Designing a Virtual Runtime for FPGA Accelerators in the Cloud

Mikhail Asiatici*, Nithin George*, Kizheppatt Vipin†, Suhaib A. Fahmy‡ and Paolo Ienne*
* Ecole Polytechnique Fédérale de Lausanne (EPFL),School of Computer and Communication Sciences
†Mahindra Ecole Centrale, School of Engineering Sciences
‡University of Warwick, School of Engineering
Email: {mikhail.asiatici, nithin.george, paolo.ienne}@epfl.ch, vipin.kizheppatt@mechyd.edu.in, sfahmy@ieee.org

***Abstract*—FPGAs can provide high performance and energy efficiency to many applications; therefore, they are attractive computing platforms in a cloud environment. However, FPGA application development requires extensive hardware design knowledge which significantly limits the potential user base. Moreover, in a cloud setting, allocating a whole FPGA to a user is often wasteful and not cost effective due to low device utilization. To make FPGA application development easier, firstly, we propose a methodology that provides clean abstractions with high-level APIs and a simple execution model that supports both software and hardware execution. Secondly, to improve device utilization and share the FPGA among multiple users, we developed a lightweight runtime system that provides hardware-assisted memory virtualization and memory protection, enabling multiple applications to simultaneously execute on the device.**

I. Introduction and related work

FPGAs can achieve high performance and energy efficiency in many applications. Recent work has demonstrated that cloud and datacenter applications can significantly benefit from using FPGAs. For instance, Microsoft's Catapult [1] leveraged FPGAs to almost double the throughput of search ranking, an enterprise-level datacenter application, with only a 10% increase in power consumption. Baidu used FPGA-accelerated neural networks to achieve an order of magnitude better performance for recognition applications at minimal additional power cost [2]. Additionally, there are also new server platforms being developed, such as the Intel XEON+FPGA platform [3], that aim to improve the integration of FPGAs into the standard computing ecosystem.

These developments have motivated research on integrating FPGAs into the cloud setting and offering them as a virtualized computing resource. For instance, Chen et al. [4], partitioned an FPGA into reconfigurable regions and partial reconfiguration is used to dynamically deploy accelerators. However, their study did not consider how design development flowc an be made easier or examine the cost of providing hardware-assisted memory management features, such as dynamic allocation and virtualization, that can greatly aid application development. Work on dynamic management of FPGA resources have explored managing partial reconfiguration from Linux [5] and extending an RTOS to manage hardware tasks [6]. However, these efforts only investigated system-on-chip scenarios with tightly coupled communication which is not the typical case in a cloud deployment.

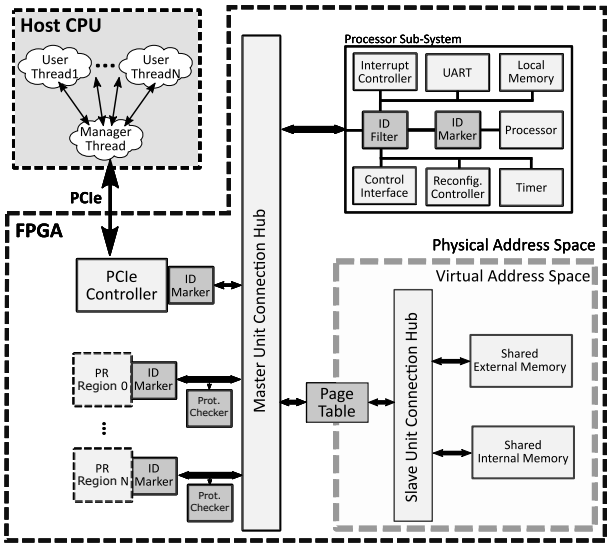

Fig. 1. Overview of the system and block diagram of the design implemented on the FPGA.

In this paper, we present a complete methodology and resource management framework that allows design and dynamic mapping of accelerators onto FPGAs in the cloud.

II. System Design

Figure 1 shows an overview of the architecture of our system. Here, a *host computer* in a cloud environment is connected to an FPGA board using a PCIe interface. The FPGA is initialized with a hardware system which has multiple regions where *hardware accelerators* can be instantiated at runtime via *partial reconfiguration* (PR). A *runtime manager* executing on the onboard processor is responsible for managing the FPGA resources and for communicating with the host over the PCIe. Users write *applications* that execute on the host CPU where some computations are accelerated by the FPGA. These parts are identified and provided to our toolchain to generate an *FPGA application package* which contains the bitstreams for the hardware accelerators and a scheduling program. The scheduling program is run on the onboard processor and is responsible for orchestrating the hardware accelerator execution by performing *system calls* to the runtime manager. The program can also utilize the low-latency access to data on

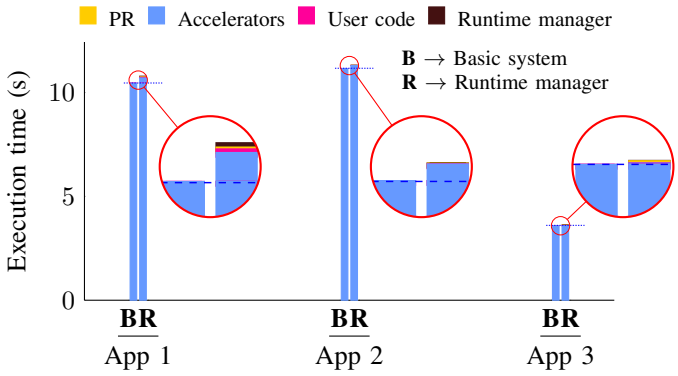

Fig. 2. Breakdown of the overhead components due to our infrastructure for all our benchmarks. The total overhead is always less than 4%.

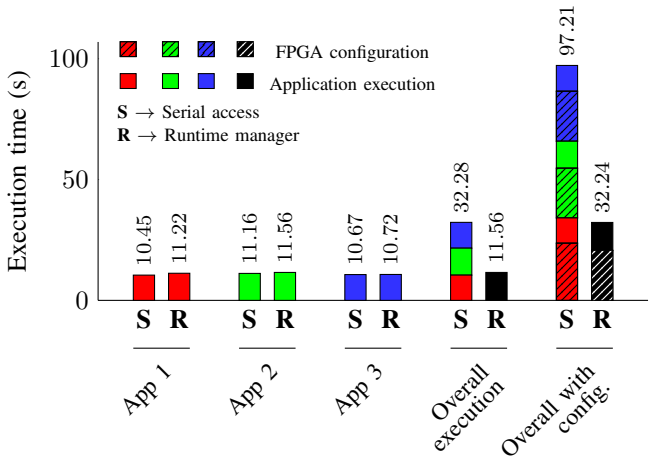

Fig. 3. Comparison between simultaneously sharing the FPGA with our runtime manager among multiple applications and serially providing exclusive use of the FPGA without PR.

the FPGA memory shared with the accelerators and perform additional computations.

On the FPGA, memory accesses from onboard processor and accelerators in the PR regions are mediated by our virtualisation infrastructure. *ID marker* modules mark every bus transaction with an ID based on the application to which the corresponding master has been currently assigned. The ID is used by the *page table* and the *ID filters* to perform the virtual-to-physical address translation and to implement memory protection.

The runtime manager is built on top of FreeRTOS, a simple open-source, multi-threading real time operating system [7]. We modified FreeRTOS' kernel in order to integrate it with our custom virtualization and protection hardware.

III. Design Flow

To easily generate applications, we extended our prior work to target this platform from high-level domain-specific language (DSL) specifications [8]. Alternatively, users can also provide high-level synthesis (HLS) or RTL specifications for the accelerators in the application. Our toolchain then automatically generates an FPGA application package which contains the accelerators partial bitstreams and the code for the onboard processor. Currently, the host CPU program is manually generated, but we plan to automate this in the future.

IV. Results

To evaluate our approach, we implemented our infrastructure on a Xilinx VC709, which includes a Virtex 7 FPGA and 8 GB of DDR3 memory, and used 3 benchmark applications.

Firstly, we measured the execution time of each application on our new platform and compared it with a simpler system without any of the virtualization infrastructure and runtime manager. The results in Figure 2 show that the overhead is less than 4%, and is mostly due to the memory virtualization infrastructure that increased the latency of each bus transaction.

Secondly, to analyse the benefits of simultaneously sharing the FPGA, we compared the execution time of a workload comprising the 3 benchmark applications on the novel and on the simpler system we used earlier for the overhead analysis. The applications execute sequentially on the simpler system since it does not have the infrastructure to share the FPGA among the applications. The results in Figure 3 clearly demonstrate the benefit of sharing the FPGA among multiple applications. If the time needed to reconfigure the FPGA is also considered, as seen in the figure, the performance advantage offered by our proposed architecture over the simpler system becomes even more evident.

V. Conclusions

To make FPGAs suitable for the cloud environment, we propose an environment that provides application developers with facilities such as memory management, virtualisation and a hardware abstraction layer. Our methodology also includes a design flow that enables developers to write FPGA-accelerated applications at different levels of abstraction, from low-level RTL to high-level DSLs. Our preliminary results show that FPGAs can be virtualised with limited overhead and that sharing an FPGA enables to increase the effective area and bandwidth utilization of the board.

References

[1] A. Putnam *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2014.

[2] J. Ouyang, S. Lin, W. Qi, Y. Wang, and B. Yu, "SDA: Software-defined accelerator for large-scale DNN systems," in *Proceedings of HOT CHIPS*, 2014.

[3] P. K. Gupta, "Intel®Xeon®+FPGA platform for the data center," 2015, talk presented at FPL'15 Workshop on Reconfigurable Computing for the Masses. [Online]. Available: http://reconfigurablecomputing4themasses.net/files/2.2%20PK.pdf

[4] F. Chen, Y. Shan, Y. Zhang, Y. Wang, H. Franke, X. Chang, and K. Wang, "Enabling FPGAs in the cloud," in *Proceedings of the ACM Conference on Computing Frontiers*, 2014.

[5] J. A. Williams and N. W. Bergmann, "Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip," in *Proceedings of the International Conference On Engineering of Reconfigurable Systems and Algorithms (ERSA)*, 2004.

[6] E. Lübbers and M. Platzner, "ReconOS: An RTOS supporting hard-and software threads." in *Proceeding of the International Conference on Field Programmable Logic and Applications (FPL)*, 2007, pp. 441–446.

[7] R. Barry, *FreeRTOS reference manual: API functions and configuration options*. Real Time Engineers Limited, 2009.

[8] N. George, H. Lee, D. Novo, T. Rompf, K. J. Brown, A. K. Sujeeth, M. Odersky, K. Olukotun, and P. Ienne, "Hardware system synthesis from domain-specific languages," in *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*. IEEE, 2014, pp. 1–8.