Task Scheduling Sensitivity to L1 Cache settings on an area-constrained 32-core RISC-V Processor

Lucas Morais*†, Daniel Jiménez-González*†, Carlos Álvarez*†
*Barcelona Supercomputing Center, Barcelona, Spain
†Universitat Politècnica de Catalunya, Barcelona, Spain
E-mail: {lucas.morais, daniel.jimenez, carlos.alvarez}@bsc.es

Keywords—RISC-V, FPGA, Rocket Chip, Task Scheduling, Cache, Design Exploration.

I. Extended Abstract

High-performance applications are highly sensitive to memory performance characteristics. While programs with comparatively low memory-to-computation ratio are less likely to be hampered by limited memory bandwidth, most parallel applications will be severely impacted by the absence of hardware support for low-latency inter-thread synchronization and data sharing.

In this paper, we report a design exploration that sought to identify the cache configuration that maximizes performance of task parallel OpenMP workloads running on a Linux-capable 32-core RISC-V system. We show that, under the constraints of a U200 Alveo FPGA, the best single-level cache configuration consists in 160 KB of coherent, core-private data caches, with a 32/128 split among instruction and program data. With such configuration, we have achieved speedups of up to 28x and 19x for the nbody and cholesky applications, respectively.

A. Background

Rocket Chip [1] is a popular open-source project allowing rapid prototyping of RISC-V systems. While systematic design explorations [2] and tools to that purpose [3] have been proposed in the past, no work had so far focused on optimizing cache configuration for Task Scheduling applications.

Such optimal cache parameters should provide performance gains complementary to those provided by low-latency hardware-accelerated data dependence resolution [4], [5].

B. Rocket Chip's cache parametrization features

Rocket Chip offers a rich set of parameters that might be used to control the organization and characteristics of many of its micro-architectural elements. For example, parameters exist that control, on a core-specific manner, whether a FPU should be integrated or how many miss status handler registers should be included. More to the point of our paper, there are options to control the number of cache sets and ways, as well as the cache line size. It is also possible to select a random, Pseudo-LRU, or a True LRU cache replacement policy.

In our different experiments, we modulated cache capacity by accordingly varying the number of ways of each cache element, as this proved to require much less FPGA resources than changing the number of sets, for example, since other options render the design much less amenable to be mapped to efficient FPGA storage elements.

	LUT	**LUTRAM**	**FF**	**BRAM**
Alveo U-200	1.18M	592K	2364K	2.16K
I-16, D-128	1.07M (90%)	32.6K (6%)	574K (24%)	1.91K (89%)
I-32, D-64	1.04M (88%)	31.9K (5%)	568K (24%)	1.32K (61%)
I-64, D-64	1.08M (91%)	32.6K (6%)	596K (25%)	1.67K (77%)
I-32, D-128	1.08M (92%)	31.9K (5%)	584K (25%)	2.09K (97%)

TABLE I: Resource usage of the different configurations.

Table I shows the resource usage of the different configurations used in the performed study. Each implemented core is an in-order Rocket core, single issue with a floating point ALU. The system runs at 60MHz in FPGA and is Linux-capable.

C. Benchmarks

We evaluate different cache configurations using four benchmarks: Cholesky, N-Body, ramspeed-reading, and ramspeed-writing. Cholesky and N-Body are extensively-used HPC kernels implemented using the Task Parallel paradigm, while the latter two are variants of a Linux tool for evaluating memory performance according to different access patterns.

More specifically, Cholesky is a kernel implementing a matrix decomposition method that is employed, for example, in linear equation solving. In turn, N-Body encapsulates programming logic that is typically found in applications where, given a system of particles, interactions among every pair of particles are evaluated at each simulation step.

Both Cholesky and N-Body make use of OpenMP's automatic data-dependence resolution. However the memory requirements for N-Body are usually lower than for Cholesky. N-Body computation grows with the square of the number of elements. In Cholesky computation grows with the cube of the problem size but data grows square with the problem size usually putting more pressure in the memory system. Both applications have been executed with the optimum task granularity for the system evaluated.

In the case of ramspeed, it evaluates the performance of a series of readings and writings in memory, giving an idea of the scalability of the memory subsytem.

D. Preliminary Results

Figure 1 shows the speedup results of the different benchmarks when executed in a different number of cores against the same application executed sequentially in the same system. Each subfigure shows the performance obtained in each of the different cache configurations analyzed. In addition, Table II shows the absolute performance obtained in each execution.

Our experiments show that the (I-16, D-128) performs poorly in spite of its large data-cache component, which

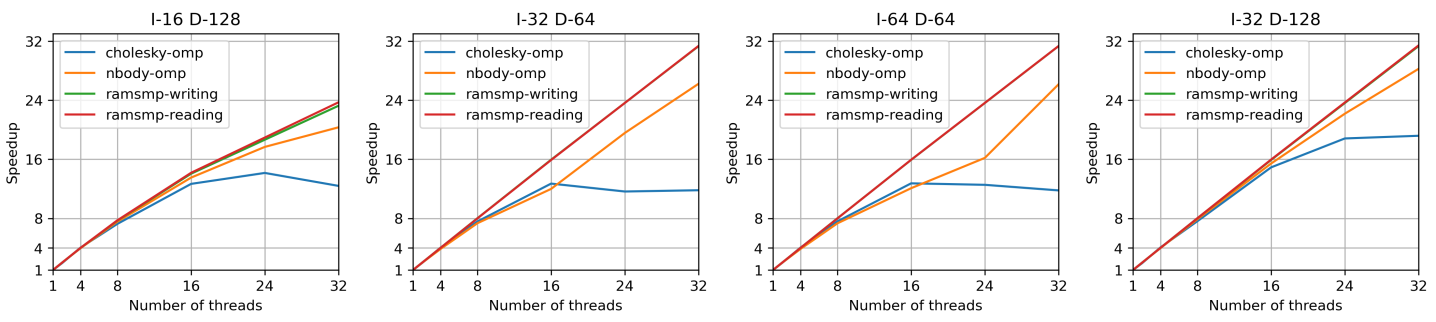

Fig. 1: Performance results across different cache designs.

	1	2	4	8	16	24	32
I-16 D-128							
cholesky	31.51	61.24	127.31	228.67	399.22	445.92	390.62
nbody	0.43	0.86	1.69	3.22	5.77	7.54	8.67
ram-reading	0.44	0.88	1.75	3.37	6.14	-	10.16
ram-writing	0.22	0.45	0.90	1.74	3.18	-	5.32
I-32 D-64							
cholesky	30.22	59.43	121.75	228.90	384.01	351.68	356.87
nbody	0.43	0.85	1.66	3.16	5.16	8.42	11.28
ram-reading	0.45	0.90	1.81	3.62	7.20	-	14.14
ram-writing	0.23	0.46	0.93	1.86	3.69	-	7.28
I-64 D-64							
cholesky	30.14	59.36	121.86	228.80	384.05	378.02	355.27
nbody	0.43	0.85	1.66	3.15	5.20	6.96	11.25
ram-reading	0.45	0.90	1.81	3.62	7.20	-	14.16
ram-writing	0.23	0.46	0.93	1.85	3.70	-	7.27
I-32 D-128							
cholesky	32.76	63.49	133.30	249.62	488.36	616.61	628.48
nbody	0.44	0.88	1.76	3.48	6.81	9.79	12.47
ram-reading	0.45	0.91	1.81	3.62	7.19	-	14.15
ram-writing	0.23	0.46	0.93	1.86	3.70	-	7.27
	Number of threads						

TABLE II: Performance results. Executions of ramsmp are evaluated in gigabytes per second, while cholesky and nbody performances are measured in megaflops and mega-pairs per second, respectively. Ramsmp only supports power-of-two numbers of threads.

suggests that 16 KB is not enough for ensuring scalability for a system with that much data cache. Moreover, the single-threaded performance of such configuration is substantially inferior to that of all other arrangements.

We also notice that the (I-32, D-64) and (I-64, D-64) designs obtain nearly the same performance for all configurations involving a number of threads different from 24, although the latter features double the data cache of the former. This is evidence that instruction-fetching overhead and related problems are not likely to be reduced by increasing instruction caches beyond 32 KB, at least for the tested workloads. On the other hand, the differences at 24 threads suggest that increasing the instruction cache beyond 32 KB can have both positive or negative effects on load balancing depending on the workload.

E. Conclusions

This work reports how Rocket Chip cache parameters might be selected to obtain maximum performance out of selected Task Parallel applications under the resource constraints of the U200 Alveo FPGA. We show that, among the considered configurations, only the one featuring 160 KB of L1 cache split in 32/128 instruction/data ratio can ensure scalability beyond 16 threads for a standard task-parallel cholesky implementation. We also indicate that, while serving 32 threads, the (32-I, 128-D) configuration improves performance by up to 54% with respect to (16-I, 128-D) or 76% with respect to (64-I, 64-D), while requiring just 11% or 25% more storage than each of these options, respectively.

As future work, we propose to evaluate how Task Parallel applications might benefit from L2 caches mapped to URAM resources. Also, we would like to evaluate whether our scalability conclusions also hold for runtimes other than OpenMP.

F. Acknowledgement

This work is supported by the TEXTAROSSA project G.A. n.956831, as part of the EuroHPC initiative from Spanish Government (PID2019-107255GB-C21/AEI /10.13039/501100011033), and from Generalitat de Catalunya (contracts 2017-SGR-1414 and 2017-SGR-1328).

References

[1] K. Asanovic *et al.*, “The rocket chip generator,” *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, vol. 4, 2016.

[2] M. Doblas Font *et al.*, “Microarchitectural design-space exploration of an in-order risc-v processor in a 22nm cmos technology,” B.S. thesis, Universitat Politècnica de Catalunya, 2020.

[3] S. Bandara *et al.*, “Brisc-v: An open-source architecture design space exploration toolbox,” *arXiv preprint arXiv:1908.09992*, 2019.

[4] L. Morais *et al.*, “Adding tightly-integrated task scheduling acceleration to a risc-v multi-core processor,” in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 861–872.

[5] J. M. de Haro *et al.*, “Towards reconfigurable accelerators in hpc: Designing a multipurpose efpga tile for heterogeneous socs,” in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022.

Lucas Morais has a Master Degree in Computer Science from the University of São Paulo, where he worked with an international team for adding native Task Parallelism support to RISC-V based multi-core systems. Previously, he received a degree in Computer Engineering from the University of Campinas, Brazil, being accoladed as the best student of his class by the Engineering Council of the São Paulo State. He is currently pursuing a PhD at Universitat Politècnica De Catalunya.