Design and implementation of an architecture-aware hardware runtime for heterogeneous systems

Juan Miguel de Haro Ruiz*†, Jaume Bosch Pons*†, Daniel Jiménez-González*†
*Barcelona Supercomputing Center, Barcelona, Spain
†Universitat Politècnica de Catalunya, Barcelona, Spain
E-mail: {juan.deharoruiz, jbosch, djimenez}@bsc.es

Keywords—heterogeneous systems, task-dependence analysis, High-performance computing, FPGA, task-based programming models.

I. Extended Abstract

Parallel computing has become the norm to gain performance in multicore and heterogeneous systems. Many programming models allow to exploit this parallelism with easy to use tools. In this work we focus on task-based programming models. The parallelism is expressed with pieces of work called tasks that have data dependencies among them, and therefore have to be executed in a certain order. However, tasks that don't depend on any other running task can be executed in parallel.

A. Picos

A software runtime that handles task dependencies and schedules them is able to exploit high levels of parallelism with moderate size tasks. Nevertheless, with fine-grained tasks it suffers from performance degradation due to the overhead and thread contention.

Picos [1] is a hardware runtime that aims to solve this problem. By moving the dependence management to a hardware designed and optimized for that, the overhead is reduced dramatically. It can be used in different scenarios, for instance, one of the prototypes was implemented on an FPGA with hardware accelerators attached to it. Picos would take tasks from a host, and distribute them among the accelerators, which are designed to do specific calculations. It was also implemented on a RISC-V multicore system embedded in an FPGA. In both cases Picos was integrated into the OmpSs programming model.

All the previous work with Picos required adapting the programming model and a lot of hand-crafting to integrate the accelerators with Picos and OmpSs. The next step involved doing the inverse, adapting Picos to the programming model OmpS@FPGA [2], an extension to OmpSs with support for automatic generation of hardware accelerators on FPGAs. This allows faster testing and easier deployment of the hardware.

However, Picos exposed some limitations regarding area and timing when being implemented along big accelerators or many of them. It uses a significant amount of resources, and its design has a long critical path. This influences negatively the routing algorithm when the FPGA has a high resource usage rate. The consequence is that designs without Picos could scale in area much more than with it, thus achieving better performance even with higher runtime overheads.

B. New design

To overcome these limitations we decided to build a new Picos based on two main principles: to be configurable and use minimum resources. The previous Picos used always a fixed amount of memory and had support for at most 15 dependencies per task. This is not ideal for some applications, which may need more memory, or on the other hand, they may work well with a small amount of memory and/or dependencies. Adding parameters to tune the resource usage of Picos gives the flexibility to adapt to every application. Also using less overall resources gives it a bigger margin for the user to add more computation resources.

For maintainability purposes, the new Picos (called Picos Daviu) does not include the scheduler, it relies on the scheduler already present in the OmpSs@FPGA hardware runtime, Smart OmpSs Manager (SOM). The latter includes similar features as the old version of Picos (called Picos++) without dependence management, which has to be performed on a CPU. However, it includes new features such as task creation on a hardware accelerator [3]. This complements with Picos Daviu, which is integrated in SOM, since it allows the hardware runtime to solve the dependencies of tasks created inside the FPGA. Therefore, the CPU does not have to take care of these tasks. Everything is managed internally, thus removing communication between FPGA and CPU. By reducing device communication, the new hardware runtime, called Picos OmpSs Manager (POM) improves dramatically the performance of systems with high latency interconnections such as cloud.

Figure 1 a) and b) show the internal design of POM and Picos Daviu compared with Figure 1 c) that shows the design of Picos++. Since POM provides more features, it contains more modules than Picos++, but in fact it consumes less resources when Picos Daviu is configured with the same parameters as Picos++. This configuration allows to store 256

TABLE I: Post-synthesis resource utilization of POM and Picos++ for Xilinx FPGAs & maximum frequency

Hardware runtime	LUT	FF	BRAM	LUTRAM	F (MHz)
POM	9492	10184	48.5	177	300
Picos++	17692	17365	142.5	518	200

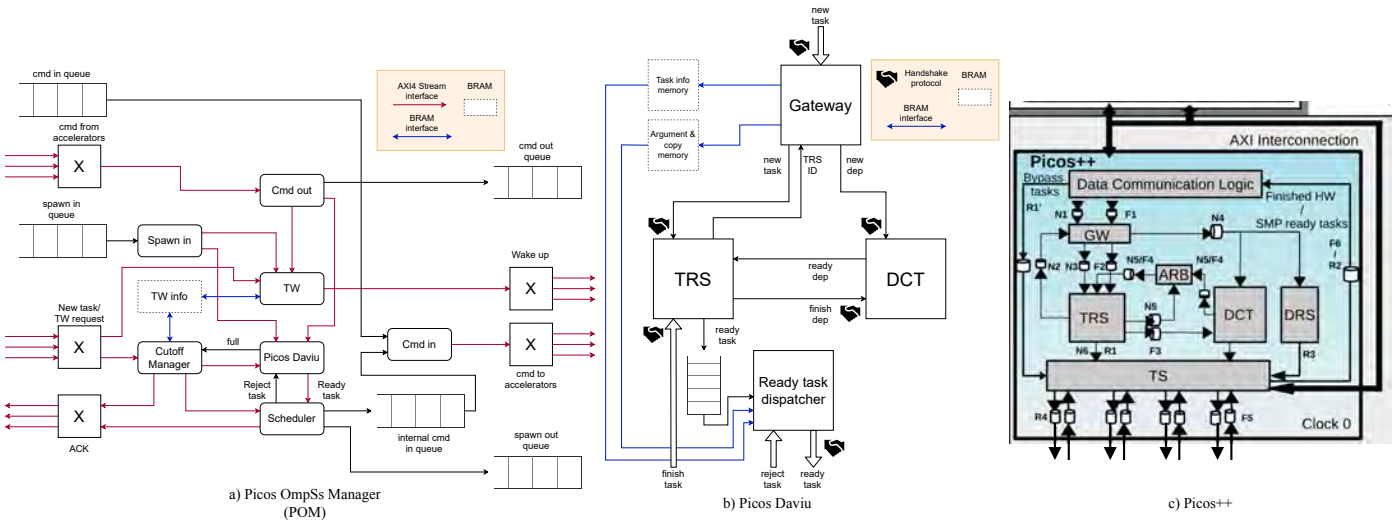

Fig. 1: POM, Picos Daviu and Picos++ internal designs

in-flight tasks, with a memory to store dependencies of 512 slots and another memory to store dependence relationship with the same number of slots. The maximum number of dependencies, arguments and copies (between main memory and the accelerator internal memory) per task is 15. The comparison of resources is shown in table I. The results have been obtained by hardware synthesis for Xilinx FPGAs. The usage of each resource is significantly decreased in POM, which shows a very high improvement in memory usage. It uses nearly 3 times less BRAMs and LUTRAMs than Picos++. The main difference is the removal of many queues used by Picos++ to communicate between its internal modules and the accelerators. In addition, the new design improves the maximum working frequency of Picos. While in the previous design a maximum of 200 MHz where achieved the new design has been successfully tested at 300 MHz.

C. Results

Both Picos versions have been tested with an N-Body like algorithm (Picos++ and Picos Daviu inside POM). In this specific application, the performance of the algorithm is directly proportional to the number of parallel operators that can be deployed in the FPGA (as it is an embarrassingly parallel application). Our preliminary tests show that the old Picos code was able to support 3 accelerators with 32 operators each (so a total of 96 operators) or 4 accelerators with 14 operators each (56 operators). With more accelerators, even if the FPGA fabric was not fully utilized, the routing tool was not able to obtain a viable design due to communication constraints between the different elements.

In contrast, POM, using exactly the same accelerators code was able to support 3 accelerators with 48 operators (144 operators in total) or 4 accelerators with 36 operators (also 144 parallel operators). Correspondingly the execution time diminished from the old Picos to the new from 292s and 173s to 116s. This performance improvement is obtained thanks to the better suitability of the new design to be deployed in an FPGA fabric and its improved connectivity.

D. Conclusion

In this work, we design a hardware runtime for task-based programming models which is configurable and uses low resources compared to its predecessor Picos. We compare both designs integrated in the OmpSs@FPGA programming model, and observe that the new design allows to attach more hardware accelerators, thus getting better performance.

II. Acknowledgment

This work has been supervised by Carlos Álvarez Martínez (Barcelona Supercomuting Center, Universitat Politécnica de Catalunya, Barcelona, Spain, calvarez@ac.upc.edu). It received funding from Spanish Government (projects SEV-2015-0493 and TIN2015-65316-P), and from Generalitat de Catalunya (contracts 2017-SGR-1414 and 2017-SGR-1328).

References

[1] X. Tan *et al.*, "A hardware runtime for task-based programming models," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 9, pp. 1933–1946, 2019.

[2] J. Bosch *et al.*, "Application acceleration on fpgas with ompss@fpga," in *International Conference on Field Programmable Technology*, ser. FPT, 2018, pp. 70–77.

[3] ——, "Breaking master-slave model between host and fpgas," in *PPoPP '20: 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, San Diego, California, USA, February 22-26, 2020*, R. Gupta and X. Shen, Eds. ACM, 2020, pp. 419–420. [Online]. Available: https://doi.org/10.1145/3332466.3374545

Juan Miguel de Haro received his BSc degree in Computer Engineering from Universitat Politècnica de Catalunya (UPC), Spain in 2018. He completed his bachelor thesis at the Department of Electronics and Information Systems in Ghent University, Belgium also in 2018. Later that year he started his MSc degree in High Performance Computing (HPC) at UPC, and started working with the OmpSs@FPGA group at the Barcelona Supercomputing Center (BSC).