Optimizing Offload Performance in Heterogeneous MPSoCs

Luca Colagrande[1], Luca Benini[1,2]

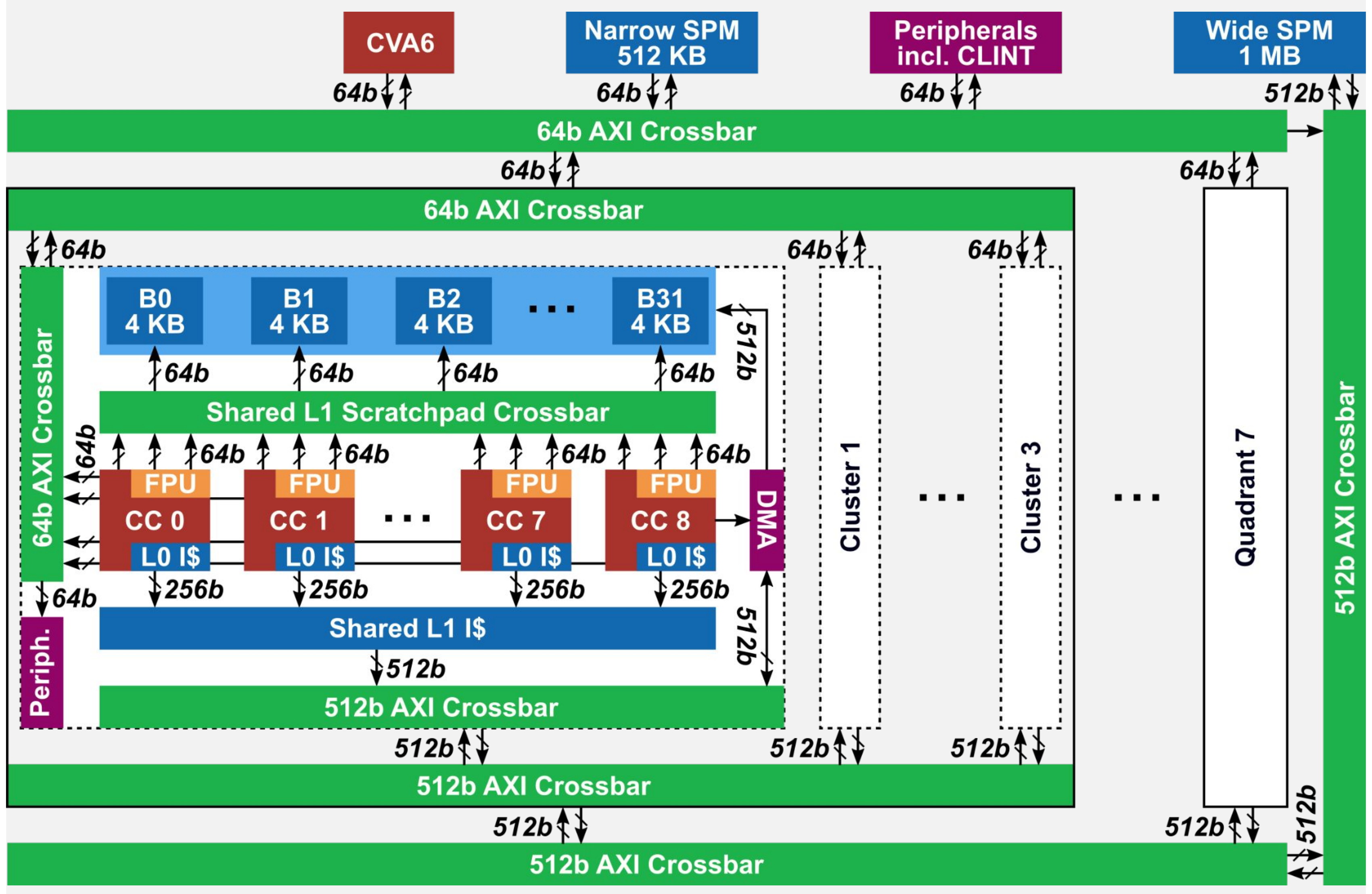
[1]Integrated Systems Laboratory, ETH Zürich; [2]DEI, University of Bologna

1 Introduction

Heterogeneous multi-core architectures combine on a single chip a few **"host"** cores, optimized for single-thread performance, with many small energy-efficient **"accelerator"** cores for data-parallel processing. **Offloading** a computation to the accelerator introduces a communication and synchronization cost which reduces the attainable speedup, particularly for small and fine-grained parallel tasks. It is the programmer's responsibility to define the **workload partition** between the host and the accelerator, and making a correct **offload decision** is non-intuitive[1]. This decision is about determining 1) if a portion of the workload can benefit or not from offloading, 2) the specifics on how to offload the workload, e.g. how many cores to employ, which can have a significant impact on performance[2].

2 Implementation

We developed this study on the fully **open-source Manticore MPSoC**[3], enabling a complete understanding of the offload overhead cycles.



We extended Manticore to support **multicasting** data from CVA6 (the host) to the individual accelerator clusters. To this end, we designed:

1. a multicast-capable AXI interconnect
2. a multicast-capable CVA6 load-store unit

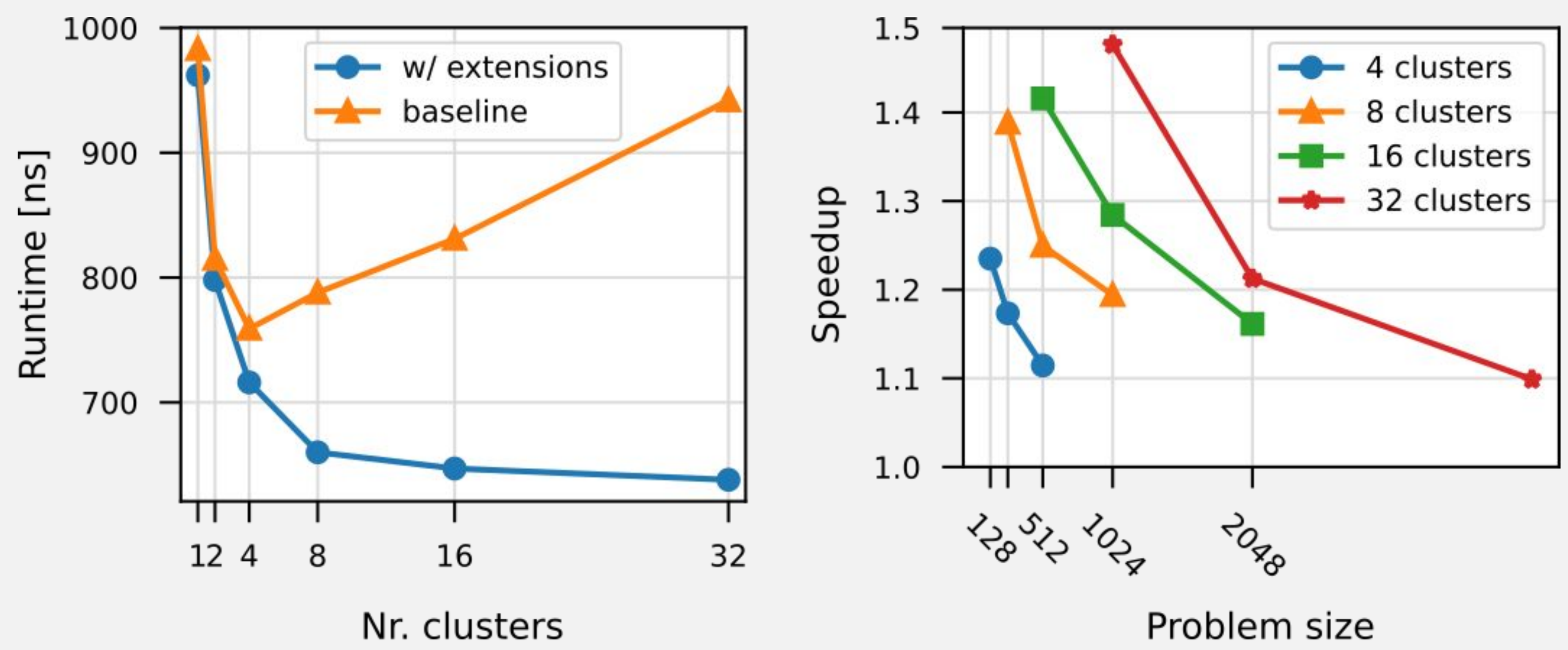
We **co-designed** the offload routines, using multicast to dispatch the job information to all accelerator clusters in parallel.

We further designed a dedicated **job completion unit**, to speed up the accelerator-to-host synchronization at the end of a job, by removing the overhead introduced by atomic operations.

3 Results and Discussion

All experiments are conducted through **cycle-accurate RTL simulations**, assuming a 1 GHz clock frequency.

We measure the runtime of an offloaded 1024-size DAXPY job with and without our extensions, for various number of clusters selected (left). Notably, the baseline presents a minimum, as the **offload overheads ∝ nr. clusters**. With our extensions this is no longer the case, yielding **speedups ∝ nr. clusters**.



We measure the speedup of the DAXPY job with our extensions, for different problem sizes (right). We find the **speedup ∝ (problem size)**$^{-1}$, as the offload overheads constitute a smaller fraction of the overall computation for larger problem sizes.

We develop a **quantitative model** for the runtime of an offloaded DAXPY kernel of size N onto M clusters, with an **error <1%**, allowing to formulate the offload decision as an optimization problem:

$$\hat{t}_{\text{offl}}(M,N) = 367 + \frac{N}{4} + \frac{2.6 \cdot N}{8 \cdot M}$$

4 Conclusion

We showed that:

1. co-designing the hardware and offloading routines can improve the speedup of an offloaded application by as much as **47.9%**, as measured on a fine-grained DAXPY kernel
2. optimizing the offload overheads is most significant for accelerators with **high core counts** and **fine-grained jobs**
3. it is possible to derive an **accurate model** of the offloading overheads, and overall offload runtime, which can be used to formulate the offload decision as an optimization problem.

References

[1] S. Che et al., "Rodinia: A benchmark suite for heterogeneous computing", IISWC, 2009.

[2] G. Araujo et al., "Nas parallel benchmarks with cuda and beyond", Software: Practice and Experience, vol. 53, no. 1, 2023.

[3] F. Zaruba et al., "Manticore: A 4096-core risc-v chiplet architecture for ultraefficient floating-point computing," Proc. of the IEEE Micro, vol. 41, no. 2, 2021.