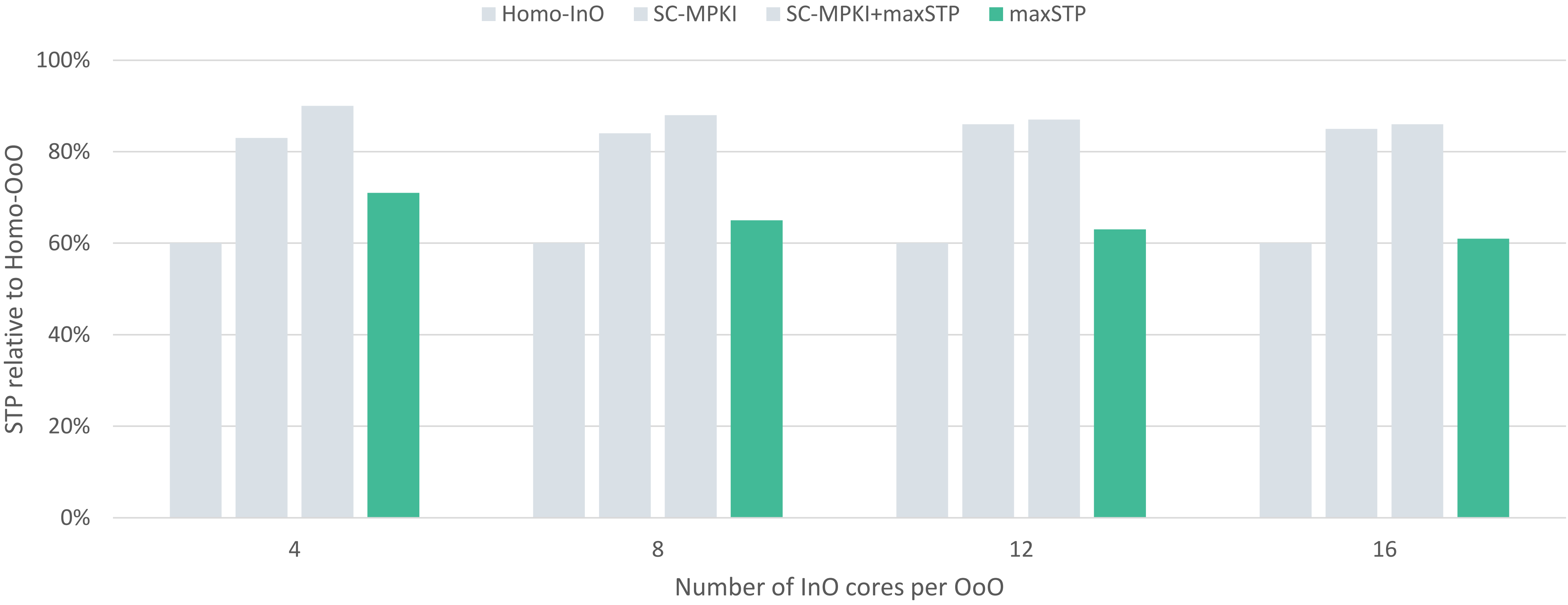
Architecture Configuration



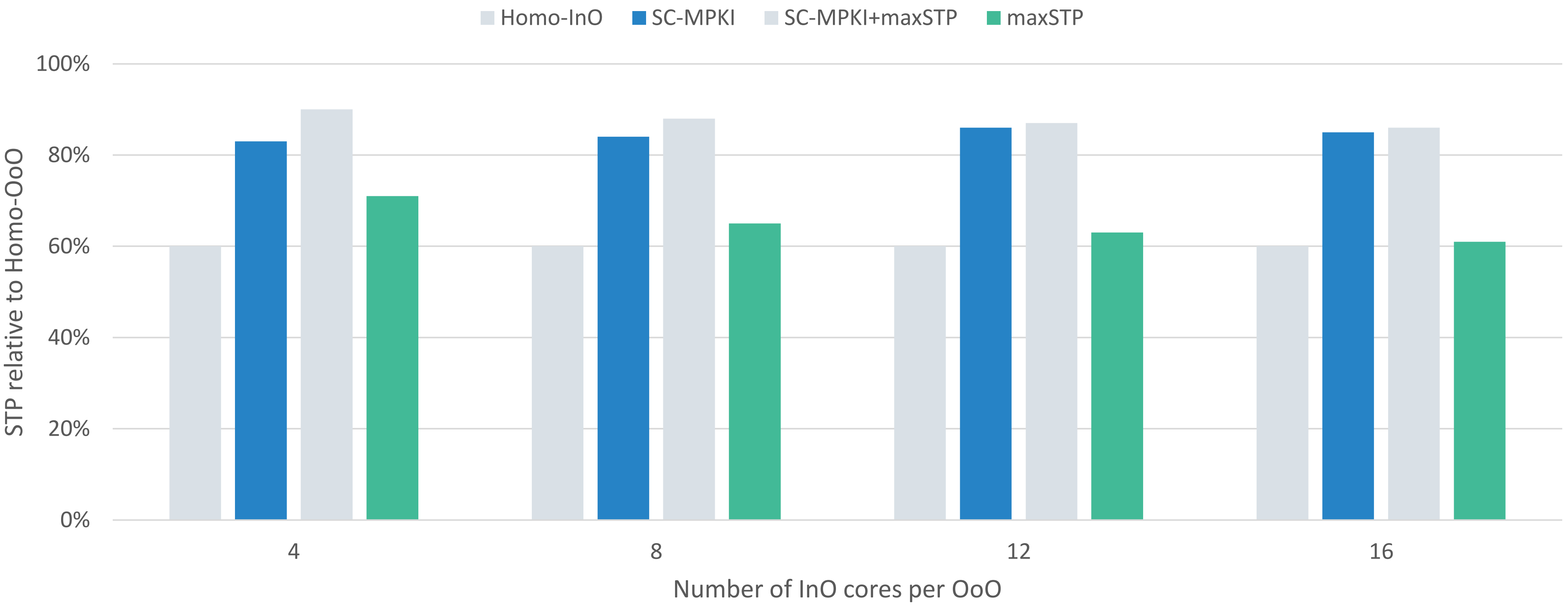
Performance



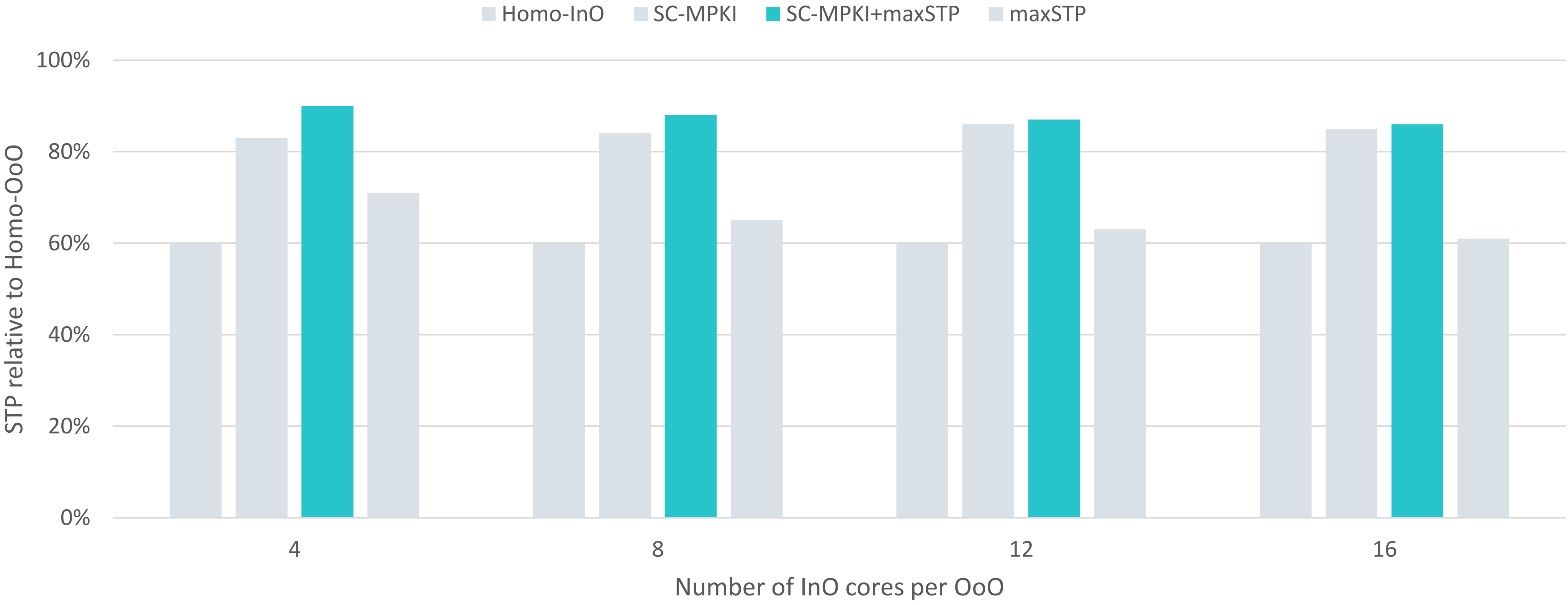
Performance



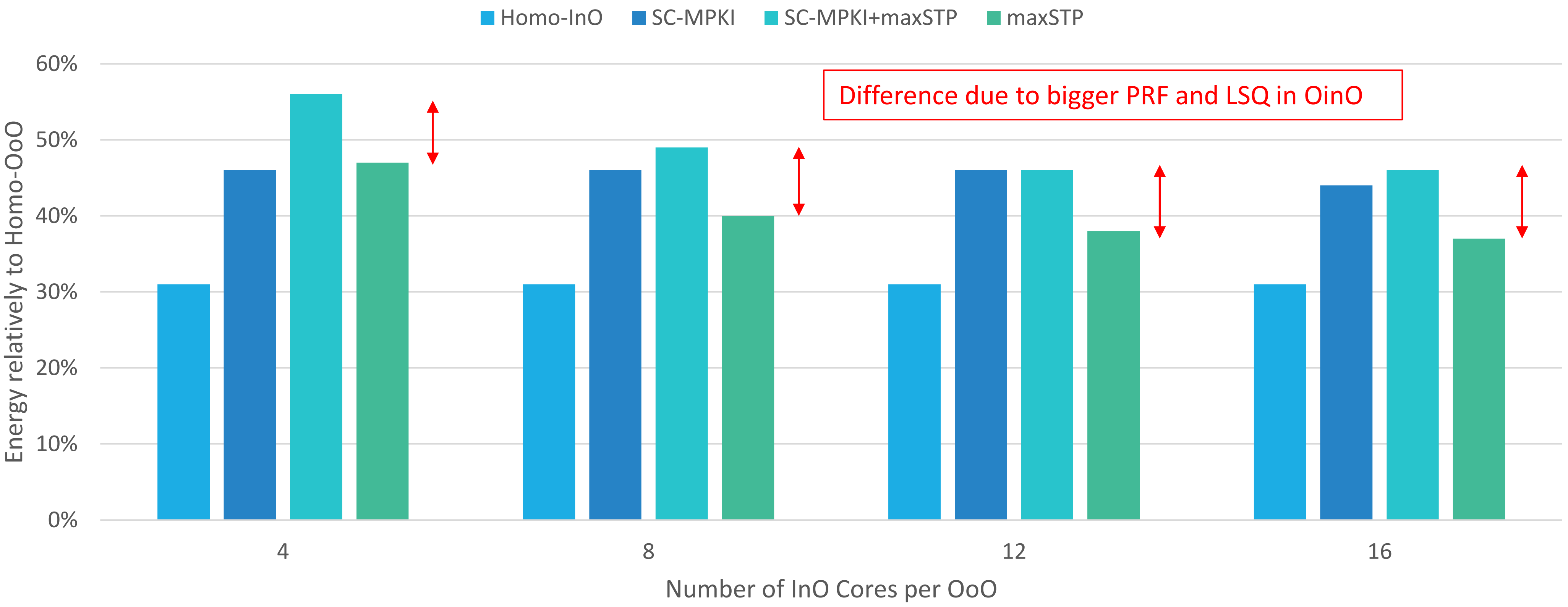
Performance



Performance

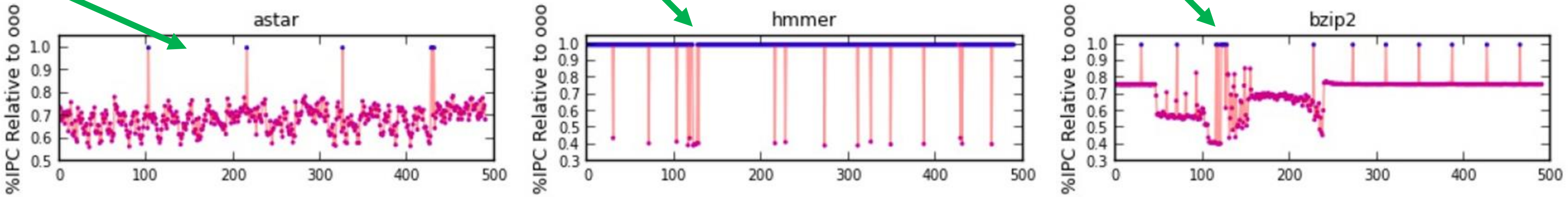


Energy Consumption

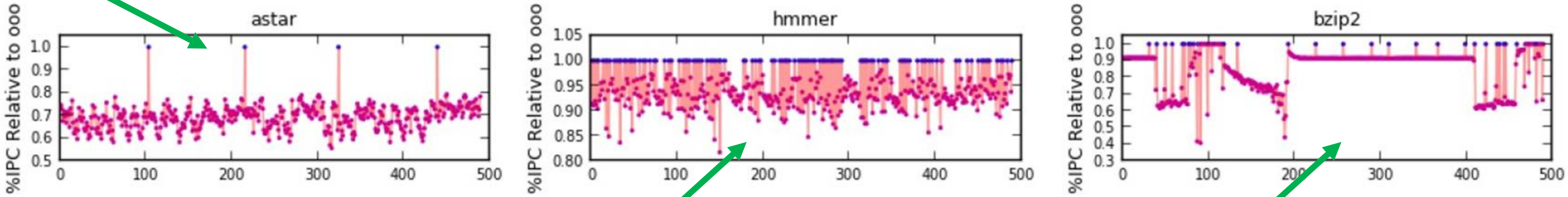


Difference due to bigger PRF and LSQ in OinO

Case Study



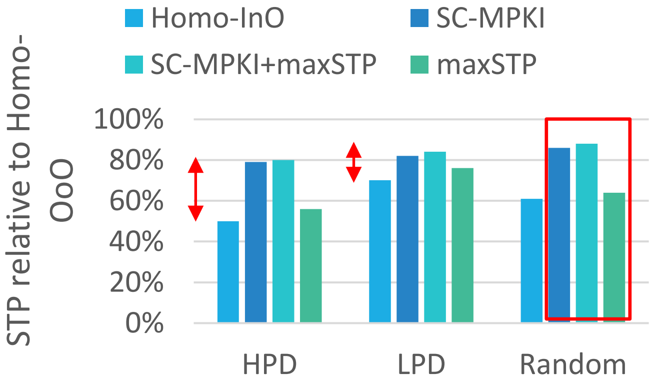
maxSTP

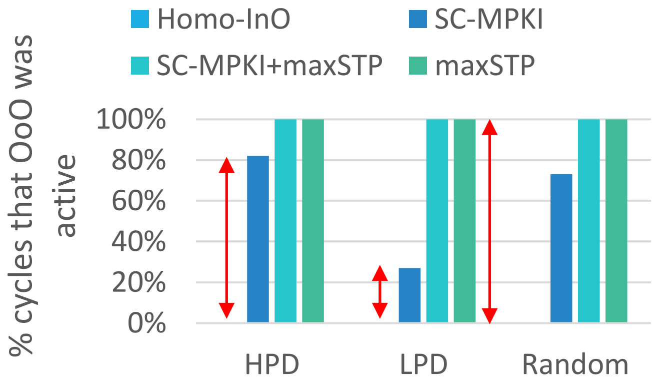


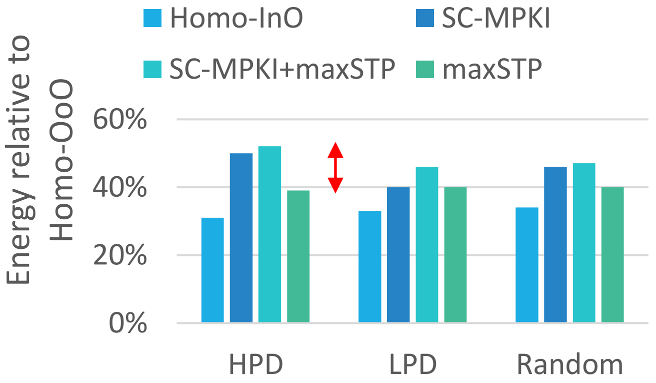
SC-MPKI

Analyses of Benchmark Categories

- 8:1 configuration

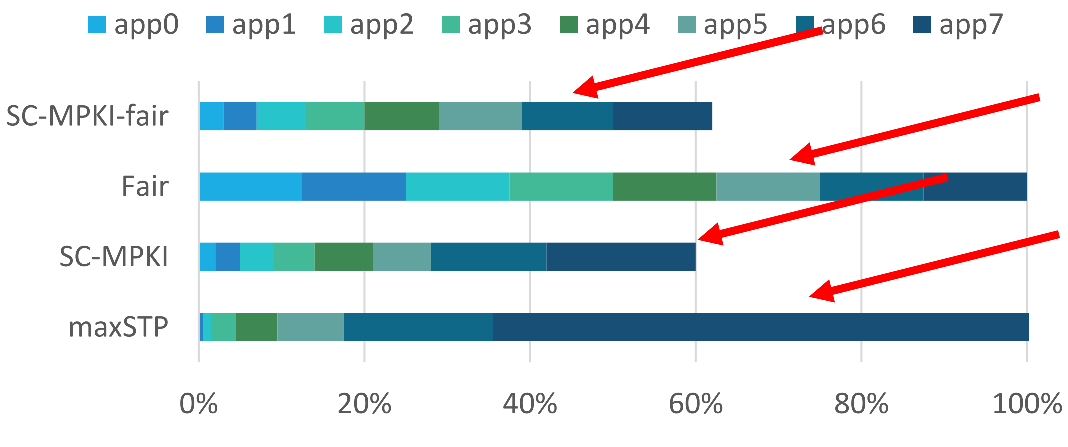




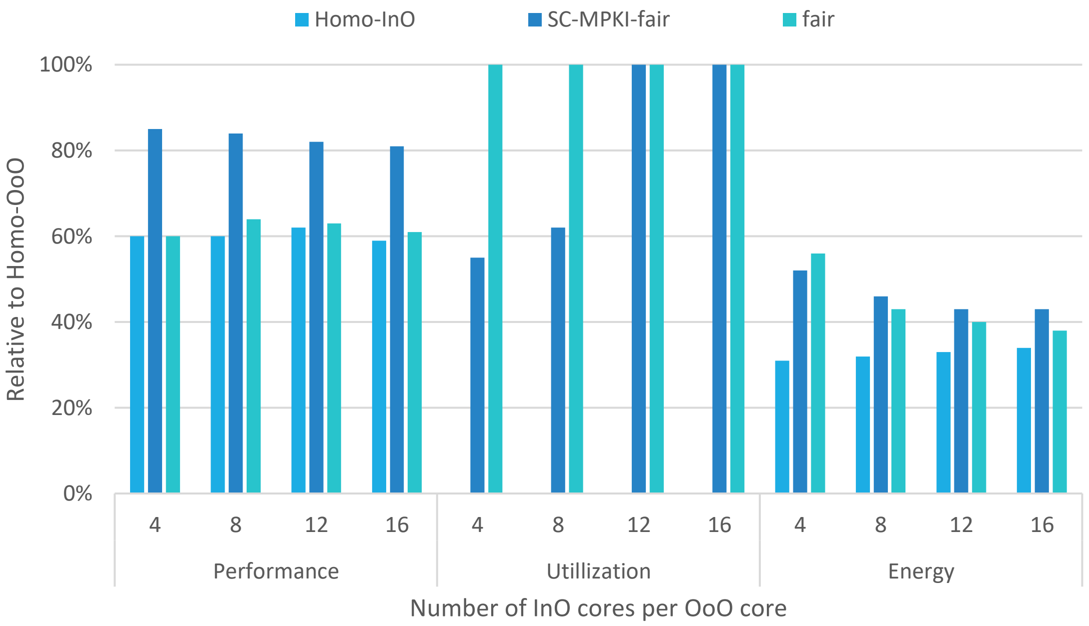


Arbitrator for Equal Resource Sharing

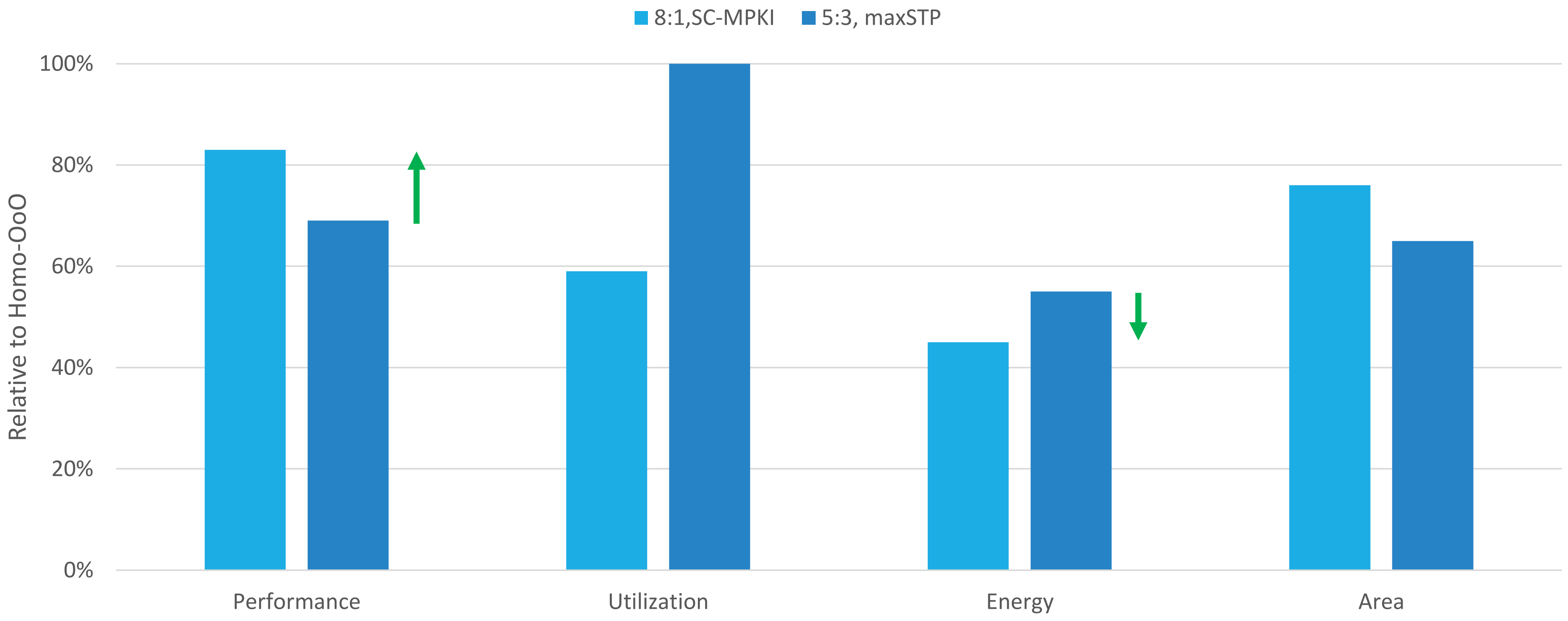
- 8:1 configuration



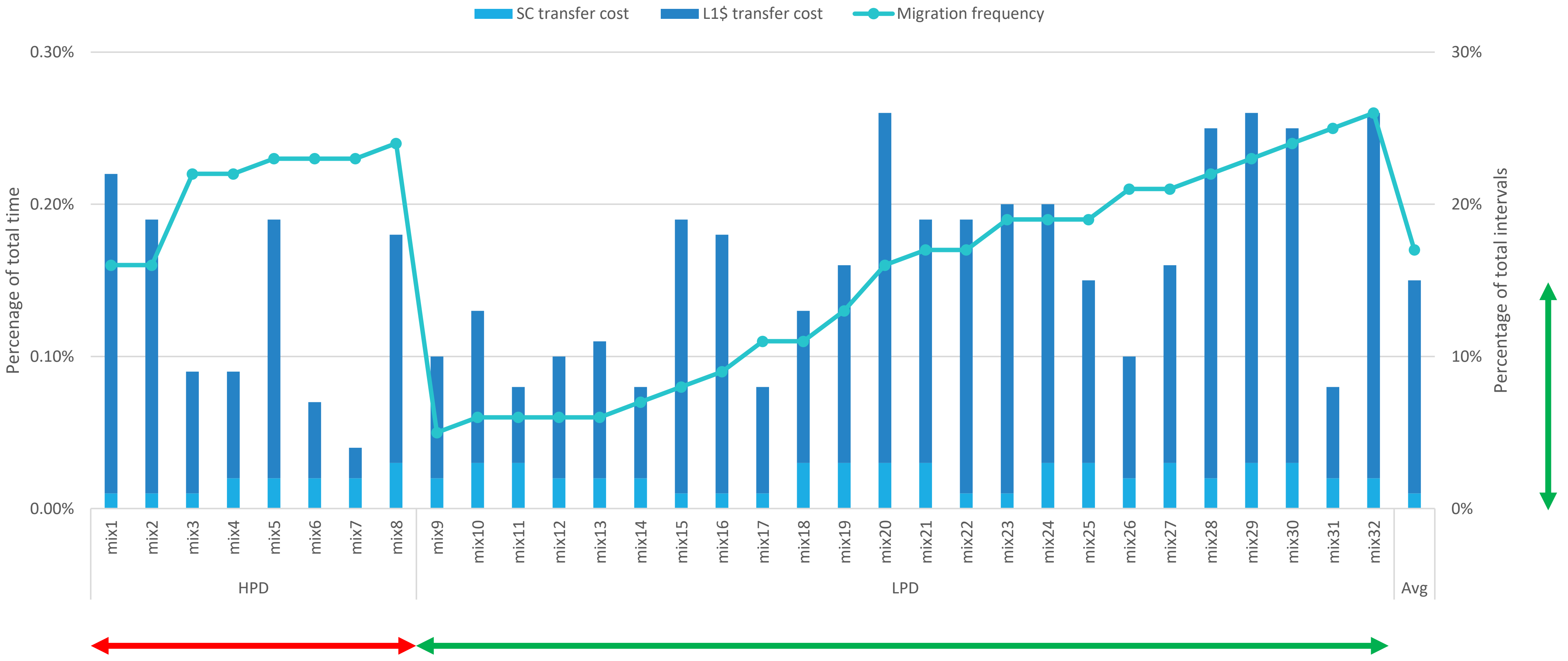
Utilization of OoO per benchmark in a workload mix for the 8:1 configuration



Area Neutral Study



Cost of Core Migration



Overview

- Background, Problem and Goal
- Novelty, Key Approach and Ideas
- Mechanisms (in some detail)
- Key results, Methodology and Evaluation
- Summary
- Strengths and Weaknesses
- Thoughts and Ideas
- Key Takeaways
- Open Discussion

Summary

- Problem:
  - Practical power and thermal constraints limit the deployment of homogeneous multicore systems with many big OoO cores
  - Low performance of InO cores limits their widespread usage
- Goal:
  - The goal is to design a Het-CMP with near OoO performance and InO energy consumption
- Idea:
  - The idea is to use clusters of InO cores around one OoO core
  - The OoO core is used as a «scheduler» and the InO cores as «workers»
- Evaluation:
  - The Mirage Core can achieve on average 84% performance of a Homo-CMP, while conserving 55% of energy and 25% of area costs

Overview

- Background, Problem and Goal
- Novelty, Key Approach and Ideas
- Mechanisms (in some detail)
- Key results, Methodology and Evaluation
- Summary
- Strengths and Weaknesses
- Thoughts and Ideas
- Key Takeaways
- Open Discussion

Strengths

- Simple Idea, that can achieve high system throughput and low energy consumption without having to make a heavy tradeoff on single thread performance.
- Scheduler is flexible to fulfil the users needs, hence applicable to many systems.
- Tackles an important problem in energy consumption
- Well-written, easy to understand paper

Weaknesses

- Does not go too much into detail when it comes to multithreaded computing
- Gives no programming model or example design
- Only looks at CPU heterogeneity
- Servers cannot profit off this architecture due to more irregular fetch patterns
- Is only efficient when there is a good mix between LPD and HPD workloads

Overview

- Background, Problem and Goal
- Novelty, Key Approach and Ideas
- Mechanisms (in some detail)
- Key results, Methodology and Evaluation
- Summary
- Strengths and Weaknesses
- Thoughts and Ideas
- Key Takeaways
- Open Discussion

Intel Core Alder Lake (2021)

- 8 «little» Gracemont cores for high efficiency
- 8 «big» Golden Cove cores for high performance with multithreading
- 24 threads in total
- including a HW scheduler
- To be released in 2021

Overview

- Background, Problem and Goal
- Novelty, Key Approach and Ideas
- Mechanisms (in some detail)
- Key results, Methodology and Evaluation
- Summary
- Strengths and Weaknesses
- Thoughts and Ideas
- Key Takeaways
- Open Discussion

Key Takeaways

- A nice approach to get high system throughput, high single-thread performance and low energy consumption at the same time.
- Does not require a lot of new additional HW
- Flexible Arbitrator Design
- There is a lot to build on with this idea
- Heterogeneous Designs are an important tool for increased energy efficiency

Overview

- Background, Problem and Goal
- Novelty, Key Approach and Ideas
- Mechanisms (in some detail)
- Key results, Methodology and Evaluation
- Summary
- Strengths and Weaknesses
- Thoughts and Ideas
- Key Takeaways
- Open Discussion

Open Discussion

- Fields where the Mirage Core can be applied
- What needs to be changed to make it efficient for servers?
- What needs to be changed to make it efficient for multithreading?
- Can the Mirage Cores problems be fixed by adding more heterogeneity in general?
- Hardware accelerators that can be used