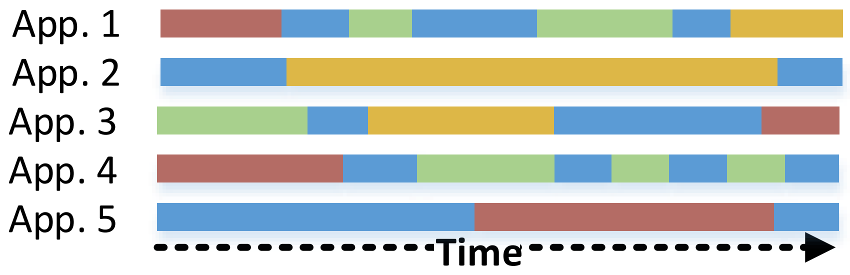
Modularizing the Microprocessor Core to Outperform Traditional Out-of-Order

Tony Nowatzki   Karthikeyan Sankaralingam

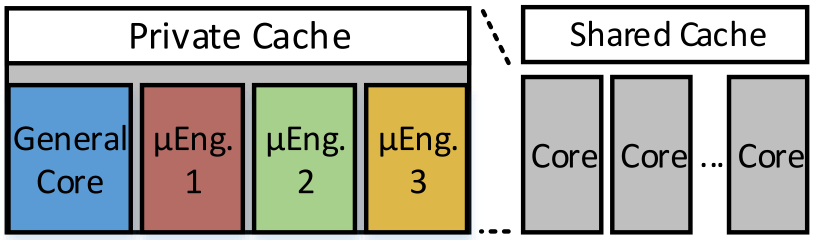
[Paper Link: Analyzing Behavior Specialized Acceleration. ASPLOS 2016]

Observation: Programs Execute in Phases



- Programs phases have distinct properties (degree of memory regularity, control criticality, data parallelism etc.)
- These properties can be exploited by highly-tailored non-general purpose architectures

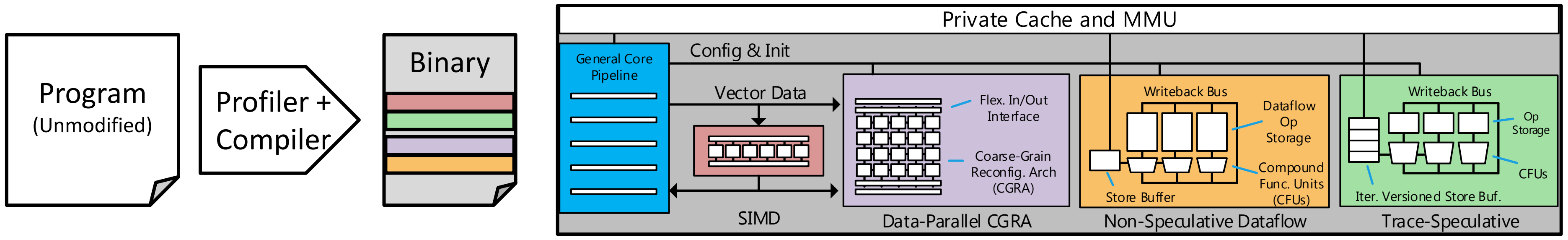
ExoCore: Modular, Heterogeneous Core



- ExoCore integrates a general purpose core with programmable µEngines, behind a single cache system
- It dynamically switches between µEngines to match the properties of the current program phase, improving performance and energy-efficiency

2-wide ExoCore provides 60% speedup over 2-wide OOO
2-wide ExoCore is 2x lower power than 6-wide OOO with same performance

ExoCore Approach Overview

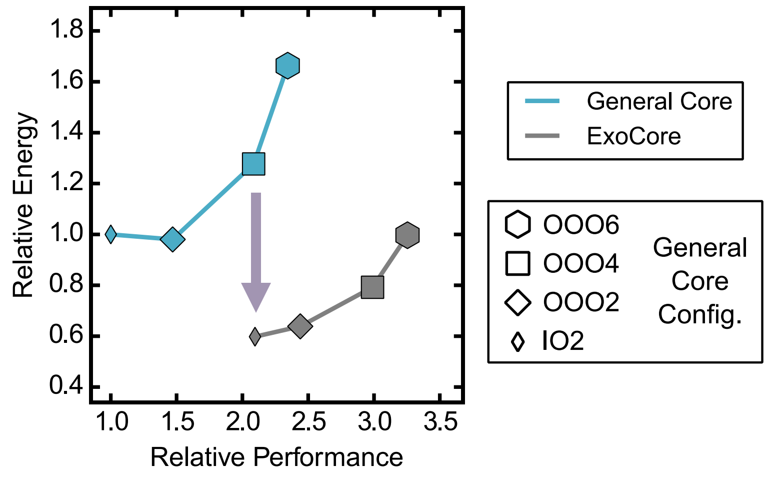


ExoCore

- ExoCore Compilation
  - Targets unmodified programs and uses profile information to assign each region to a µEngine
  - µEngine configurations and decisions stored in “fat binary”
- Execution Model
  - General core hands-off execution to µEngines on region entry (loops/funcs)
  - µEngines (and general core) go into low-power state when not-in use
- Example µEngines
  - Data parallel CGRA: targets vectorizable regions with high degree of computation
  - Non Speculative Dataflow: targets non-data parallel codes with non-critical control flow
  - Trace-Processor: non-data parallel loops with highly biased branches

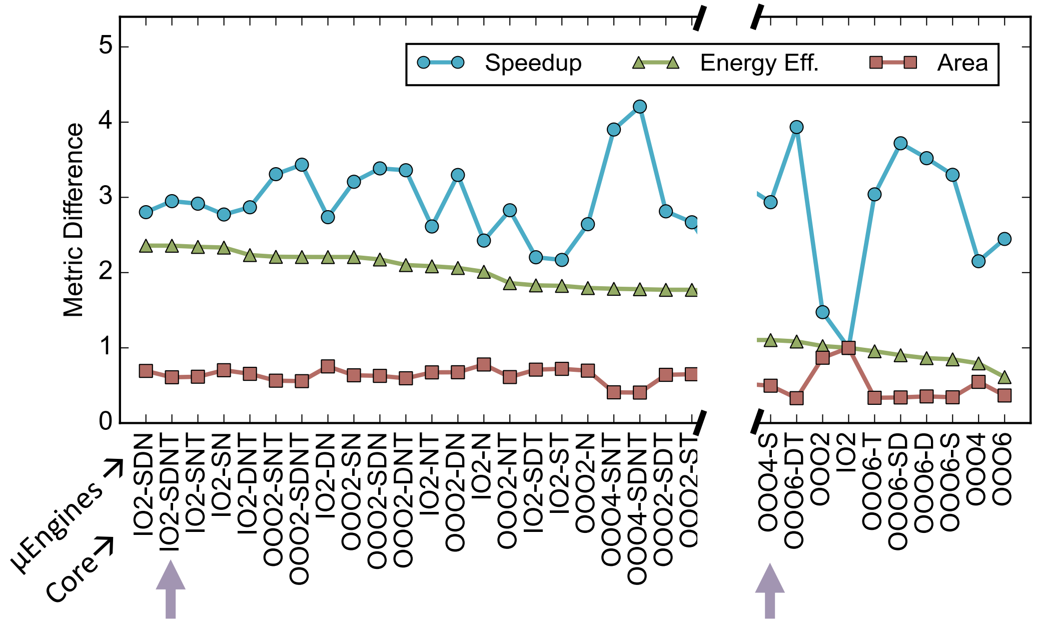
Results Summary

Performance/Energy Tradeoffs



Core Type	**ExoCore Speedup**	**ExoCore Energy Reduction**
Small (in-order)	2x	1.7x
Med. (2-Wide OOO)	1.6x	1.5x
Large (6-Wide OOO)	1.4x	1.7x

Design Space (4 Cores × 16 Accel. Combos)



- Modular design enables simple trade-off of performance, area, energy, and design complexity.
- In-order core with four μEngines matches the perf. of a 4-wide OOO core, with 15% less area and 2× energy efficiency.

Implications and Ongoing Work

- Implications
  - Modularity simplifies general purpose core design – µEngines can be designed and integrated without disruptive change to the core.
  - Promising approach for exceeding the performance/energy frontier of conventional Out-of-Order processors, and for continual improvements.
- Ongoing Work – RISCV-based Prototype RISC-V
  - Goal 1: Fully open-source configurable ExoCore prototype in Chisel, integrated with Berkley Rocket Core, with working compiler.
  - Goal 2: Complete FPGA prototype in 6-months
  - Goal 3: Tape-out at 28nm in 1-year