Design Trade-offs for Emerging HPC Processors Based on Mobile Market Technology

Adrià Armejach[1,2]
Marc Casas[1]
Miquel Moretó[1,2]

Abstract High-Performance Computing (HPC) is at the crossroads of a potential transition towards mobile market processor technology. Unlike in prior transitions, numerous hardware vendors and integrators will have access to state-of-the-art processor designs due to *Arm*'s licensing business model. This fact gives them greater flexibility to implement custom HPC-specific designs.

In this paper, we undertake a study to quantify the different energy-performance trade-offs when architecting a processor based on mobile market technology. Through detailed simulations over a representative set of benchmarks, our results show that: (i) a modest amount of last-level cache per core is sufficient, leading to significant power and area savings; (ii) in-order cores offer favorable trade-offs when compared to out-of-order cores for a wide range of benchmarks; and (iii) heterogeneous configurations help to improve processor performance and energy-efficiency.

Keywords multicore design trade-offs · energy efficiency · heterogeneous processors

1 Introduction

Today's High-Performance Computing (HPC) deployments are dominated by commodity server processors. These processors feature a few aggressive out-

Adrià Armejach
adria.armejach@bsc.es

Marc Casas
marc.casas@bsc.es

Miquel Moretó
miquel.moreto@bsc.es

[1] Barcelona Supercomputing Center, Barcelona, Spain
[2] Universitat Politècnica de Catalunya, Barcelona, Spain

of-order cores clocked at high frequencies with large shared on-chip caches. However, the same trends that forced the transition from special-purpose vector architectures to RISC processors used in workstations in the mid 1990s, and later to CISC architectures used in contemporary commodity PCs are starting to appear again in the low-power mobile market segment. The higher volumes at which the mobile market operates enables faster product cycles, leading to a fast-paced ecosystem that pushes performance and technological improvements. This paradigm change can enable the use of low-power mobile market technologies in HPC in the near future [1, 2].

In fact, these trends are already materializing and vendors have started to roll out competitive processors that target the HPC market based on low-power ARM-based systems-on-chip (SoC), e.g., the ThunderX processors family [3]. In addition, several HPC companies are already deploying large ARM-based machines, e.g., Cray with the deployment of a 10,000+ ARMv8-A core system [4, 5], and in the near future Fujitsu's next flag-ship machine - The Post-K [6].

Such a change towards processors that leverage technology from a fast-paced market, with potentially multiple competing vendors and integrators involved in architecting and assembling these chips, can lead to a higher degree of design freedom. Opening the door to tailored designs that fit HPC workload needs, in order to increase computational throughput while optimizing energy efficiency.

In this paper, we explore the potential advantages in terms of energy and performance when designing HPC processors based on emerging mobile market technology. We focus on three fundamental design choices that we believe can play a major role in the performance and energy efficiency of a processor. First, careful sizing of the last-level cache (LLC) is paramount to design a processor that exhibits good energy efficiency. The LLC is a major contributor in terms of area and power. Therefore, an over-provisioned LLC, as current designs implement, wastes resources that can be used to increase computational throughput. Second, we analyze the energy-performance trade-offs of employing both out-of-order and in-order cores in homogeneous processor configurations and outline their strengths and weaknesses for different workloads. Third, we also evaluate heterogeneous multi-core designs to determine if they are able to further improve the performance and energy trade-offs seen in homogeneous configurations.

We use detailed cycle-accurate full-system simulations and refined area and power models on an extensive list of relevant HPC benchmarks, covering a wide range of computation and communication patterns defined by the different HPC dwarfs [7, 8] to demonstrate that:

- A modestly sized LLC retains the performance of larger caches common in current HPC processors. Our findings contradict current trends that led to large LLC slices per core. We show performance returns diminish significantly as cache sizes increase, while imposing steep power/area costs that severely impact energy efficiency. We find that, a 6MB LLC compared

to a 16MB LLC ocupies 2.32× less area and consumes 2.46× less power - the equivalent to 6 out-of-order or 12 in-order cores, while performance degrades on average only by 1.9% and 5.1% for processors with 8 out-of-order and 16 in-order cores respectively. This freed power/area due to using a smaller LLC can be devoted to integrate more cores, thereby increasing overall processor throughput.
- When evaluating homogeneous configurations with out-of-order or in-order cores, we demonstrate that, overall, in-order cores offer better energy-performance trade-offs. For example, we show that 16 in-order cores consistently outperform 8 out-of-order cores (1.28× on average) while maintaining a lower area and power budgets, leading to significant energy delay product (EDP) gains of 1.84× on average. We also show that as designs integrate higher core counts, they may offer diminishing returns due to software overheads (e.g., runtime) and hardware constraints (e.g., memory bandwidth), which can hinder performance and energy efficiency.
- The use of heterogeneous core configurations can alleviate some of the identified core scaling challenges. We show that processor designs with an out-of-order core, to speed-up runtime events and sequential code regions, combined with a pool of in-order cores to maximize energy efficiency, delivers the best energy-performance trade-offs. Compared to an area and power equivalent homogeneous out-of-order processor, performance and EDP improve 1.26× and 1.71× on average, respectively.

2 Background and Motivation

HPC has experienced two major shifts in processor technology. First, by moving from the use of specific vector architectures to RISC commodity workstations and, in the mid 1990s, to CISC architectures used in contemporary commodity PCs. The same trends that triggered these technological shifts, such as higher manufacturing volumes, faster design cycles, and economic factors are starting to appear again for processors in the mobile market segment [1, 2].

However, in this particular shift there are two key differences: (i) there are numerous hardware integrators and manufacturers looking to expand their business due to the magnitude of the mobile market; and (ii) easier access to state-of-the-art processor designs due to *Arm*'s licensing business model. These factors create a highly competitive scenario with a leveled play-field, opening the door to tailored designs that are HPC specific, and for which custom decisions made by integrators and manufacturers are easier to implement.

Table 1 lists a set of characteristics across dominant and emerging processor architectures used in HPC. As can be seen in the top four rows of the table, dominant architectures integrate fewer cores but dedicate a significant amount of resources per core, providing deep out-of-order pipelines and speculation with wide frontend and backend engines; or in the case of BlueGene/Q, wide simultaneous multi-threading (SMT) and two in-order pipelines per core. In addition, dominant architectures have large LLC slices per core, and provide

Table 1 Dominant (top) and emerging (bottom) architectures. Ch. – Channels; BW – Bandwidth

Processor family	Cores		LLC	DRAM	
	Count	Type	Size (MB)	Ch.	BW
Intel Xeon	8-16	Aggressive OoO	20-50MB	4	76GB/s
AMD Opteron	8-16	Aggressive OoO	16MB	4	51GB/s
SPARC64	8-32	Aggressive OoO	12-24MB	8	120GB/s
BlueGene/Q	16	Aggressive SMT	32MB	4	43GB/s
Sunway MPP	260	Simple CPE*	17MB**	4	136GB/s
Intel Xeon Phi	64-72	Moderate OoO	32-36MB	6	115GB/s
Cavium ThunderX	48	Moderate OoO	16MB	4	55GB/s

* 4 Management Processing Elements (MPE) + 256 Compute Processing Elements (CPE)
** 256KB per MPE + 64KB scratch per CPE

as many memory channels as pin count permits. On the other hand, emerging architectures feature higher core counts but employ simpler designs. For example, the Sunway MPP [9, 10] has 256 simple in-order compute cores with a small 64KB scratchpad memory, while both the Intel Xeon Phi[1] [11] and the Cavium ThunderX [3] employ modified core architectures that come from the low-power mobile market domain. Despite retaining out-of-order capabilities, these cores have shallower pipelines, narrower frontend and backend engines, and downsized support for speculation; which can lead to hefty power reductions when compared to an aggressive out-of-order core, and several times less area [12, 13]. Emerging architectures also have significantly less LLC per core and, as in dominant architectures, offer as much off-chip bandwidth as possible.

Exploring fundamental power, area, and performance trade-offs when designing these emerging processors is essential to obtain a balanced architecture that attains high performance for certain power and area envelopes. In this paper, we investigate these trade-offs focusing on three key design factors: (i) the amount of LLC, (ii) the use of moderate out-of-order as well as simpler in-order cores, and (iii) heterogeneous core configurations. We motivate the importance of these three design points in the reminder of this section. The following results are obtained from detailed simulations and estimations over a comprehensive set of benchmarks. Additional details on our methodology can be found in Section 3.

Figure 1 shows the amount of area and power consumption for different LLC sizes and 8 moderate out-of-order cores. As expected, the LLC area and power increase almost linearly with cache size since most of the transistor budget is used for data cells and power is dominated by leakage. We can observe that a 6MB LLC has an area equivalent to 8 cores with moderate out-of-order capabilities, while the cores consume as much power as 16MB of LLC. Sizing the amount of LLC needs careful consideration in order to minimize area and power consumption which may allow fitting more cores into the design, thereby

[1] The Xeon Phi product line has been discontinued

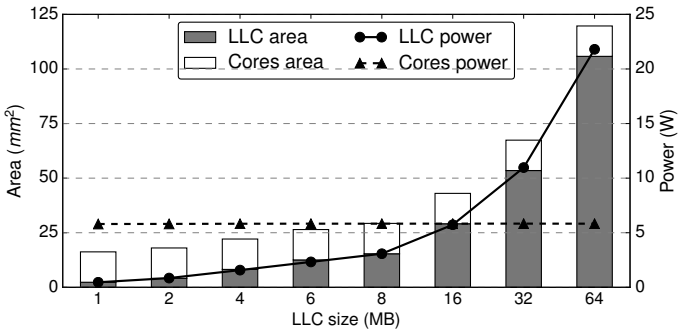


Fig. 1 Area and average power over a set of benchmarks (see Table 5) for different LLC sizes and a constant number of 8 moderate out-of-order cores.

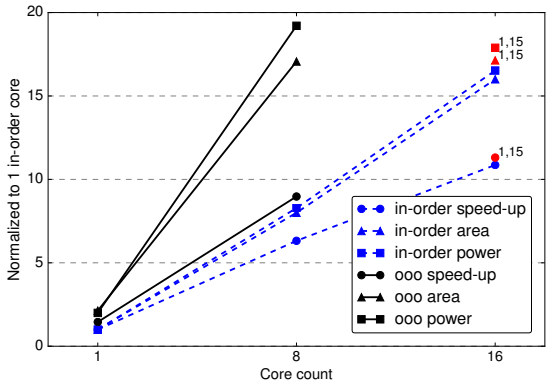


Fig. 2 Performance, area, and power trade-offs for moderate out-of-order and in-order cores over a set of benchmarks (see Table 5).

increasing computational efficiency. Over-dimensioning the LLC can quickly lead to significant power and area overheads with limited returns in performance. Therefore, it is important to determine at which point the performance returns of a larger cache are not worth the additional overheads.

Figure 2 shows power, area, and performance trade-offs for out-of-order and in-order cores that are typically found in the mobile market segment. The power and area estimates in this figure only consider the cores and their respective private caches to focus on the actual core trade-offs. As can be seen in the figure, a system with a single out-of-order core is 1.45× more performant than the in-order counterpart. However, performance gains come at a 2× cost in area and power. Similar trade-offs have also been observed in the datacenter and general-purpose domains [12, 14, 15, 16], showing that their findings may also apply to HPC benchmarks, and suggesting that further studies are needed to better understand these trade-offs at a finer granularity, i.e., per benchmark type. A consequence of these trade-offs is that 16 in-order cores perform significantly better than 8 out-of-order cores with lower area and power consumption. Also note that the gap between performance and

both area and power increases faster for out-of-order than for in-order cores, a trend that becomes increasingly important as on-chip core counts scale up.

Figure 2 also contains data points for a heterogeneous configuration that features 1 out-of-order core and 15 in-order cores - termed `(1,15)`. This design performs better than the 16 in-order configuration at the expense of increased area and power consumption. Having one or more fast cores in a processor can be beneficial, especially as thread counts increase since sequential sections of code or actions such as communication become more relevant and can hinder performance significantly due to Amdahl's law [17]. Also, as more cores are integrated, the area and power overheads of including one or a few bigger cores becomes less relevant.

3 Experimental Methodology

We evaluate performance, area, and power of emerging HPC processors by combining cycle-accurate simulations and fine-tuned modeling tools over a comprehensive set of benchmarks.

Performance Evaluation: We use gem5 [18] for cycle-accurate full-system simulations of various processor configurations. Gem5 is an open-source simulator that has received significant contributions from the industry (i.e., both *Arm* and *AMD*) in recent years. The simulator faithfully models microarchitectural details of the out-of-order and in-order cores, the cache hierarchy and the memory subsystem; including the on-chip interconnect, contention for shared resources, off-chip memory channels, DRAM bank conflicts, etc. The simulator models the ARMv8-A ISA and boots a recent linux kernel v4.3. For all benchmarks we simulate the entire region of interest and collect time to solution among other numerous statistics.

We model processors from 1 to 32 cores and various cache sizes. Out-of-order cores are modeled after an ARM Cortex-A72, while in-order cores resemble an ARM Cortex-A53. Table 2 details simulated architectural parameters.

Area and Power Evaluation: Based on the parameters introduced in Table 2, we estimate area and power for the different processor configurations using McPAT 1.3 [19] with 22nm technology models and an on-chip supply voltage of 0.8V. We have enhanced the McPAT models by applying the modifications proposed by Xi et al. [20], which greatly increase accuracy by modeling more core structures and by solving erroneous modeling assumptions. We feed McPAT models with detailed architectural input descriptions and statistics from our performance simulations. In addition, Xi et al. note that McPAT should not be used for uncore components - e.g. network-on-chip or memory controllers. For this reason, we perform estimations just for the cores (including private caches) and the LLC, for which we employ CACTI 6.5 [21]. These two structures are by far the dominant contributors to both area and power [19].

Table 2 Parameters for full-system simulations

Processor size	1 – 32 cores.
Cores	*out-of-order:* 3-wide issue/retire, 40-entry instruction queue, 128-entry ROB, 32 LDQ + 32 STQ, 2GHz *in-order:* 2-wide issue, 5-entry store buffer, 2GHz
L1 I/D Caches	*out-of-order:* L1I: 48KB, 3-way, 2 cycle, 2 ports, 8MSHRs LID: 32KB, 2-way, 2 cycle, 2 ports, 16MSHRs *in-order:* L1I: 32KB, 2-way, 2 cycle, 1 port, 8MSHRs LID: same as out-of-order but with 1 port
Last-level Cache	1 – 64MB, 16-way, 64B lines, 8 banks, 64MSHRs See Table 4 for data bank access latencies
NoC	Coherent crossbar, 128-bit wide, 2 cycles
Main Memory	4 DDR4-2400 channels, 2 ranks/channel, 16 banks/rank, 8KB row-buffer 128-entry write and 64-entry read buffers per channel 75GB/s peak bandwidth. Bank conflicts and queuing delays modeled

Table 3 Single core area and power estimations

Core	**in-order**		**out-of-order**	
	Absolute	Normalized	Absolute	Normalized
Area (mm^2)	0.82	1.0×	1.74	2.12×
Power (W)	0.30	1.0×	0.61	2.03×

Table 4 Timing, area, and power estimations for a 16-way 8-banked LLC

	Last-level Cache Sizes (MB)							
	1	2	4	6*	8	16	32	64
Latency (c)	9	9	9	9	10	11	13	17
Area (mm^2)	2.32	4.11	8.20	12.54	15.36	29.11	53.49	105.78
Power (W)	0.46	0.85	1.58	2.33	3.08	5.74	10.97	21.81

* Due to geometry constrains it uses 6 banks and 12-ways.

Table 3 shows area and power estimations for a single out-of-order and in-order core. Both absolute and relative numbers normalized to the in-order core are shown. Power numbers are the average across all evaluated benchmarks. Table 4 shows timing, area, and power estimations for the LLC. We evaluate systems that sweep the LLC size from 1MB to 64MB. The latency reported by CACTI is used in our performance simulations for LLC data bank access latency.

Throughout our evaluation, area and power numbers include both the cores (with private caches) and the LLC. We evaluate homogeneous as well as heterogeneous processor configurations. For homogeneous processors we evaluate a system with 8 out-of-order cores, denoted `(8,0)`; and systems with 16, 24, and 32 in-order cores, denoted `(0,16)`, `(0,24)`, and `(0,32)` respectively. In the `(0,16)` configuration we substitute each out-of-order core for 2 in-order cores, which leads to a configuration that features lower area and power con-

Table 5 Evaluated benchmarks.

Benchmark	HPC dwarf [7, 8]	Parallel construct	Short description	Problem size	Memory Footprint
bodytrack [22, 23]	dense linear	Tasks	Employs an annealed particle filter to track the 3D pose of a human body using edges. Computation and memory intensive.	36 frames 2K particles	233MB
blackscholes	dense linear	Tasks	Solves the Black-Scholes partial differential equation (PDE). Limited by amount of FP operations a processor can perform.	64K options	8.3MB
streamcluster	dense linear	Tasks	Clustering algorithm for a stream of input points. Presents balanced memory and computation intensity.	64K points 128 dimensions	136MB
HPCCG [24]	sparse linear	Loops	Local symmetric Gauss-Seidel conjugate gradient. Solves a sparse linear system on a 27-point 3D grid. Memory intensive.	161^3 nodes	1,591MB
facesim	sparse linear	Tasks	Simulates the underlying physics of human face movement.	1 frame 372K mesh	177MB
fluidanimate	sparse linear n-body	Tasks	Uses an extension of the Smoothed Particle Hydrodynamics (SPH) method to simulate an incompressible fluid.	3 frames 500K particles	117MB
CoMD	n-body	Loops	Classical molecular dynamics algorithms that evaluates all forces between atom pairs within a cutoff distance.	500K atoms	132MB
canneal	unstructured	Tasks	Cache-aware simulated annealing. Performs little computation and is memory intensive with a pseudo-random access pattern.	400K elements	103MB
LULESH [25]	unstructured	Loops	Represents a typical hydrocode and uses an unstructured hex mesh. Has high instruction and memory-level parallelism.	90^3 elements	705MB
heat	structured	Tasks	Iterative solver for heat distribution using a Gauss-Seidel algorithm with 5-point stencil computations.	8K resolution 2 heat sources	516MB
fft [26]	spectral methods	Loops	Computes a one dimensional Fast Fourier Transform.	16M elements	260MB
graph500 [27]	graph algorithms	Loops	Breadth-First Search of a large graph. High memory reference density with a near-random access pattern.	2^{20} vertices	285MB
swaptions	embarrassingly parallel	Tasks	Monte Carlo simulations to solve PDEs to price a portfolio of swaptions. Highly parallel with medium computation intensity.	256 swaptions 5K simulations	1.4MB
XSBench [28]	embarrassingly parallel	Loops	Kernel of the Monte Carlo neutronics application OpenMC. Stresses the memory system but performs little computation.	34 nuclides 1M lookups	140MB

sumption (see Figure 2). The `(0,24)` and `(0,32)` configurations are more aggressive and use 3 and 4 in-order cores per out-of-order, respectively. For heterogeneous systems we evaluate the intermediate configurations, i.e, for the 2 in-order per out-of-order: `(6,4)`, `(4,8)`, `(2,12)`, and `(1,14)`; as well as one totaling 16 cores with a single out-of-order - `(1,15)`. Similarly for the other two more aggressive configurations.

Evaluated Benchmarks: Table 5 lists the evaluated benchmarks. We have classified them according to the computational *dwarf* [7, 8] they belong, i.e., dense linear algebra, unstructured grid, etc. We cover all the HPC-relevant dwarfs with at least one representative. For the most important ones, we use more than one benchmark to stress different workload characteristics that may

appear within a dwarf, leading to an extensive coverage in terms of computational patterns.

Table 5 also includes a short description with insight about the benchmark characteristics, the problem size employed, and the memory footprint. We have spent significant effort ensuring all input problem sizes are representative and deliver real-world LLC behavior for all the evaluated LLC sizes. For benchmarks that admit arbitrarily large problem sizes we selected representative real-world values that allow for feasible simulation time.

Benchmarks that employ parallel loops to expose parallelism have been configured to always use dynamic scheduling to perform well under heterogeneous configurations. Similarly, task-based parallelism which is increasingly common in HPC [29], also employs dynamic scheduling. When evaluating heterogeneous configurations we force the master thread to execute in an out-of-order core at all times. The master thread executes the sequential code regions between parallel sections and generates the loop chunks or tasks to be executed by all available cores dynamically.

4 Design Trade-offs Evaluation

4.1 Last-level Cache Size

The rapid growth in transistor count per chip has led to an increase in the number of cores and LLC sizes. In dominant HPC architectures, per core LLC slices have increased substantially in recent years despite the large area and power cost associated. These large caches can quickly become a dominant component in terms of area and power in emerging processor architectures, since less aggressive cores are employed. Therefore, if the performance returns of a larger cache are not significant, the overall energy efficiency of the processor can be affected deeply, making LLC sizing a key design decision. To gain insight on the appropriate design points for emerging HPC processors, we evaluate the performance benefits of employing various cache sizes using the parameters presented in Table 4 over four homogeneous processor configurations, one with eight out-of-order cores `(8,0)`, and three with 16, 24, and 32 in-order cores - termed `(0,16)`, `(0,24)`, and `(0,32)` respectively.

Figure 3 shows overall system performance normalized to 1MB of LLC for each evaluated configuration. The following paragraphs analyze the results per computational dwarf.

Dense linear algebra: *blackscholes* is computationally intensive and performance is limited by the amount of floating point throughput the processor can achieve. Performance improves until the relevant portions of the working set fit in the cache, which happens at 6MB. For larger caches, performance starts to degrade due to higher access latencies. Similar behaviors are observed in *bodytrack* and *streamcluster*, with timid performance improvements after 6MB of LLC that are not worth the steep increase in area and power costs. Most dense linear algebra benchmarks rely on blocking to partition the

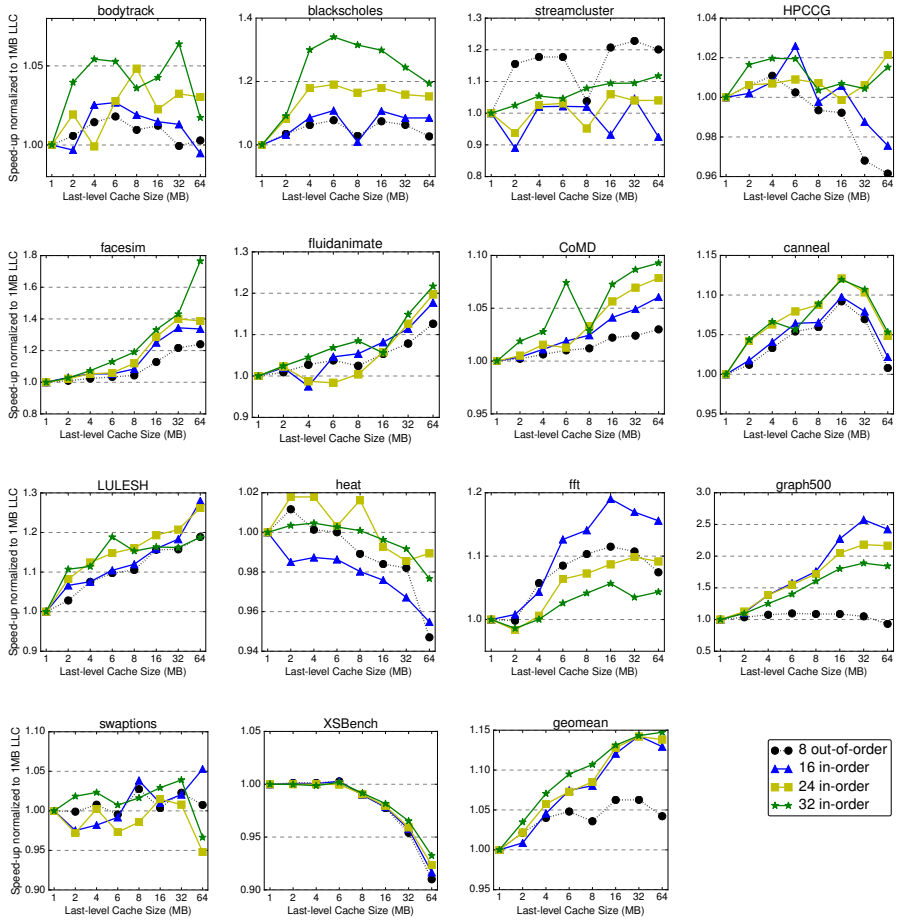


Fig. 3 Last-level cache size sensitivity study.

work, and modest cache sizes are sufficient to hold the working sets for the evaluated core counts.

Sparse linear algebra and n-body methods: Both *HPCCG* and *facesim* contain a conjugate gradient kernel. In the former, data is represented as a 3D grid to do 27-point stencil calculations, while the latter uses a tetrahedral mesh. *HPCCG* is not sensitive to LLC size because it has low data reuse and the main bottleneck at higher core counts is off-chip memory bandwidth, with over 17 misses per kilo instruction (MPKI) for `(0,32)`. On the other hand, *facesim* attains significant performance improvements as cache sizes increase, as data can be reused across the three different kernels that constitute an iteration. With large caches a good portion of the mesh can be stored in the cache, which allows for an even higher level of reuse as nodes may be visited multiple times.

The n-body benchmarks partition the space and calculate forces between close particles. Therefore, working set sizes for the partitions are manageable. *CoMD* achieves modest improvements when going above medium sized caches, i.e., 6-8MB. *Fluidanimate* performs better with the largest cache sizes, up to 1.23× over the 1MB setup for `(0,32)`. However, the costs of increasing the cache to these sizes outweigh the performance benefits over a modest 6-8MB cache size.

Unstructured and structured grids: Unstructured grid benchmarks like *LULESH* or *canneal* have complex memory access patterns that lead to poor spatial locality. As shown in the figure, cache sizes of 6-8MB already obtain most of the benefits for all core counts. LULESH further improves performance with larger cache sizes, but in the best case it would be by 14% for a cache that increase from 8MB to 64MB, which would greatly degrade energy efficiency. *Canneal* obtains limited improvements across core counts by doubling the cache from 8MB to 16MB, and for larger caches performance degrades due to increased access latencies. On the other hand, structured grids have more regular access patterns, however, data reuse is low. As can be seen in *heat*, increasing the cache size can actually decrease performance as the benchmark is more sensitive to latency than to capacity.

Spectral methods: The main characteristic of this computational dwarf is the employed all-to-all communication pattern. In *fft*, maximum performance is achieved at 16MB, however, it offers diminishing returns for caches larger than 6MB. Therefore, as happens in most benchmarks that employ blocking techniques, modest cache sizes are sufficient.

Graph algorithms: *graph500* traverses a large graph generating memory reference bursts that have near-random access patterns. Since the memory references are independent, out-of-order cores perform significantly better by issuing them in parallel without blocking the pipeline. For `(8,0)` performance improvements saturate at 6MB of LLC cache size. However, in-order cores perform poorly since they are unable to exploit the fact that memory references are independent and there is no spatial locality. Larger caches alleviate these issues by caching more visited nodes that might be revisited later, achieving large improvements of over 2.5× for 16 cores.

Embarrassingly parallel: We evaluate two highly parallel benchmarks with different characteristics. *swaptions* has medium computation intensity and a small dataset, therefore cache size does not affect performance in a significant manner. On the other hand, *XSBench* performs little computation and stresses the memory hierarchy demanding a lot of off-chip bandwidth, with over 42 MPKI for all evaluated configurations. Therefore, the LLC access latency is on the critical path and performance degrades for large caches.

Conclusions: On average, and for most benchmarks, we find that a 6MB LLC is a good design point for all evaluated core counts. This observation differs from the trends followed in dominant architectures, which employ significantly larger cache sizes. Compared to a 16MB LLC, a 6MB LLC ocupies 2.32× less area and consumes 2.46× less power with small or non-existent performance penalties, that on average are 1.9% for out-of-order cores and 5.1%

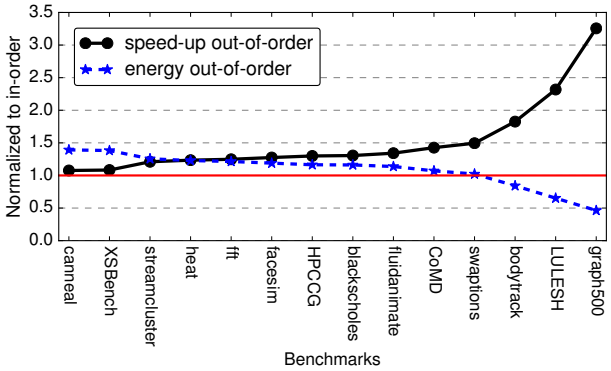


Fig. 4 Speed-up and energy-efficiency of a single out-of-order with respect to an in-order core.

for 16 in-order cores (worse case). *graph500* and *facesim* could benefit from a large cache, however, in the case of *graph500* it is patching an underlying problem that is related to in-order cores not being suited for this benchmark, and in the case of *facesim* the improvements that would be achieved do not justify the necessary investment for a 16MB or 32MB cache just for this benchmark. We advocate that the freed area and power should be devoted to integrate more cores. This approach will make future processor designs for HPC more performant and energy efficient. The reminder of this paper assumes a 6MB LLC.

4.2 Out-of-Order vs In-Order Execution

Performance and energy implications of using out-of-order or in-order execution greatly depend on the characteristics of the executed benchmark. Benchmarks that have high instruction-level parallelism (ILP), i.e., independent instructions that can be issued in parallel, or memory-level parallelism (MLP), i.e., multiple inflight memory requests, greatly benefit from out-of-order execution. Contrarily, if memory references or executed instructions are dependent, in-order cores can provide competitive performance. These observations are well known, however, their impact in terms of performance and energy for emerging HPC processor designs and how they affect specific benchmarks remains to be seen.

Figure 4 shows performance and energy ratios for a single out-of-order core over a single in-order core for all benchmarks using their serial implementations. Energy calculations take into account the core as well as the 6MB LLC. *Canneal* and *XSBench*, which are memory intensive but cannot expose enough MLP due to dependent access patterns, obtain almost no benefit from out-of-order execution - i.e., only 1.08× faster than in-order while consuming 1.38× more energy. At the other end of the spectrum we find *bodytrack*, *LULESH*, and *graph500*, which have abundant ILP and MLP. In these scenarios, out-of-

Table 6 Area and power estimations for evaluated homogeneous core configurations.

Cores	**out-of-order**	**in-order**		
	8	16	24	32
Area (mm^2)	1.0×	0.93×	1.40×	1.87×
Power (W)	1.0×	0.86×	1.28×	1.71×

order execution performs much better and attains speed-ups of 1.83×, 2.31×, and 3.25× respectively, with significant energy savings. Between these two extremes we find most benchmarks, with speed-ups using out-of-order that range from 1.21× (*streamcluster*) to 1.49× (*swaptions*). However, for all these benchmarks the in-order core is more energy-efficient. These results show that in-order processors could outperform out-of-order processor configurations under the same area and power budgets.

Next, we evaluate homogeneous core configurations with 8 out-of-order cores `(8,0)` and 16 `(0,16)`, 24 `(0,24)`, and 32 `(0,32)` in-order configurations. Table 6 shows area and power estimations for these configurations normalized to `(8,0)`. Both the cores and the LLC are considered. The `(0,16)` configuration is below `(8,0)` in terms of area and power, while the more aggressive `(0,24)` and `(0,32)` configurations are above.

Figure 5 shows energy-performance trade-offs for the evaluated homogeneous configurations. We plot energy efficiency in nano-joules (nJ) per flop (left axis), and energy delay product (EDP) normalized to `(8,0)` (right axis). Note that we plot the inverse of the normalized EDP - higher is better. Performance is presented as scalability over a single in-order core in the x-axis.

Dense linear algebra: Both *bodytrack* and *blackscholes* present similar trends with significant improvements in energy efficiency when moving from `(8,0)` to in-order configurations; up to 1.22× and 1.60× respectively. For `(0,32)` there is a slight increase in energy due to lower gains in performance scaling at higher thread counts. However, in terms of EDP, `(0,32)` is the best configuration as the gains in performance offset the loss in energy efficiency, reaching 2.10× and 3.16× better EDP than `(8,0)` respectively. *streamcluster* shows similar trends for `(0,16)`, but the application fails to scale well with more threads which hinders both energy efficiency and EDP, reaching maximum EDP with `(0,24)`.

Sparse linear algebra and n-body methods: *HPCCG* and *facesim* also obtain significant energy reductions and performance improvements with `(0,16)` over `(8,0)`. However, the memory bound nature of this benchmarks limits scalability at higher core counts. As we can observe, with more than 16 cores *HPCCG's* performance stagnates and there is a sharp drop in EDP, while in *facesim* EDP continues improving as its scalability improves sufficiently.

On the other hand, both n-body benchmarks, *fluidanimate* and *CoMD*, fare significantly better under in-order configurations, achieving large energy reductions per flop. In addition, both scale well as core counts increase due to their simple yet effective parallelization scheme that has a good balance

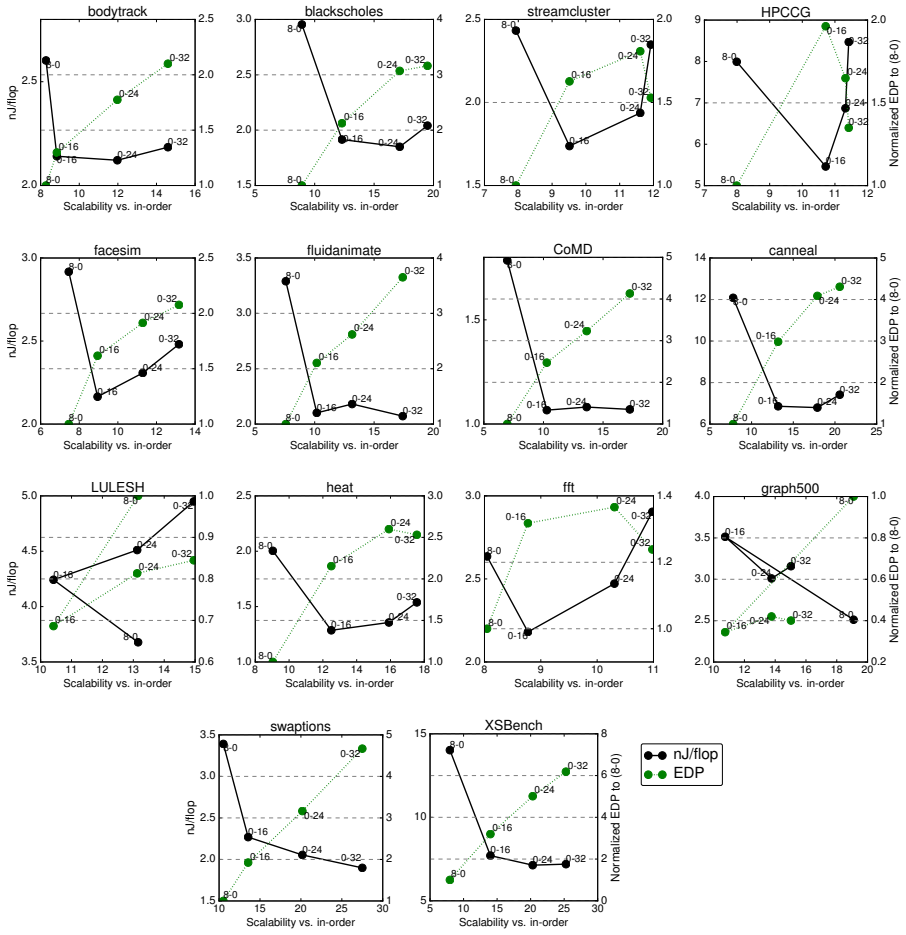


Fig. 5 Energy-performance trade-offs for evaluated homogeneous processors.

between computation and memory intensity. These two factors lead to large EDP improvements with `(0,32)`, of 3.65× and 4.13× over `(8,0)`, respectively.

Unstructured and structured grids: The unstructured grid benchmarks exhibit disparate behaviors. *canneal* is memory intensive but references are dependent and cannot be issued in parallel, therefore, in-order performance is almost as good as out-of-order. The obtained results are similar to those seen for n-body benchmarks, but with a costlier energy per flop rating due to memory intensity. However, LULESH benefits greatly from out-of-order execution due to high ILP and MLP. For this reason, `(8,0)` performs better than `(0,16)` and on par with `(0,24)`. Even though `(0,32)` outperforms `(8,0)`, it does so at a much higher energy cost, and is therefore not able to beat the baseline EDP. *heat* exhibits good performance scaling up to `(0,24)`, with a peak EDP

improvement of 2.6× over `(8,0)`. Performance is limited for `(0,32)` due to a lack of tasks to feed all cores, i.e., task creation is not fast enough.

Spectral methods: *fft* has limited scalability at high core counts due to its costly all-to-all communication patterns. For this reason, it scales reasonably well with up to 16 cores, but further increasing core count delivers diminishing returns. Nonetheless, `(0,16)` offers better energy trade-offs and outperforms the `(8,0)` configuration.

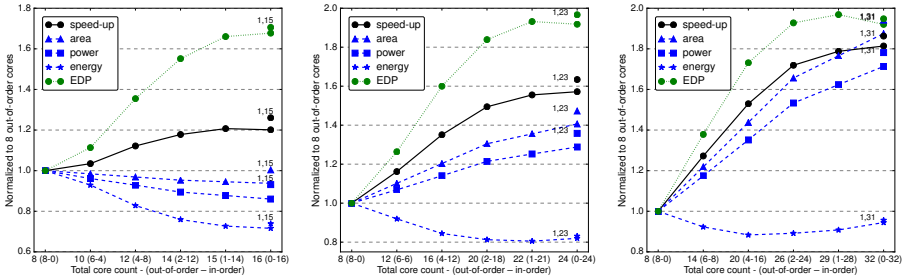
Graph algorithms: *graph500* has ample MLP that can be exploited by out-of-order execution. As observed in *LULESH*, this makes `(8,0)` the most energy efficient configuration with the best EDP metric. It is worth mentioning that GPUs are a good match for this particular benchmark [30, 31], as they have abundant memory bandwidth and hundreds of cores available.

Embarrassingly parallel: As expected, both *swaptions* and *XSBench* attain the best scalability, with energy reductions of 1.79× and 1.97×, and EDP improvements of 4.67× and 6.19×, respectively. Note that due to *XSBench* memory intensity, the energy spent per flop is significantly larger than in most of the other benchmarks.

Conclusions: *LULESH* and *graph500* did not benefit from in-order configurations due to their high ILP and MLP that favors out-of-order execution. However, we can observe a clear trend where `(0,16)` consistently outperforms `(8,0)` in terms of energy per flop (1.5× on average), performance (1.28× on average), and EDP (1.84× on average). Note that `(0,16)` also has lower area and power envelopes than `(8,0)`. This indicates that in-order cores are good candidates and can play a major role in future HPC processor designs. In addition, almost all benchmarks obtain further performance and EDP improvements with `(0,24)` and `(0,32)`. Exceptions include benchmarks that are limited by the amount of available work or memory bandwidth (*HPCCG*). We also observe that as more cores are present, performance scaling is increasingly difficult due to both hardware and software limitations. An example of the former is poor memory bandwidth scaling with respect to core count, while an example of the latter is the overheads associated with managing more threads and the limits imposed by Amdahl's law [17]. These limitations can be solved by using appropiate technologies like high-bandwidth memory, and by exploiting heterogeneous configurations as we show in the following section.

4.3 Heterogeneous Core Configurations

Figure 6 shows the different trends in terms of performance, area, power, energy and EDP for several evaluated heterogeneous configurations. Numbers are averaged across all evaluated benchmarks to grasp the general trends. Both the cores and the LLC are considered for area and power estimations. Each of the three plots starts with the homogeneous `(8,0)` configuration and gradually shifts towards `(0,16)`, `(0,24)`, and `(0,32)` configurations respectively. For example, Figure 6a shows results for `(8,0)`, `(6,4)`, `(4,8)`, `(2,12)`, `(1,14)`,



(a) From 8 ooo to 16 in-order (b) From 8 ooo to 24 in-order (c) From 8 ooo to 32 in-order

Fig. 6 Average trends for evaluated heterogeneous processors normalized to `(8,0)`. Area, power and energy lower is better. For speed-up and the inverse of normalized EDP higher is better.

and `(0,16)`; as well as a 16-core configuration with a single out-of-order core - `(1,15)`.

In Figure 6a all heterogeneous designs occupy less area and consume less power than `(8,0)`. As we have shown in Table 3, out-of-order cores are more than 2× the area and power of an in-order core. Performance increases as more in-order cores are added and it peaks at the configurations that have one out-of-order core, `(1,14)` and `(1,15)`. The latter being the best both in performance and EDP. Figures 6b and 6c show more aggressively sized heterogeneous designs where area and power is above that of the baseline `(8,0)` configuration. As in the previous figure, as more in-order cores are added both performance and EDP improve. The best results are again obtained by configurations that feature at least 1 out-of-order core. This suggests that having an out-of-order core can be useful in certain scenarios to help improve pitfalls that occur in homogeneous processor designs. We investigate this further in the following paragraphs. We can also observe that on average `(1,23)` has better EDP than `(1,15)`, however, `(1,31)` is slightly below, which illustrates the difficulties to scale to high core counts.

Figure 7 shows energy-performance trade-offs for heterogeneous configurations. Energy efficiency is again measured in nJ per flop, and performance using scalability over an in-order core. We plot 3 lines with several data points. Each line starts with `(8,0)`, the intermediate data points are the heterogeneous configurations, adding in-order cores in detriment of out-of-order ones. We again plot a data point that has the maximum number of total cores and one of them is out-of-order, i.e., `(1,15)`, `(1,23)`, and `(1,31)`.

Dense linear algebra: For *bodytrack* and *blackscholes* both performance and energy efficiency improve as in-order core counts increase. For `(0,32)` we observe worse energy efficiency than in `(1,28)`, and in the case of *blackscholes* performance also degrades due to scalability limitations. In general, `(1,15)`, `(1,23)`, and `(1,31)` are the best performing, sometimes at a slight cost in terms of energy. In *blackscholes*, `(1,31)` significantly improves performance by 7.4% over `(0,32)`. By using output traces from our simulations we have been able to verify that the bottleneck for `(0,32)` is task creation, i.e., running out

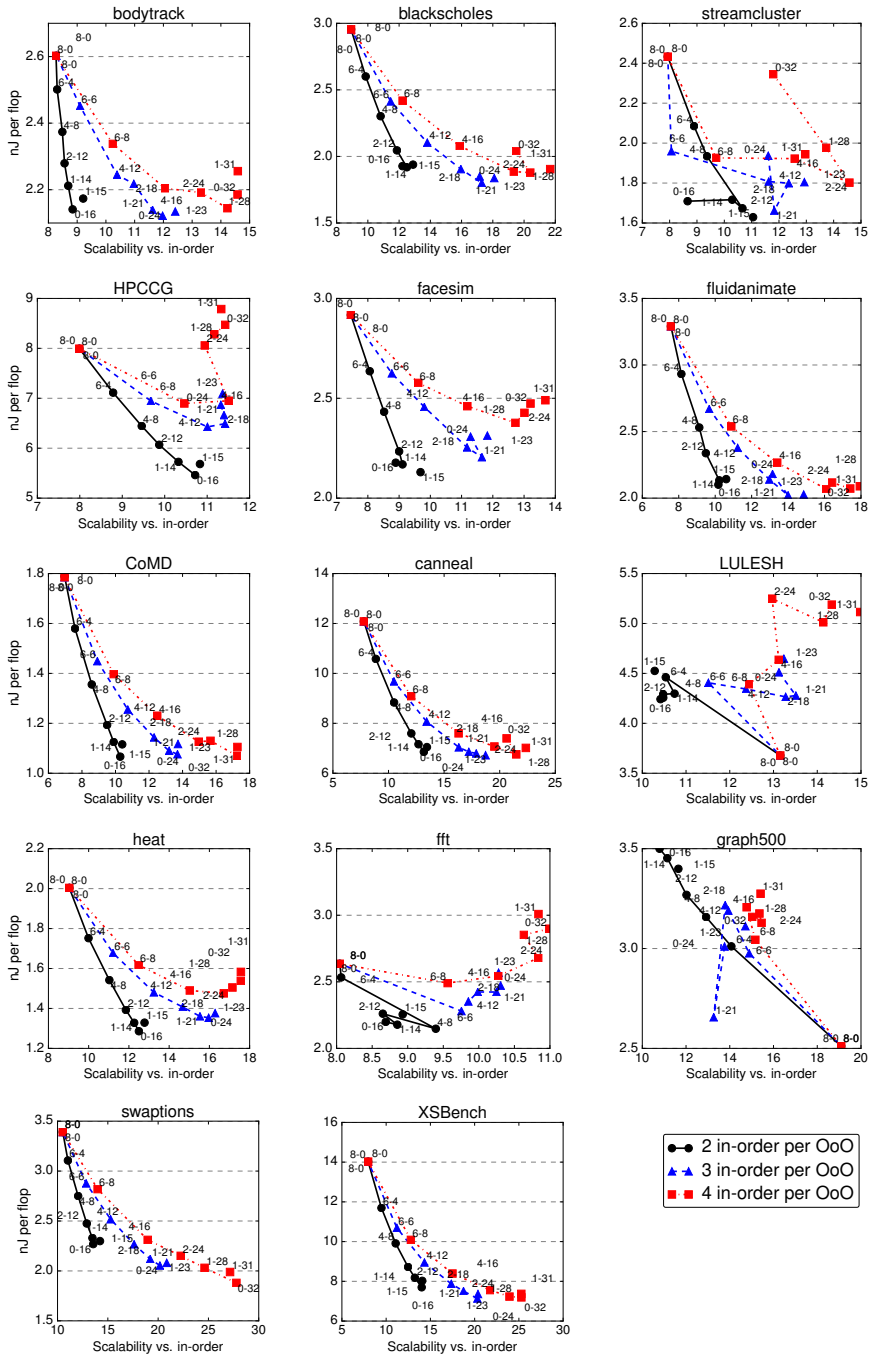


Fig. 7 Energy-performance trade-offs for evaluated heterogeneous processors.

of available work due to tasks not being created fast enough. By having a faster out-of-order core, we can ensure that all in-order cores have tasks available to execute at all times. Streamcluster presents an erratic behavior due to load imbalance that is difficult to control in heterogeneous configurations since its regular execution pattern is sensitive to thread count.

Sparse linear algebra and n-body methods: *HPCCG* becomes memory bound as core count increases. With more than 16 - 20 cores performance stagnates and energy efficiency starts to degrade. The `(0,16)` configuration for which memory is still not a problem is the most efficient in terms of energy. *Facesim* loses performance in `(0,16)` and `(0,24)` due to the inability to timely provide work for all cores. However, when an out-of-order core is used, both performance and energy efficiency improve. This is due to the ability of the out-of-order core to generate tasks faster, providing enough work timely for the in-order cores.

fluidanimate and *CoMD* present similar trends, where having one out-of-order core improves performance due to faster task creation in *fluidanimate* and lower thread management overheads in the case of *CoMD*, further showing the positive impact a faster core can have in alleviating runtime and thread management overheads.

Unstructured and structured grids: In *canneal*, `(1,15)`, `(1,23)`, and `(1,31)` improve performance over homogeneous in-order configurations. In our traces we observe that `(0,32)` presents load imbalance for the thread that creates tasks, which always finishes its tasks with a bit of delay after all other threads reach the barrier. This is solved by having an out-of-order core doing task creation, eliminating load imbalance. In *LULESH* energy efficiency degrades as the number of in-order cores increases. The best performing configuration is `(1,31)`, but at a steep energy per flop cost compared to `(8,0)`, which offers the best trade-offs. In *heat* scalability improves notably as in-order cores are added. Again, the best performing configurations include an out-of-order core, with an affordable energy cost.

Spectral methods: Due to its all-to-all communication pattern, *fft*'s performance scaling degrades as total core count increases. The first step that increases core counts improves performance significantly compared to the following steps, for which performance stagnates or even degrades, leading to worse energy efficiency.

Graph algorithms: As explained before, *graph500* favors out-of-order execution. As more in-order cores are added, both performance and energy efficiency suffer severe degradation.

Embarrassingly parallel: Both *swaptions* and *XSBench* scale well as total core count increases. In *swaptions* `(1,15)` and `(1,23)` perform best in their respective data lines, however, `(1,31)` loses a bit of performance due to load imbalance. *XSBench* does not see any additional performance improvements when using an out-of-order core, yielding the same performance as the all in-order configurations, which have slightly better energy efficiency.

Conclusions: As shown in Figure 6a, `(8,0)` and `(1,15)` are area and power equivalent. However, `(1,15)` improves performance and EDP by 1.26×

and 1.71× respectively, while also improving performance over `(0,16)` by 4.9%. In addition, we have shown that speeding up runtime related events such as task creation or thread management in parallel loops is advantageous, specially for large core counts. The energy penalty of including an out-of-order core is not significant (see Figure 6) and becomes smaller as more cores are integrated. Having an out-of-order core can also help speed up initialization phases and other sequential code sections, which quickly become relevant bottlenecks as core counts increase [17].

5 Related Work

Prior work has identified significant over-provisioning in conventional server processors in the context of datacenters and warehouse computing, both in core capabilities and in the memory hierarchy [14, 32, 33]. These works focused on request-oriented workloads such as web search. For this type of server workloads they find that large instruction footprints limit system performance, which does not happen in HPC. Even though their findings differ from the ones discussed in this paper, there is a clear trend by different computing communities to move from dominant server architectures towards more specialized designs that can help reduce the costs while optimizing for the targeted metrics relevant to each space.

Albericio *et al.* [34] propose a cache design that only stores data for reused cache lines, reducing the size of the data array significantly. The paper is focused on architecting this design by introducing changes in the data allocation and replacement policies, as well as modifications in the coherence protocol. The authors find that storing a small subset of the cache lines yields good performance, hinting in passing that LLC sizes are over-provisioned. Similarly, Siddique *et al.* [35] show that cache locality for certain scientific applications is low, and that local memories are much more area and energy efficient than regular LLCs.

Huh *et al.* [36] explored the design space of CMP systems for in-order and out-of-order cores. They conclude that out-of-order cores are more area-efficient than in-order cores. However, on that study only performance and area was taken into account, the lack of power measurements clearly favors out-of-order designs. Moreover, as the paper authors mention, the use of SPEC2000 workloads may not be representative of HPC applications due to small data footprints, which makes workloads more compute bound, for which out-of-order cores are better suited. Laurenzano *et al.* [37] performed a thorough comparison of an out-of-order (Cortex A15) and an in-order (Cortex A9) processor using real test platforms. In their evaluation, the performance ratio of the out-of-order versus the in-order system was higher than the one we have observed in our experiments. However, this can be explained by the differences present in the employed real test platforms, which makes the actual core comparison difficult. Some of these differences include: LLC size (2MB vs. 4MB), DDR3 speed (1333MHz vs. 1600MHz), vectorization capabilities, and different

floating-point ISA support. In our experiments, however, the entire system is exactly the same for each core type.

Prior work has shown that mobile processors have promising qualities that make them candidates for HPC in the near future [1, 2]. Driven by a rapidly growing market that is demanding more and more performance, mobile processors have more aggressive roadmaps and integrate technological innovations faster. Several cluster prototypes built with low-power commodity processors started to be deployed in the early 2000s [38, 39, 40, 41]. Tibidabo was the first large-scale HPC cluster based on mobile processors [42]. Deployed in 2011, it had 256 nodes of dual-core ARM Cortex A9 processors and it was the first to have a full HPC software stack.

Since then, several companies have already developed custom processors based on the ARMv8-A architecture, e.g., Cavium's ThunderX [3]. In addition, leading HPC companies have already announced large ARM-based machines to be deployed in the near future, like the Fujitsu Post-K supercomputer [6] and a 10,000+ ARMv8-A system by Cray [4]. With the arrival of these machines, the shift towards processors based on mobile market technology will likely accelerate. Therefore, the study undertaken in this paper is a necessary first step to understand the energy-performance trade-offs present in processors based on mobile market technology.

6 Conclusions

In this paper we evaluate and quantify several design trade-offs for emerging HPC processors based on mobile market technology. We have focused our study on three main fronts: (i) last-level cache sizing, (ii) the use of out-of-order and in-order homogeneous configurations, and (iii) heterogeneous designs.

Contrary to the trends followed in dominant architectures, we find that a modestly sized LLC can significantly improve energy-efficiency without compromising performance. In addition, we show that in-order core designs offer better energy-performance trade-offs than out-of-order for a wide range of HPC benchmarks. Comparing `(8,0)` and `(0,16)`, the latter has, on average, better energy per flop (1.50×), performance (1.28×) and EDP (1.84×). Finally, heterogeneity can help alleviate software constrains that hinder performance scaling by speeding up runtime events using an out-of-order core that generates work for a pool of in-order cores, which in our evaluation delivers the best energy-performance trade-offs; improving performance (1.26×) and EDP (1.71×) compared to an area and power equivalent out-of-order processor.

Our study quantifies emerging trends that hardware vendors are starting to adopt and will soon make their way into production systems, providing a better understanding of the expected energy-performance implications in such systems. Our results show that future processor designs with a carefully sized LLC and that mainly employ simple in-order cores, can deliver better energy-performance characteristics than current dominant architectures, consolidating the shift towards HPC processors based on mobile market technology.

Acknowledgements This work has been partially supported by the European HiPEAC Network of Excellence, by the Spanish Ministry of Economy and Competitiveness (contract TIN2015-65316-P), and by the Generalitat de Catalunya (contracts 2014-SGR-1051 and 2014-SGR-1272). The Mont-Blanc project receives funding from the EUs H2020 Framework Programme (H2020/2014-2020) under grant agreements no. 671697 and no. 779877. M. Moreto has been partially supported by the Spanish Ministry of Economy, Industry and Competitiveness under Ramon y Cajal fellowship number RYC-2016-21104. M. Casas has been partially supported by the Secretary for Universities and Research of the Ministry of Economy and Knowledge of the Government of Catalonia and the Cofund programme of the Marie Curie Actions of the European Union (Contract 2013 BP B 00243). Finally, A. Armejach has been partially supported by the Spanish Ministry of Economy, Industry and Competitiveness under Juan de la Cierva postdoctoral fellowship number FJCI-2015-24753.

References

1. N. Rajovic, P. M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero, "Supercomputing with Commodity CPUs: Are Mobile SoCs Ready for HPC?" in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '13. New York, NY, USA: ACM, 2013, pp. 40:1–40:12. [Online]. Available: http://doi.acm.org/10.1145/2503210.2503281
2. N. Rajovic et al., "The mont-blanc prototype: an alternative approach for HPC systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16, 2016, pp. 444–455. [Online]. Available: https://doi.org/10.1109/SC.2016.37
3. L. Gwennap, "ThunderX Rattles Server Market," *Microprocessor Report*, vol. 29, no. 6, pp. 1–4, 2014.
4. M. Feldman, "Cray to Deliver ARM-Powered Supercomputer to UK Consortium," 2017, accessed: 2017-02-22. [Online]. Available: https://www.top500.org/news/cray-to-deliver-arm-powered-supercomputer-to-uk-consortium/
5. S. McIntosh-Smith, T. Deakin, and A. Poenaru, "Comparative Benchmarking of the First Generation of HPC-Optimised ARM Processors on Isambard," in *Cray User Group (CUG) Conference*, May 2018.
6. T. Yoshida, "Fujitsu High Performance CPU for the Post-K Computer," in *Hot Chips 30 Symposium (HCS)*, ser. Hot Chips '18. IEEE, 2018.
7. K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick, "The Landscape of Parallel Computing Research: A View from Berkeley," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-183, Dec 2006. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html
8. K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, and K. Yelick, "A view of the parallel computing landscape," *Commun. ACM*, vol. 52, no. 10, pp. 56–67, Oct. 2009. [Online]. Available: http://doi.acm.org/10.1145/1562764.1562783
9. J. Dongarra, "Report on the Sunway TaihuLight System," University of Tennessee, Oak Ridge National Laboratory, Tech. Rep. Tech. Rep. UT-EECS-16-742, 2016.
10. H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao, W. Zhao, X. Yin, C. Hou, C. Zhang, W. Ge, J. Zhang, Y. Wang, C. Zhou, and G. Yang, "The Sunway TaihuLight supercomputer: system and applications," *SCIENCE CHINA Information Sciences*, vol. 59, no. 7, pp. 072 001:1–072 001:16, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11432-016-5588-7
11. A. Sodani, R. Gramunt, J. Corbal, H. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. Liu, "Knights Landing: Second-Generation Intel Xeon Phi Product," *IEEE Micro*, vol. 36, no. 2, pp. 34–46, 2016. [Online]. Available: http://dx.doi.org/10.1109/MM.2016.25
12. Khubaib, M. A. Suleman, M. Hashemi, C. Wilkerson, and Y. N. Patt, "MorphCore: An Energy-Efficient Microarchitecture for High Performance ILP and

High Throughput TLP," in *45th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '12, 2012, pp. 305–316. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2012.36

13. M. Guevara, B. Lubin, and B. C. Lee, "Strategies for anticipating risk in heterogeneous system design," in *20th IEEE International Symposium on High Performance Computer Architecture*, ser. HPCA '14, 2014, pp. 154–164. [Online]. Available: http://dx.doi.org/10.1109/HPCA.2014.6835926
14. P. Lotfi-Kamran, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, S. Idgunji, E. Ozer, and B. Falsafi, "Scale-out processors," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ser. ISCA '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 500–511. [Online]. Available: http://dl.acm.org/citation.cfm?id=2337159.2337217
15. K. V. Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez, and J. Emer, "Scheduling heterogeneous multi-cores through performance impact estimation (PIE)," in *39th Annual International Symposium on Computer Architecture (ISCA)*, ser. ISCA '12, June 2012, pp. 213–224.
16. O. Azizi, A. Mahesri, B. C. Lee, S. J. Patel, and M. Horowitz, "Energy-performance Tradeoffs in Processor Architecture and Circuit Design: A Marginal Cost Analysis," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 26–36. [Online]. Available: http://doi.acm.org/10.1145/1815961.1815967
17. M. D. Hill and M. R. Marty, "Amdahl's Law in the Multicore Era," *Computer*, vol. 41, no. 7, pp. 33–38, Jul. 2008. [Online]. Available: http://dx.doi.org/10.1109/MC.2008.209
18. N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718
19. S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *42st Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '09, 2009, pp. 469–480. [Online]. Available: http://doi.acm.org/10.1145/1669112.1669172
20. S. L. Xi, H. M. Jacobson, P. Bose, G. Wei, and D. M. Brooks, "Quantifying sources of error in McPAT and potential impacts on architectural studies," in *21st IEEE International Symposium on High Performance Computer Architecture*, ser. HPCA '15, 2015, pp. 577–589. [Online]. Available: http://dx.doi.org/10.1109/HPCA.2015.7056064
21. N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 3–14. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2007.30
22. C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '08. New York, NY, USA: ACM, 2008, pp. 72–81. [Online]. Available: http://doi.acm.org/10.1145/1454115.1454128
23. D. Chasapis, M. Casas, M. Moretó, R. Vidal, E. Ayguadé, J. Labarta, and M. Valero, "PARSECSs: Evaluating the Impact of Task Parallelism in the PARSEC Benchmark Suite," *TACO*, vol. 12, no. 4, pp. 41:1–41:22, 2016. [Online]. Available: http://doi.acm.org/10.1145/2829952
24. P. S. Crozier, H. K. Thornquist, R. W. Numrich, A. B. Williams, H. C. Edwards, E. R. Keiter, M. Rajan, J. M. Willenbring, D. W. Doerfler, and M. A. Heroux, *Improving performance via mini-applications.*, Sep 2009. [Online]. Available: http://dx.doi.org/10.2172/993908
25. R. D. Hornung, J. A. Keasler, and M. B. Gokhale, "Hydrodynamics Challenge Problem," Lawrence Livermore National Laboratory, Tech. Rep. LLNL-TR-490254, 2011.

26. N. Rajovic, A. Rico, J. Vipond, I. Gelado, N. Puzovic, and A. Ramirez, "Experiences with Mobile Processors for Energy Efficient HPC," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '13, 2013, pp. 464–468. [Online]. Available: http://dl.acm.org/citation.cfm?id=2485288.2485400
27. R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Ang, "Introducing the graph 500," Cray User's Group, Tech. Rep., May 2010.
28. J. R. Tramm, A. R. Siegel, T. Islam, and M. Schulz, "XSBench – The development and verification of a performance abstraction for Monte Carlo reactor analysis," *The Role of Reactor Physics toward a Sustainable Future (PHYSOR)*, 2014.
29. P. Thoman, K. Dichev, T. Heller, R. Iakymchuk, X. Aguilar, K. Hasanov, P. Gschwandtner, P. Lemarinier, S. Markidis, H. Jordan, T. Fahringer, K. Katrinis, E. Laure, and D. S. Nikolopoulos, "A taxonomy of task-based parallel programming technologies for high-performance computing," *The Journal of Supercomputing*, vol. 74, no. 4, pp. 1422–1434, Apr 2018. [Online]. Available: https://doi.org/10.1007/s11227-018-2238-4
30. T. Suzumura, K. Ueno, H. Sato, K. Fujisawa, and S. Matsuoka, "Performance characteristics of Graph500 on large-scale distributed environment," in *Proceedings of the International Symposium on Workload Characterization*, ser. IISWC '11, Nov 2011, pp. 149–158.
31. K. Fujisawa, "How to win Graph500," 2015, accessed: 2017-03-28. [Online]. Available: http://co-at-work.zib.de/files/20151002-Fujisawa_lecture.pdf
32. M. Ferdman, A. Adileh, Y. O. Koçberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," in *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2012, London, UK, March 3-7, 2012*, ser. ASPLOS '12, 2012, pp. 37–48. [Online]. Available: http://doi.acm.org/10.1145/2150976.2150982
33. K. Lim, P. Ranganathan, J. Chang, C. Patel, T. Mudge, and S. Reinhardt, "Understanding and Designing New Server Architectures for Emerging Warehouse-Computing Environments," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ser. ISCA '08, Washington, DC, USA, 2008, pp. 315–326. [Online]. Available: http://dx.doi.org/10.1109/ISCA.2008.37
34. J. Albericio, P. Ibez, V. Vials, and J. M. Llabera, "The reuse cache: Downsizing the shared last-level cache," in *46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2013, pp. 310–321.
35. N. A. Siddique, P. A. Grubel, A.-H. A. Badawy, and J. Cook, "A performance study of the time-varying cache behavior: a study on APEX, Mantevo, NAS, and PARSEC," *The Journal of Supercomputing*, vol. 74, no. 2, pp. 665–695, Feb 2018. [Online]. Available: https://doi.org/10.1007/s11227-017-2144-1
36. J. Huh, D. Burger, and S. W. Keckler, "Exploring the design space of future cmps," in *2001 International Conference on Parallel Architectures and Compilation Techniques (PACT 2001), 8-12 September 2001, Barcelona, Spain*, 2001, pp. 199–210. [Online]. Available: https://doi.org/10.1109/PACT.2001.953300
37. M. A. Laurenzano, A. Tiwari, A. Jundt, J. Peraza, W. A. W. Jr., R. L. Campbell, and L. Carrington, "Characterizing the performance-energy tradeoff of small ARM cores in HPC computation," in *Euro-Par 2014 Parallel Processing - 20th International Conference, Porto, Portugal, August 25-29, 2014. Proceedings*, 2014, pp. 124–137. [Online]. Available: https://doi.org/10.1007/978-3-319-09873-9_11
38. M. S. Warren, E. H. Weigle, and W.-C. Feng, "High-Density Computing: A 240-Processor Beowulf in One Cubic Meter," in *Proceedings of the ACM/IEEE Supercomputing Conference*, ser. SC '02, Nov 2002, pp. 61–61.
39. H. Nakashima, H. Nakamura, M. Sato, T. Boku, S. Matsuoka, D. Takahashi, and Y. Hotta, "Megaproto: 1 tflops/10kw rack is feasible even with only commodity technology," in *Proceedings of the ACM/IEEE Supercomputing Conference*, Nov 2005, pp. 28–28.
40. V. Vasudevan, D. Andersen, M. Kaminsky, L. Tan, J. Franklin, and I. Moraru, "Energy-efficient Cluster Computing with FAWN: Workloads and Implications," in *Proceedings of the 1st International Conference on Energy-Efficient Computing and*

Networking, ser. e-Energy '10. New York, NY, USA: ACM, 2010, pp. 195–204. [Online]. Available: http://doi.acm.org/10.1145/1791314.1791347

41. K. Fürlinger, C. Klausecker, and D. Kranzlmüller, "Towards Energy Efficient Parallel Computing on Consumer Electronic Devices," in *Proceedings of the First International Conference on Information and Communication on Technology for the Fight Against Global Warming*, ser. ICT-GLOW'11. Heidelberg: Springer-Verlag, 2011, pp. 1–9. [Online]. Available: http://dl.acm.org/citation.cfm?id=2035539.2035541
42. N. Rajovic, A. Rico, N. Puzovic, C. Adeniyi-Jones, and A. Ramírez, "Tibidabo: Making the case for an ARM-based HPC system," *Future Generation Comp. Syst.*, vol. 36, pp. 322–334, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.future.2013.07.013