Energy-Efficient Two-level Instruction Cache Design for an Ultra-Low-Power Multi-core Cluster

1st Chen Jie
DEI
University of Bologna
Grenoble, France
jie.chen@greenwaves-technologies.com

2nd Igor Loi
Research and Development
GreenWaves Technologies
Grenoble, France
igor.loi@greenwaves-technologies.com

3rd Luca Benini
DEI
University of Bologna
Bologna, Italy
luca.benini@unibo.it

4th Davide Rossi
DEI
University of Bologna
Bologna, Italy
davide.rossi@unibo.it

***Abstract*—High Energy efficiency and high performance are the key regiments for Internet of Things (IoT) edge devices. Exploiting cluster of multiple programmable processors has recently emerged as a suitable solution to address this challenge. However, one of the main power bottlenecks for multi-core architectures is the instruction cache memory. We propose a two-level structure based on Standard Cell Memories (SCMs) which combines a private instruction cache (L1) per-core and a low-latency (only one cycle latency) shared instruction cache (L1,5). We present a detailed comparison of performance and energy efficiency for different instruction cache architectures. Our system-level analysis shows that the proposed design improves upon both state-of-the art private and shared cache architectures and balances well performance with energy-efficacy. On average, when executing a set of real-life IoT applications, our multi-level cache improves performance and energy efficiency both by 10% with respect to the private instruction cache system, and improves energy efficiency by 15% and 7% with a performance loss of only 2% with respect to the shared instruction cache. Besides, relaxed timing makes two-level instruction cache an attractive choice for aggressive implementation, with more slack for convergence in physical design.**

***Index Terms*—instruction cache, parallel architecture, energy efficiency, relaxed-timing.**

I. Introduction

With the growing demand for edge computing devices in IoT market, such as drones, smart watches, and wearable devices, it is becoming urgent to improve the capabilities of edge devices towards higher performance, to deal with complex near sensor analytics algorithms, and low power to extend device working longevity. As such, heterogeneous computing architectures have been proposed to accelerate specified algorithms within power envelopes of only a few mW or even less [1], [2]. On the other hand, (Symmetric MultiProcessing) SMP parallel architectures have been proposed recently to achieve high performance and high energy-efficiency exploiting parallelism coupled to low-voltage operation [3], in a fully programmable environment. In this scenario, optimizing the instruction memory hierarchy of tightly coupled clusters of processors is one of the most challenging research topics towards flexible and energy efficient computing platforms.

A common method to reduce energy consumption of the instruction cache hierarchy in ultra-low-voltage systems is to use advanced memories, such as 8T or 10T SRAMs optimized for sub-threshold operation. Other approaches exploit Standard Cell Memories (SCMs) [4] [5], which are more flexible and easily portable from one technology to another. SCMs present extremely interesting features for small memory size, low-voltage and energy-efficient designs, since: i) they can operate with very low voltage, even lower than 10T SRAMs optimized for low voltage [6] ii) their energy per access is significantly smaller than SRAMs at the same voltage supply [5]. Nevertheless, although controlled placement of standard cells memory array reduces area overhead [5], it is still huge as 2x to 4x the area of the same size SRAMs based memory. This makes the usage of SCM-based private instruction caches in tiny IoT end nodes only feasible for relatively small sizes, significantly degrading performance for applications with large memory footprint or for those applications relying on software libraries frequently polluting the instruction cache [3].

Explored approaches to take full benefit from the energy efficiency of SCM-based instruction memories in multi-core systems leads to the exploitation of shared instruction caches [7]. Compared to traditional private instruction cache, sharing the cache among multiple cores allows to provide the processors with a larger "virtual" cache capacity, reducing the overheads of area expensive latches. Indeed, during the execution of parallel code, a large fraction of the cached instructions is replicated in every private cache, leading to inefficient usage of program cache capacity. This situation is exacerbated if we consider a parallel program that relies on a parallel programming model such as OpenMP [8]. Moreover, instruction caches are well suited for sharing: their read-only nature inherently limits the complexity of hardware implementation and does not require to maintain coherency. As such, Loi et al. [9], proposed two different instruction cache architectures, one leveraging a crossbar between processors and single-port cache banks, and another leveraging banks with multiple read ports, publicly available on github[1].

Nevertheless, the first level cache is often on the critical path of simple single-issue microprocessors featuring a flat pipeline [10]. Moreover, DSP-oriented processors designed for high energy efficiency typically feature a prefetch buffer to support compressed instructions and reduce the pressure on the memory hierarchy (low power) and even more complex instruction fetch stages to avoid stalls (high efficiency). However, this extends the critical path toward the instruction cache [11]. Finally, this path is further extended in shared caches, due to the need of an additional crossbar or multiple ports [7] towards the cache banks. These considerations, joint with the well know routing bottlenecks for deep submicron technology nodes, potentially pose significant limitations on the performance, energy efficiency, and scalability of future software-programmable parallel processor clusters.

In this context, the contributions of the paper are the following:

- We propose a two-level instruction cache combining a private instruction cache (L1) with a single cycle latency shared instruction cache (L1,5) to relax the timing and balance the performance with the energy efficiency in tightly coupled clusters of parallel processors.

[1]https://github.com/pulp-platform

978-3-9819263-4-7/DATE20/©2020 EDAA

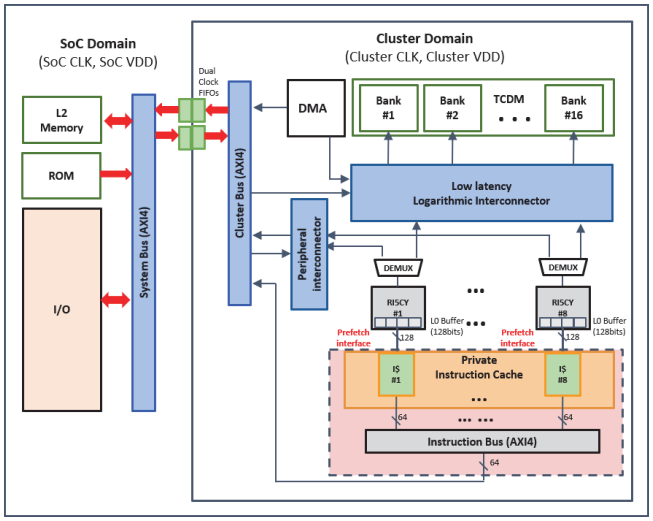


Fig. 1: Baseline SoC architecture, featuring a cluster with private instruction cache

- We synthesized and implemented (place and route) an 8-core cluster integrating the proposed instruction cache architectures in commercial 22nm FD-SOI technology, and we performed an extensive and comprehensive exploration, including a comparison with state-of-the-art instruction cache architectures based on physical implementation results (i.e. area, power), and performance and efficiency of synthetic benchmarks.
- We implemented an extensive set of real-life applications on the evaluated cache architectures analyzing performance and energy efficiency of the proposed solutions on typical signal processing and CNN IoT applications.

Results show that, for an energy efficient implementation of the cluster (i.e. timing relaxed by ∼30% with respect to the maximum operating frequency), the proposed two-level cache architecture improves the performance with respect to the private cache by 10% both in throughput and in energy efficiency, on average. When compared with shared cache systems, for low miss rate applications (real-life IoT application), it features 15% and 10% better energy efficiency than the multi-port and single-port respectively, while loses only up to 2% of throughput due to its hierarchical structure. In applications with high I-cache miss rate, not common in IoT applications, the new cache organization still has much better performance than private caches of same total cache capacity, thanks to larger shared L1,5 and has an acceptable performance loss with respect to the flat shared caches. Finally, the proposed cache provides strong advantages in implementation effort with respect to the flat shared caches, improving the scalability of multi-core systems both in terms of number of cores per cluster and maximum operating frequency.

II. Background

A. Global Architecture

The baseline system we consider in this work is a tightly coupled cluster including 8 32-bit RISC-V cores [11]. The RISC-V cores are based on an in-order, single-issue, four stage pipeline micro-architecture without branch prediction, improved with extensions such as hardware loops, load/store with pre/post increment, SIMD operations for higher throughput and energy efficiency in parallel signal processing workloads [11]. No data cache is present for avoiding memory coherency and large power overhead [12]. Each core shares a 128 KB ultra-low-latency L1 multi-banked Tightly Coupled Data Memory (TCDM) and a DMA tightly coupled to the TCDM [13].

Both of the instruction cache and DMA are connected to an AXI4 cluster bus for fetching off-cluster data. Besides, to increase the throughput of cores, a prefetch buffer of one 16-byte cache line is adopted between the core and instruction cache. Since SoC domain and cluster domains have its own Frequency-Locked Loops (FLLs) and supply voltages, it is possible to adjust the frequency and supply voltage according to use case. Finally, thanks to the low-voltage SRAMs for L2 memory of SoC domain and cluster TCDM, and to the near-threshold 22nm FDX technology, the supply voltage of the SoC and cluster can scale down to near-threshold 0.65V to improve its energy efficiency.

B. Explored I$ Architectures

Based on previous work, three type of instruction caches are analyzed, described in the following. All the presented solutions are implemented with latch-based SCMs for improved energy efficiency. More details about the presented instruction cache architectures can be found in this work [9].

1) Private cache: The baseline cluster features private instruction caches (Fig. 1). Each private cache bank is composed of 3 elements, TAG and DATA array which are implemented using SCMs and cache controller which uses request-grant handshake protocol with a pseudo-random (PRAND) replacement policy. Private caches are fast (i.e. small critical path) and simple (i.e. low-power). Both data replication and high miss penalty for large footprint applications are major drawbacks for private instruction cache which decrease their performance and energy efficiency.

2) Single-port Shared Cache: Shared instruction cache benefits from large cache capacity to avoid data replication while needs to minimize the access cycle (single-cycle-latency) and area overhead. The single-ported shared instruction cache features a read-only low-latency crossbar which uses round-robin arbitration policy for each core's fetch request [9]. Since one cache bank can serve one refill request each time, it causes congestion when several cores access to the same cache bank for parallel applications. Moreover, a long path is present between the instruction fetch stage of the processors to the cache banks through the interconnect.

3) Multiple-port Shared Cache: Another method minimizes the access cycle to first level cache while takes the advantage of large cache capacity is to use multiple ports memory banks, TAG and DATA memories are shared, while keeping the cache controllers private for each core and close to the core fetch interfaces. Nevertheless, even it avoids the heavy congestion for cores' access, 8-ported TAG and DATA memories have a serious area overhead issue, as a result it is suitable only for cache sizes up to a few KB.

III. Two-Level Instruction Cache

This section describes the proposed two-level instruction cache which combines small private I$ (level 1) with a tightly coupled (1 clock cycles of latency) shared I$ (Level 1,5). The two caches are connected together through a single clock latency interconnect [14]. This approach avoids long critical paths from the core to the interconnect and back to the core

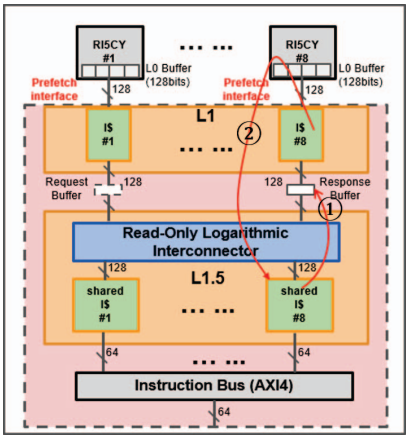


Fig. 2: Two-level Instruction cache which combines private I$ (L1) with shared I$ (L1,5) with critical path which is solved by adding response buffer

(Fig. 2 while benefits from the low-latency access time from the core to the private I$, and from the private I$ to the L1.5$.

When a refill is done, the instruction is fetched by the Instruction Fetch Unit (IFU) of the cores crossing the low-latency interconnect. Since a prefetch buffer is present on the core, the current fetched instruction is being decoded to fetch the following instruction. Then, a request is sent from core to shared cache crossing the low-latency interconnect. If two or more cores compete to fetch instructions stored on the same cache bank, a round-robin arbitration policy ensures that only one core can access the cache bank while the others are stalled.

As shown in Fig. 2, the proposed hierarchical cache features a request buffer and a response buffer featuring design configurable pipeline stages (i.e. they can be enabled with a System Verilog parameter) to eliminate this critical path through the I$. Since in the presented cluster implementation the response buffer is sufficient to cut the critical path, the request buffer has been disabled. On the other hand, this buffer is a powerful knob to improve the scalability of the system towards high-end clusters optimized for frequency or featuring a larger number of cores (e.g. 16), taking advantage of the hierarchical structure of the cache.

In the proposed implementation, the access time of L1 I$ is 1 cycle, of L1,5 I$ is 2 cycle when cache hit separately. Going from the L1 to the L1,5 is one cycle of latency in case of no conflicts, and more cycles in case of conflicts. However, the contention on the memory banks is largely decreased by adopting a banking factor of 2 and L1 filter cache. For L1,5, shared cache banks have been designed to support multiple outstanding transactions and non-blocking behavior in case of miss. If a miss happens, the cache bank asserts the respective refill request, and then, while waiting for the refill response from L2, processes incoming fetches for the other cores. The shared cache controller is also able to track more than one pending refill.

IV. Implementation and Evaluation

A. Experimental Setup

To perform a detail analysis of the trade-offs between the discussed cache architectures in terms of performance, energy and area, we consider a single cluster with 8 processors, and a configurable size of the instruction cache (4kB to 8kB) which

Mnemonic	Type	Refill Cycles	Description
4K-PR	Private	15	512B I$ bank per core
8K-PR	Private	15	1024B I$ bank per core
4K-SP	Shared	17	8x 512B I$ banks, single-port
8K-SP	Shared	17	8x 1024B I$ banks, single-port
4K-MP	Shared	19	2x 2048B I$ banks, 8-port
8K-MP	Shared	19	2x 4096B I$ banks, 8-port
4K-HIER	Private	19	512B I$ bank per core
	Shared		2x 2048B I$ banks, single-port
8K-HIER	Private	19	512B I$ bank per core
	Shared		2x 4096B I$ banks, single-port

TABLE I: Architectural configurations details

are reasonable for the kind of IoT applications we target and suitable area overheads, while the size of the TCDM is fixed to 128 Kb. Table I shows all the architectures used in the proposed experiments, where a 4-way set associative configuration with a cache line width of 16 bytes (4 words) is used, providing a well balanced trade-off between performance and area overhead for low-power multicores [9]. To obtain physical characteristics (i.e. power, area) of the design was synthesized and implemented with GF 22nm FD-SOI technology library. We use Cadence Genus-19.10 to synthesize the design and Cadence Innovus-18.10 for the place and route. To characterize the power consumption of the proposed instruction cache architectures, we assume that the both of SoC domain and the cluster domain operate at the supply voltage of 0.65V.

To obtain the average power consumption of the cluster we simulated its post place-and-route netlist when running a series of synthetic tests with specific process, voltage and temperature (PVT) corner. In this exploration, we run all the tests at the operating frequency of 200MHz and supply voltage of 0.65V for cluster with 25°C. The chosen frequency is 30% smaller than the maximum frequency, since the target is maximum energy efficiency instead of max performance. Then, for each test, we back-annotated the switching activity traces in VCD (Value Change Dump) format with questasim 10.7b only in third execution to avoid cold boot, and we passed it to Synopsys PrimePower M-2016.12 together with the parasitic file (in SPEF format) of certain RC corner to achieve an accurate cluster power estimation. Nevertheless, cluster power only is not enough for power estimation, we need to consider also the L2 AXI refill power with L2 SRAM access power due the characteristics of different instruction caches. Finally, we use this accurate power estimation to do a detail comparison.

Nevertheless, it is not feasible to run power analysis on large benchmarks, so we use an FPGA for running the benchmarks since it is cycle accurate emulation. Then, we need to find a method to evaluation the power with the FPGA running result. Here, we measured the relationship between the power and the miss rate. If one application has high miss rate it means that the cluster can stall for several cycles. As the result, during the refill period, the cores are stalled so the average power of cluster decreases (Fig. 4b). Thus, based on the assumption that the higher the miss rate, the lower the average power of cluster. Finally, we have $P_{Average} \propto (1/MissRate)$. After the collection of all the average power for synthetic tests, we created a look up table (LUT) model describing the relationship between miss rate and average power for each architecture. Finally, we use this LUT to evaluate in depth with the real-life IoT applications.

The synthetic test is a simple parallel matrix multiply test for 8192 words. We use Loop unrolling technique to force

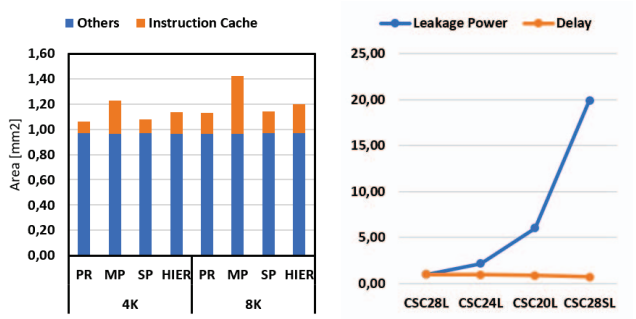


Fig. 3: Left : Cluster Area breakdown for the different cache architecture and configurations. Instruction cache contribution is placed on top. Right : Comparison of cells' leakage power and latency normalized to CSC28L

the size of instructions within the main loop to be 0.375KB, 0.75KB, 1.5KB, 3KB, 6KB and 12KB. Finally, we have 6 tests for each architecture and all the tests have the same number of operations for cluster. In the end, we can have a LUT of relationship between miss rate and average power for each architecture configuration.

B. Physical Implementation Results

To characterize the performance of the various instruction cache architectures, it is necessary to perform a wide physical exploration in terms of timing, area, power for all the configurations shown in Table I. In the following, we will refer to the private cache, multi-ported, single-ported shared cache and the proposed two-level cache designs with the following acronyms: PR, MP, SP, HIER.

Fig. 3 left illustrates the silicon area costs for all cache architecture configurations, for each same instruction cache size, private cache is smaller than all other caches thanks to its simpler architecture, two-level cache is bigger than SP since it always has more 4KB (8x512B) L1 memory than SP and MP has the largest area because of large number of read ports and lot of routing congestion. Besides, to evaluate the timing pressure of design, we use the table II which lists the percentage of different standard library cells of GF 22nm used in the design. The tool adds more leaky cells (CSC28SL GF standard supper low Vt cells, see Fig. 3 right) if the design has bigger timing pressure. As we can see, the private cache has a small timing pressure but as we will see later that it is much less efficient in terms of performance. The MP has less timing pressure while the SP suffers from big timing pressure. The HIER has the least usage of CSC28SL which means HIER has the best timing in all of the I$ architectures and can ease the timing convergence for physical implementation (place and route).

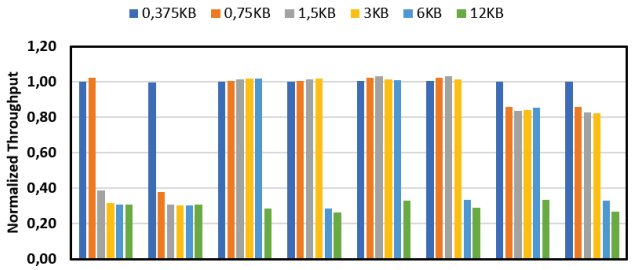
C. Evaluation

This section evaluates the the most significant parameters, performance, power and energy of the presented instruction cache architectures for low-cost, low-power processor clusters, which gives us a global view of the characteristic for each cache architecture. All the following figures show the normalized results over the 8-core 8K-PR configuration for test of 0,375KB instruction size which we consider as a baseline.

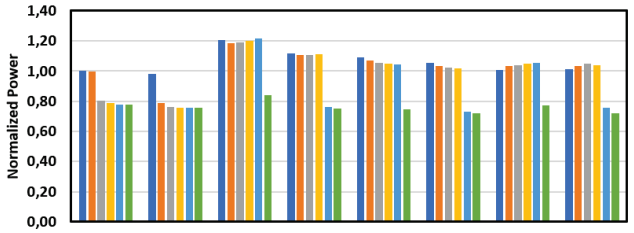
Fig. 4a summarizes the throughput (execution efficiency) of all synthetic tests measured on the different architectural

Size	Type	CSC28L [%]	CSC24L [%]	CSC20L [%]	CSC28SL [%]
8K	PR	59,8	4,3	8,7	27,2
	MP	62,9	4,0	6,1	27,1
	SP	58,7	4,6	8,4	28,4
	HIER	62,1	4,1	8,3	25,6
4K	PR	57,3	4,5	9,1	29,1
	MP	60,1	3,9	7,3	28,6
	SP	56,7	4,9	9,0	29,4
	HIER	59,5	4,2	9,0	27,3

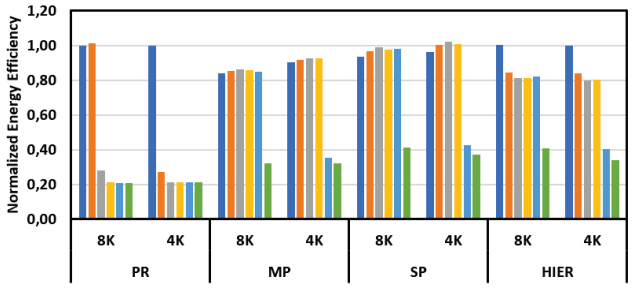
TABLE II: Percentage of different standard library cells of GF 22nm used in different I$ configuration



(a) Throughput of tests normalized to 8K-PR with 0,375KB cache size



(b) Cluster power of tests normalized to 8K-PR with 0,375KB cache size



(c) Energy efficiency of tests normalized to 8K-PR with 0,375KB cache size

Fig. 4: Scalability analysis of the cache architectures increasing the instruction size. (b) shows only the normalized cluster domain power for all I$. (c) shows the inverse of energy which includes the L2 AXI refill energy, L2 SRAM access energy and the cluster energy

configurations. There are three points that we can see. First, the performance drops significantly when over the limit of total cache capacities, we can see for private cache of 4KB (8 x 0,5KB) and 8KB (8 x 1,0KB) total cache capacity, the performance drops dramatically for 0,75KB and 1,5KB synthetic test respectively. It is the same for shared and two-level cache. Second, if we consider the same total cache

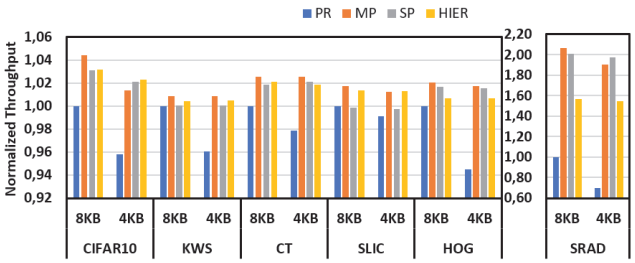
APP	Size	Class	Description
CIFAR10	7.1KB	CNN	Object Recognition
KWS	3.5KB	CNN	Key word spotting
CT	2.9KB	Long-Jump	Color Tracking
SLIC	26.1KB	Long-Jump	Simple Linear Iterative Clustering for image segmentation
HOG	31.1KB	Library	Histogram of Oriented Gradients
SRAD	31.7KB	Library	Speckle Reducing Anisotropic Diffusion

TABLE III: Benchmark details

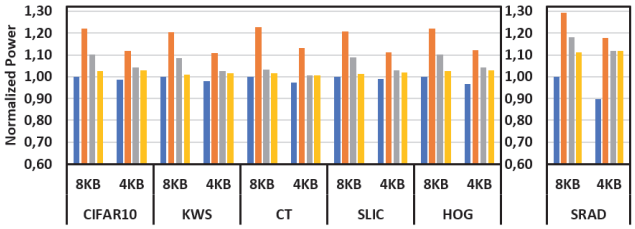
capacity, two-level and shared cache architectures always show better or equal performance when compared to the related private configurations. We should notice that that shared caches always have larger L1 cache capacities (8x) than private cache. In the end, shared cache architectures always have better performance compared with two-level cache because two-level cache has smaller L1 cache capacity and there are about two more refill cycles when there is a miss. So when the instruction size is between the L1 cache capacity (total cache capacity / 8) and L1.5 cache capacity (total cache capacity), it loses about 16% throughput while the miss rate is higher than 50%.

Fig. 4b shows the power of all synthetic tests measured on the different architectural configurations. As we can see, MP always has the largest power while PR has the least power because of the design area. For SP and HIER it varies a little, when the instruction size is between the L1 cache capacity (total cache capacity / 8) and L1.5 cache capacity (total cache capacity), we can see the power of SP is smaller than HIER, otherwise the power of SP is bigger than HIER.

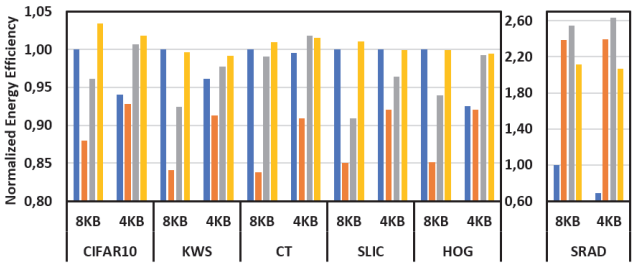
Fig. 4c shows the energy efficiency of all synthetic tests measured on the different architectural configurations. Here the energy includes the L2 AXI refill energy, L1 SRAM access energy and cluster energy to obtain an accurate evaluation. There are two points need to notice. First, when the instruction size is below the PR cache capacity, PR always has the best energy efficiency. Otherwise, two-level and shared caches always have better energy efficiency than PR and SP has the best energy efficiency. Second, when compare HIER with shared cache, only when the instruction size is smaller than the L1 cache capacity (total cache capacity / 8), HIER has better energy efficiency than shared cache architectures.



(a) Throughput of applications normalized to the 8K-PR



(b) Cluster power of applications normalized to 8K-PR



(c) Energy efficiency of applications normalized to the 8K-PR

Fig. 5: Scalability analysis of the cache architectures with real-life applications. (b) shows only the normalized cluster domain power for all I$. (c) shows the inverse of energy which includes the L2 AXI refill energy, L2 SRAM access energy and the cluster energy

V. Benchmarking

A. Description of benchmarks

To analyze in depth the behavior of each architecture, we used four benchmarks based on full-fledged optimized OpenMP implementation [8] applications and two CNN applications which use a CNN library to manage memory transfer between L1 and L2 data memory automatically for embedded system [15], each featuring a different behavior in terms of access patterns to the instruction memory subsystem, as well as diversified memory footprint and execution time.

Table III shows in detail the characteristics of each application, including each code section size in Kb. Since cache performance is influenced strongly by code locality and code size, we classify the applications into 3 groups, CNN class or Short-Jump class includes all CNN kernel based applications, which are loop-based applications with most of loop bodies smaller than four lines of cache. The second class is Long-Jump, groups all the loop-based applications with loop bodies greater than four lines of cache, or based on extensive use of control flow instructions, including CT and SLIC. In the end, the Library class includes HOG and SRAD which use libraries to manage non-native data types, such as float, and fixed point arithmetic, generating a big stress in caches [11].

All above applications are fairly complex and long-lived (millions of instructions in most cases), which takes too much time to do the exploration with RTL post place-and-route simulation. For this reason, we analyse the performance based on measures coming from RTL-equivalent FPGA implementations, mapped on Xilinx Zynq ZCU102 FPGA using Vivado 2018.2 only in 8-core situation. The FPGA emulation allows to execute at up to 50 MHz, enabling near-to-real-life execution time. The performance analysis is based on statistics collected by hardware counters implemented inside the instruction cache. Then we calculate the miss rate of each application and use the miss rate to power LUT to estimate the final cluster power for each I$ architecture. Then, we combine the cluster power with L2 AXI refill and L2 SRAM access power to obtain the final energy efficiency for each I$ architecture.

B. Evaluation

Fig. 5 shows all the results of execution efficiency, power and energy efficiency, normalized to the 8K-PR configuration results of each application. If we consider pure architectural performance (execution efficiency), the flat shared cache configurations perform better than both the two-level and the private caches, joining the benefits of a larger cache capacity with respect to the private, and a smaller L2 refill latency with respect to the two-level (see table I) which causes a drop in performance of only 2%, on average. However, this comes with a significant cost in terms of average power consumption. Indeed, looking at Fig. 5b, it is possible to note that, on average, both the SP- and the MP-based clusters consume significant more power (up to 20%) than the two-level. This is due to the high timing pressure on the paths between the instruction fetch stage of the cores and the cache, which increases the power consumption of both the cache and the cores. This is true both for the SP, due to the interconnection in between the cores and the cache, and the MP, featuring large area overhead due to the multiple ports which leads to significant routing congestion, logic spreading and significant buffering.

Joining the architectural efficiency and power considerations we can conclude that for most applications, featuring a miss rate smaller than 5%, very common in the near-sensor processing domain, the energy efficiency of the proposed two-level cache surpasses both the private and the shared caches, when considering most of the capacity configurations analyzed. We should note that a significant energy saving for the shared cache configurations (both flat and hierarchical) comes from the merged refill requests to the energy expansive L2 memory. If we compare the 8K-PR configuration with the 4K-SP configuration, we can find that for a similar area occupation (Fig. 3), the two caches deliver a similar performance and energy efficiency. However, the two-level cache provides significantly more robustness with respect to applications with large footprint and long branches. This can be clearly noted in SRAD, where the shared caches deliver more than 2x better energy efficiency than the private configurations. For this application, the two-level cache provides slightly smaller performance (20%) than the flat shared caches, due to the large number of L1,5 refills, which becomes more significant for high miss rate applications such as SRAD.

However, this is largely compensated by the significant benefits in terms of implementation effort (i.e. smaller routing congestion and smaller utilization of low-Vt cells for physical implementation), especially when considering advanced technology nodes featuring the well known routing bottleneck. Moreover, the flexibility given by the possibility to configure the cache with up to 2 pipeline stages, leads to a better scalability both in terms of frequency and number of cores per cluster with respect to flat shared caches, paving the way to powerful and energy efficient software programmable clusters for next generation IoT end-nodes in advanced silicon technologies.

VI. Conclusion

In this work, we proposed a hierarchical instruction cache architecture for energy efficient and cost effective tightly coupled clusters of processors. Exploiting a small level-zero private cache tightly coupled to a single-clock latency L1.5 shared cache, the proposed architecture joins the benefits of private caches (low-power consumption) with large capacity typical of shared caches (execution efficiency). We benchmark the proposed architecture on a wide range of real-life IoT workloads, showing that for low miss rate applications the proposed architecture improves the energy efficiency by 10% with respect to private caches, and from 7% to 15% with respect to flat shared caches. In applications with high I-cache miss rate, not common in IoT applications, the new cache organization still has much better performance than private caches of same total cache capacity, thanks to larger shared L1,5 and has an acceptable performance loss with respect to the flat shared caches. As a conclusion, the proposed cache provides strong advantages in implementation effort with respect to the flat shared caches, improving the scalability of multi-core systems both in terms of number of cores per cluster and maximum operating frequency. Finally, more effect will be paid to solve the issue of high miss rate applications, such as adding prefetching feature between L1 and L1,5.

References

[1] U. Banerjee, C. Juvekar, A. Wright, Arvind, and A. P. Chandrakasan, "An energy-efficient reconfigurable dtls cryptographic engine for end-to-end security in iot applications," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb 2018, pp. 42–44.

[2] F. Conti, A. Marongiu, and L. Benini, "Synthesis-friendly techniques for tightly-coupled integration of hardware accelerators into shared-memory multi-core clusters," in *Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 5:1–5:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2555692.2555697

[3] D. Rossi, A. Pullini, I. Loi, M. Gautschi, F. K. Grkaynak, A. Teman, J. Constantin, A. Burg, I. Miro-Panades, E. Beign, F. Clermidy, P. Flatresse, and L. Benini, "Energy-efficient near-threshold parallel computing: The pulpv2 cluster," *IEEE Micro*, vol. 37, no. 5, pp. 20–31, Sep. 2017.

[4] P. Meinerzhagen, S. M. Y. Sherazi, A. Burg, and J. N. Rodrigues, "Benchmarking of standard-cell based memories in the sub-$v_t$domain in 65-nm cmos technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 2, pp. 173–182, June 2011.

[5] A. Teman, D. Rossi, P. Meinerzhagen, L. Benini, and A. Burg, "Power, area, and performance optimization of standard cell memory arrays through controlled placement," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 21, no. 4, pp. 59:1–59:25, May 2016. [Online]. Available: http://doi.acm.org/10.1145/2890498

[6] P. Andreas Meinerzhagen, S. M. Y. Sherazi, A. Burg, and J. Rodrigues, "Benchmarking of standard-cell based memories in the sub-v_{rmT} domain in 65-nm cmos technology," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 1, pp. 173 – 182, 07 2011.

[7] I. Loi, D. Rossi, G. Haugou, M. Gautschi, and L. Benini, "Exploring multi-banked shared-l1 program cache on ultra-low power, tightly coupled processor clusters," in *Proceedings of the 12th ACM International Conference on Computing Frontiers*, ser. CF '15. New York, NY, USA: ACM, 2015, pp. 64:1–64:8. [Online]. Available: http://doi.acm.org/10.1145/2742854.2747288

[8] A. Marongiu, A. Capotondi, G. Tagliavini, and L. Benini, "Simplifying many-core-based heterogeneous soc programming with offload directives," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 4, pp. 957–967, Aug 2015.

[9] I. Loi, A. Capotondi, D. Rossi, A. Marongiu, and L. Benini, "The quest for energy-efficient i$ design in ultra-low-power clustered many-cores," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 2, pp. 99–112, April 2018.

[10] D. A. Patterson and J. L. Hennessy, "Large and fast: Exploiting memory hierarchy," in *Computer Organization & Design The Hardware/Software Interface : Fifth Edition*, 2014.

[11] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Grkaynak, and L. Benini, "Near-threshold risc-v core with dsp extensions for scalable iot endpoint devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700–2713, Oct 2017.

[12] L. Benini, E. Flamand, D. Fuin, and D. Melpignano, "P2012: Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2012, pp. 983–987.

[13] D. Rossi, I. Loi, G. Haugou, and L. Benini, "Ultra-low-latency lightweight dma for tightly coupled multi-core clusters," 05 2014.

[14] A. Rahimi, I. Loi, M. R. Kakoee, and L. Benini, "A fully-synthesizable single-cycle interconnection network for shared-l1 processor clusters," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[15] GreenWaves Technologies, "Gap8 auto-tiler manual," 2018, https://greenwaves-technologies.com.