# Optimizing Offload Performance in Heterogeneous MPSoCs

Luca Colagrande
*Integrated Systems Laboratory, IIS*
ETH Zürich, Switzerland
colluca@iis.ee.ethz.ch

Luca Benini
*Integrated Systems Laboratory, IIS*
ETH Zürich, Switzerland
lbenini@iis.ee.ethz.ch

***Abstract*—Heterogeneous multi-core architectures combine a few "host" cores, optimized for single-thread performance, with many small energy-efficient "accelerator" cores for data-parallel processing, on a single chip. Offloading a computation to the many-core acceleration fabric introduces a communication and synchronization cost which reduces the speedup attainable on the accelerator, particularly for small and fine-grained parallel tasks. We demonstrate that by co-designing the hardware and offload routines, we can increase the speedup of an offloaded DAXPY kernel by as much as 47.9%. Furthermore, we show that it is possible to accurately model the runtime of an offloaded application, accounting for the offload overheads, with as low as 1% MAPE error, enabling optimal offload decisions under offload execution time constraints.**

***Index Terms*—heterogeneous systems, fine-grain parallelism, job offloading, manycore accelerators**

## I. Introduction

In the post Dennard's scaling era, heterogeneous computing offers a solution to improve system performance, by integrating different compute units tailored to a diverse range of applications onto a single system. This work targets a class of heterogeneous computing systems referred to as heterogeneous (or asymmetric) multi-processor systems-on-chip (MPSoCs), which have met increasing interest in recent years [3], [4], [9]. These systems couple one or more high-performance "host" cores, guaranteeing high single-thread performance, with a multi-cluster fabric of energy-efficient "accelerator" cores, enabling the execution of intensive data-parallel computations at high energy-efficiency and performance points.

In this context, we refer to the process of handing over parts of the computation to the accelerator as *job offloading*. It is the programmer's responsibility to define the workload partition between the host and the accelerator, i.e. to define jobs to offload to the accelerator, and making a correct offload decision is non-intuitive [2]. This decision is not only about determining *if* a portion of the workload can benefit or not from offloading, but also about the specifics on *how* to offload the workload, e.g. how many cores to employ for a job, which may have a significant impact on performance [1].

To complicate the matter, offloading introduces several overheads, in the form of synchronization and communication between host and accelerator, which add up to the runtime and energy consumption of the job execution on the accelerator. Therefore, to enable effective fine-grained heterogeneous execution and exploit the full potential of heterogeneous architectures, these overheads must be reduced to a minimum.

Several works looked into the problem of quantifying offload overheads, but do not propose a generalized model, fail to assess the accuracy of the proposed models [6], or both [2], [5], [7]. All of these works study discrete CPU-GPU heterogeneous architectures, where CPU and GPU reside on separate chips interconnected by a PCIe bus. Due to the proprietary closed-source nature of these architectures, precise evaluation of the offload overheads' magnitude is not possible. Finally, to the best of our knowledge, no prior work looked into reducing the overheads associated to offloading in heterogeneous MPSoCs.

With this work, we show that by co-designing the hardware and offload routines it is possible to 1) decrease the offload overheads and improve offloaded application performance and 2) develop an accurate runtime model which can be used to formulate the offload decision as an optimization problem.

## II. Implementation

We develop this study on the Manticore MPSoC [9], as it combines the benefits of recent architectural trends with a fully open-source hardware design, allowing for a complete understanding of the offload overhead cycles. Manticore is a heterogeneous MPSoC targeting data-parallel floating-point workloads. It features a CVA6 host core [8] coupled to a symmetric multi-processor accelerator comprising a configurable number of energy-efficient cores, up to 288 in our experiments. The accelerator cores are organized in clusters of 9 cores each.

We extend Manticore's interconnect and CVA6's load-store unit and memory subsystem to support multicasting data from CVA6 to the individual clusters. Design details and implementation results are omitted as in this brief we focus on assessing and modeling the perfomance benefits of multicast communication on offload performance.

In addition, we designed and integrated a dedicated unit to handle the accelerator to host synchronization at the end of an offload. This unit implements a centralized credit counter. Upon an offload, CVA6 sets the number of accelerator clusters selected for offload as a threshold for the counter. When a cluster is done with the job, it atomically increments the counter by writing to a register which triggers the increment as a side effect. As soon as the counter reaches the threshold value set by CVA6, it automatically fires an interrupt notifying CVA6 of job completion.

979-8-3503-4859-0/DATE24/© 2024 EDAA

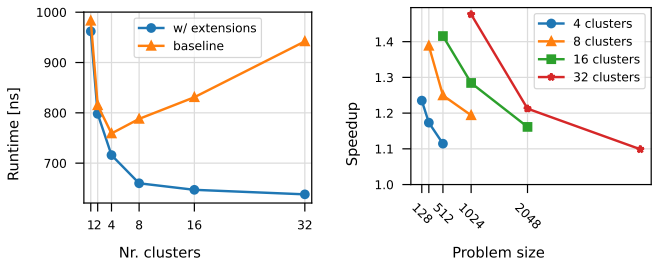


Fig. 1: Runtime of a 1024-dimension DAXPY job for various numbers of clusters employed (left); speedup of the DAXPY job with our extensions over the baseline implementation for various problem sizes and numbers of clusters (right).

## III. Results

All experiments are conducted through cycle-accurate RTL simulations of the Manticore MPSoC using QuestaSim 2022.3. The testbench drives all clocks at a frequency of 1 GHz, thus all runtimes reported in nanoseconds are in 1:1 correspondence with the runtime in CPU cycles.

To quantify the impact of the offload overheads, and evaluate the improvement from our extensions, we measure the runtime of an offloaded 1024-dimension DAXPY kernel with and without our extensions. Fig. 1 (left) presents the results of this comparison. The runtime in the baseline implementation presents a global minimum, owing to the fact that the offload overheads actually increase with the number of clusters. In fact, the job handler and arguments have to be dispatched to every cluster and, as this can only be done sequentially, the overhead depends linearly on the number of clusters. In the baseline design, when the number of clusters in the accelerator grows above four, the offload overhead starts to dominate (as it grows linearly, while the amount of work per cluster decreases).

With the multicast extension, dispatching occurs in parallel, so the overhead is instead constant. Thus, we can leverage additional clusters up to 32 while still decreasing execution time. The improvement of the proposed multicast solution reaches more than 300 cycles difference in the 32-clusters configuration. Offloading to more clusters would lead to negligible further improvements because of Amdahl's law.

In the previous experiment, we used a fixed problem size to highlight the dependency with the number of clusters, serving to show that *optimizing the offload overheads is most critical for large many-core accelerators*, as the workload distributed to each cluster gets smaller. Fig. 1 (right) adds the dependency on the problem size to the picture. It displays the speedup with our extensions over the baseline for different problem sizes and numbers of clusters. The speedup is always greater than one, although, for a fixed number of clusters employed, it decreases with the problem size. Indeed, as the problem size increases, also the job execution time increases, while the time to communicate the job information stays constant. It follows that the improvement on the latter has a smaller impact on the runtime (and therefore the speedup) of the overall offload. These results confirm that *optimizing the overheads is most significant for fine-grained parallel tasks*.

By inspecting the hardware and the compiled application, we can quantitatively model the overall runtime of an offloaded DAXPY kernel of dimension $N$ onto $M$ clusters:

$$\hat{t}_{offl}(M,N) = 367 + \frac{N}{4} + \frac{2.6 * N}{M * 8} \tag{1}$$

According to this model, the offloaded DAXPY kernel responds to Amdahl's law. The speedup attained by scaling the computation to multiple accelerator clusters is limited by the serial fraction made up in part by the offload overheads.

We validate the accuracy of the model on multiple problem sizes and number of clusters. For each problem size ($N \in \{256, 512, 768, 1024\}$) we calculate the mean absolute percentage error (MAPE) over all tested offload configurations ($M \in \{1, 2, 4, 8, 16, 32\}$):

$$MAPE(N) = \frac{100}{M} * \sum_{M} \frac{|t_{offl}(M,N) - \hat{t}_{offl}(M,N)|}{t_{offl}(M,N)} \tag{2}$$

The error is consistently lower than 1%, proving that the model can accurately estimate offloaded application runtime.

Such a model can be used to solve the offload decision problem for $M$. For instance, under the maximum runtime constraint $t_{offl}(M) \leq t_{max}$, we can easily invert Eq. (1) to derive the minimum number of clusters that would be needed for the offload to satisfy the constraints:

$$M_{min} = \left\lceil \frac{2.6 * N}{8 * (t_{max} - 367 - \frac{N}{4})} \right\rceil \tag{3}$$

## IV. Conclusion

We showed how, by enabling multicasting in the host-to-accelerator clusters interconnect, we can improve the speedup of an offloaded application by as much as 47.9%, as measured on a low vector dimension (1024) DAXPY kernel. Furthermore, we demonstrated the feasibility of developing an accurate runtime model accounting for the offload overheads, and showed how such a model can be used to derive optimal offloading parameters under an offload execution time constraint.

## References


[1] G. Araujo *et al.*, "Nas parallel benchmarks with cuda and beyond," *Software: Practice and Experience*, vol. 53, no. 1, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3056

[2] S. Che *et al.*, "Rodinia: A benchmark suite for heterogeneous computing," in *2009 IEEE International Symposium on Workload Characterization (IISWC)*, 2009.

[3] D. R. Ditzel *et al.*, "Accelerating ml recommendation with over 1,000 risc-v/tensor processors on esperanto's et-soc-1 chip," *IEEE Micro*, vol. 42, no. 3, 2022.

[4] D. Ignjatović *et al.*, "The wormhole ai training processor," in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, 2022.

[5] A. Kannan *et al.*, "Offloading to the gpu: An objective approach," 06 2015.

[6] S. Pei *et al.*, "Extending amdahl's law for heterogeneous multicore processor with consideration of the overhead of data preparation," *IEEE Embedded Systems Letters*, vol. 8, no. 1, 2016.

[7] S. Pei *et al.*, "Reevaluating the overhead of data preparation for asymmetric multicore system on graphics processing," *KSII Transactions on Internet and Information Systems*, vol. 10, 06 2016.

[8] F. Zaruba *et al.*, "The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit risc-v core in 22-nm fdsoi technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, Nov 2019.

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