Providing of FPGA Resources as a Service: Technologies, Deployment and Case-Study

Inna Kolesnyk[1], Artem Perepelitsyn[2], Vitaliy Kulanov[3]

National Aerospace University "KhAI", Chkalov str. 17, 61070 Kharkov, Ukraine.
[1] i.kolesnyk@csn.khai.edu
[2] a.perepelitsyn@csn.khai.edu
[3] v.kulanov@csn.khai.edu

Abstract. In this study we analyzed some aspects of applying the Field Programmable Gate Array (FPGA) technology based on Peripheral Component Interconnect Express (PCIe) bus to create a cloud service. Task classification for a FPGA as a Service (FaaS) was proposed. We considered various approaches to FaaS deploying and feasible ways of communication between the cloud infrastructure and the FPGA platform. We elaborated and approved cost-effective FaaS architecture, which is based on a set of FPGA boards. The input-oriented task based on brute force search of polynomials for nonlinear feedback shift registers of the second degree was implemented. The approach of creating multiparametrized tasks for a wide range of FPGA resources was proved to be effective.

Keywords: Cloud Service, FPGA, FPGA as a Service, FaaS Platform, FaaS Tasks, FaaS Deployment, Brute Force Search, NLFSR

Key Terms: Computation, ConcurrentComputation, ServiceComposition, Data, HighPerformanceComputing

1 Introduction

The growing demand for the services provided by cloud technologies is due to their advantages over traditional computing. Accessibility anywhere, relatively low requirements for computing power of the client machine, and as a result, lower power consumption for end user, saved hardware and time resources - all greatly accelerate this trend.

Applying the FPGA technology in a cloud computing is of great interest nowadays [1-3]. It can significantly speed up many performance-intensive tasks (or services): from digital signal processing (digital video processing, audio signal processing, spectral estimation, speech recognition, imaging processing, biomedicine, radar, sonar, etc.) to specific mathematical calculations for science. It also allows to propose more energy-efficient solutions for data centers [4, 5], especially in cases where some part of a cloud infrastructure can be deployed on FPGA platform.

In an article on FPGA application analysis [6] it is shown that FPGA can interact with a cloud infrastructure on three different levels. They are the following:

1. FPGA as a Service (FaaS) - providing end users with "raw" FPGA resources and giving them ability to define their own projects.
2. FPGA for a Service (FfaS) - providing end users with already defined services (e.g. audio/video processing, specific DSP algorithms etc.). In this case customer may not even know about the hardware configuration/characteristics.
3. FPGA for Cloud Infrastructure - the case when FPGA is used to support cloud infrastructure itself (networking, virtualization platform etc.).

Among the variety of existing levels, FaaS still requires more attention and research as there are many methods, tools and techniques that can be applied. The scope of this paper is to analyze existing FaaS solutions, propose FaaS deployment techniques and carry out a research work in the deployed computational cluster.

2 FPGA as a Service: PCIe-based Approach

For high-performance computing there must be a highly effective interaction between the CPU (physical hardware) and the accelerators on the FPGA. Among the various methods of connecting, the PCIe bus is suitable for loosely coupled accelerators because of its high capacity. A high-performance library used for communication of the PCIe FPGA with the rest of the system is the key to the enhanced use of FPGA-accelerators. However, due to the fact that such a universal library does not exist, FPGA developers have to write a significant amount of code on the side of FPGA for FPGA. They must also develop a custom code (e.g. drivers, APIs) to use the FPGA accelerators. All of this only complicates the developers' work. Today it is a major problem for the mass usage of FPGA based PCIe accelerators.

We analyzed the existing solutions targeted at incresing the efficiency and flexibility of using FPGAs with integrated PCIe.

PCIe is a multi-layered protocol that includes a physical layer, a data link layer and a transaction layer. Data is formed into packets and transmitted on the transaction level. In order to interact with the FPGA through the PCIe, developers can use only a limited set of common functions reserved for the data transmission. Those who are interested in working with the low-level functions can use IP-cores for the PCIe, provided directly by the vendors. However, it is often necessary to use third-party solutions, which simplify the communication process with the FPGA via the PCIe and accelerate the development. Existing solutions consist of hardware (IP-core) and software (drivers, GUI) parts and a set of libraries/APIs.

On the hardware side, developers gain access to the PCIe interface via IP-core. It does not require knowledge of addresses, buffer sizes or PCIe packet format. The data is received and sent by invoking special functions, which act in FIFO-like manner.

The software includes drivers and the utilities for configuring. The driver can support simultaneous work of the multiple FPGAs. The software can include a set of libraries in such languages as C/C++, Python, Java and others to write third-party programs.

Some features of existing PCIe-based FPGA projects are shown in table 1. A large number of projects have open source code and support different operating systems.

Table 1. The Features of the FPGA based on PCIe bus

Name	FPGA Vendor	PCIe interaction method	OS Compatibility	API	Open Source
Riffa [7]	Altera, Xilinx	DMA	Windows, Linux	C, C++, Python, Matlab, Java	Verilog, VHDL, C, C++, Java, Python, Matlab
MPRACE [8]	Xilinx	DMA	Linux		-
XAPP 1052 [9]	Xilinx	DMA	Windows, Linux	-	Verilog, VHDL, C
XAPP 859 [10]	Xilinx	DMA	Windows	-	Verilog
PLDA EZDMA2 [11]	Altera, Xilinx	DMA	Windows, Linux	C++	-
PLDA XpressRICH3 [12]	Altera	DMA	Windows, Linux	C++	Verilog, C, C++, Java

3 FaaS Task Classification

One of the main advantages of the FPGA technology application is the ability to implement non-standard hardware based solutions, especially when microprocessors show low efficiency/resource capability, and production of ASIC is still unreasonable due to the price per product unit. Also, there are special types of computational tasks, which show better performance and energy-efficiency for FPGA compared with CPU-based solutions and the increase is immense for GPU-based solutions [13].

Exactly for such type of tasks FPGA resources may be provided as a service. Typically that is data processing for science, imaging, cryptography, medicine and industry. The data flow of such tasks is shown in Fig. 1.

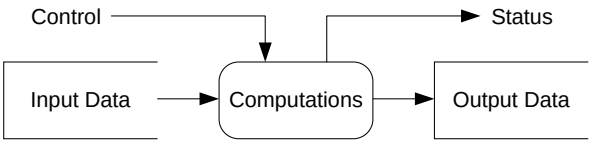

Fig. 1. Data Flow in typical tasks for FPGA as a Service

According to symmetry of data path throughputs the FaaS tasks is classified into:

- Symmetric Tasks - symmetrical throughputs (for symmetric algorithms);
- Input-oriented Tasks (IOT) - dominance of input dataflow (computationally intensive tasks such as filtering, inverse problem solving, convenient number search, or brute force search);
- Output-oriented Tasks (OOT) - dominance of output dataflow (all types of data generators).

Symmetrical or output oriented throughput tasks require a communication channel with a high bandwidth. In this case, the PCI Express or Gigabit Ethernet interfaces are strongly recommended. A large amount of tasks with non-intensive data-exchange operations may be implemented in FPGA even without fast communication channels, simply via USB, UART and other widely used interfaces.

4 FaaS Deployment Approach

In spite of all the advantages of the FPGA PCIe-based platforms, they are still too expensive, and when dealing with input-oriented tasks more cost-effective solutions can be applied.

The proposed coarse-grained FaaS infrastructure consists of several components (see Fig. 2):

- FPGA Development Kit Boards, connected to a server via the USB interface;
- A server machine with running JTAG Server, FaaS Tasks WEB Server and Server-side application for distributing and collecting data;
- User PC with CAD Tool and a WEB browser.

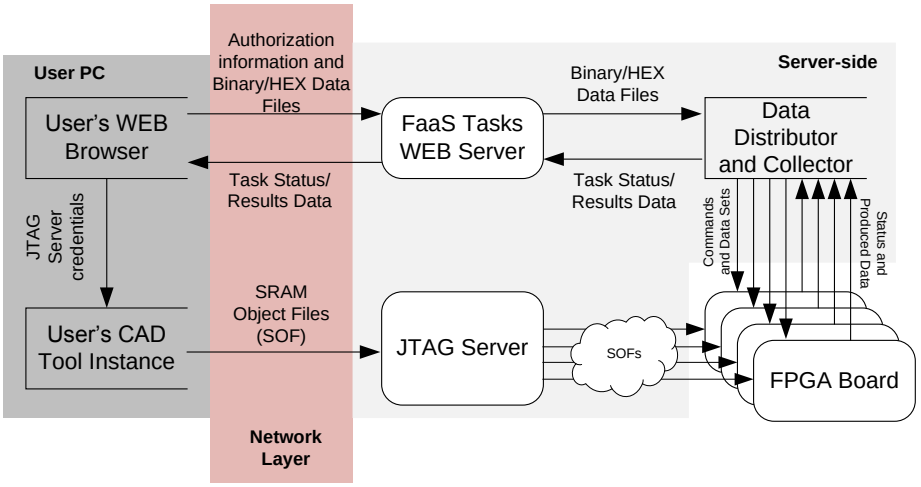

Fig. 2. Coarse-grained FaaS infrastructure

The efficiency of task implementation in FaaS depends on its parallelization feature. If the data dependency can be reduced and the algorithm allows the parallelization, it's possible to organize the scaling of the FPGA system in a wide range of available resources. The implementation of tasks using multiparametrization allows creating a universal project, which can be used in a wide range of FPGA chips without redesigning.

5 Case-Study: Brute Force Search

To verify the proposed approach, FaaS was deployed. A set of Altera DE2 boards was connected to the host computer (server) via USB interface. In order to give the remote access and ability to program FPGA boards, the JTAG server was configured.

The described service was used for solving a scientific problem dealing with the brute force search of polynomials for nonlinear feedback shift registers [14]. This task is input-oriented, which means that it has the dominance of pre-generated on the user side input data. The data set consists of millions of coefficients for non-linear polynomials. The coefficients of each polynomial can be processed separately. Therefore, this task is completely suitable for parallelization by means of multiparametrization. Only a small part of these coefficients can generate maximum length sequence.

To achieve the best performance, the implementation of search block was separated from the other parts of the project using dual port RAMs. The dataflow in one channel (FPGA) is shown in Fig. 3.

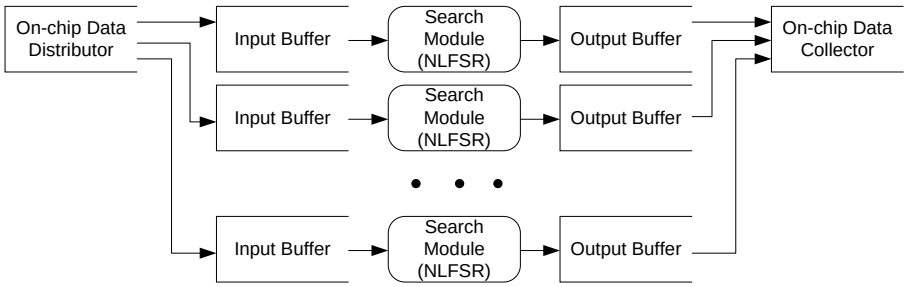

Fig. 3. Scalable implementation of brute force search task for FaaS

Full amount of input data was divided between four FPGA boards programmed with the same project (the same SOF). The distribution and collection of data was carried by a custom communication software.

This example of FaaS shows that such type of tasks can be parallelized not only inside of an integrated circuit (chip) but also between independent boards. If multiparametrization was used during development of task implementation, the project may be ported to another chip family without redesigning. FaaS in this case can include various types of boards.

6 Conclusion

Over the past years the FPGA technology has taken a big leap towards conquesting the cloud service market. The common trend is that the growth in this area is still in progress due to the advantages that programmable logic can offer.

In this article FPGA as a Service was considered. The analysis of existing techniques showed that one should take into account a great variety of different approaches of deploying FaaS. In most cases the choice depends on many factors, from the

final cost of FaaS infrastructure to type of task that is going to be executed. It is also showed that according to the symmetry of data path throughputs the FaaS tasks can be classified to symmetric, input-oriented, and output-oriented tasks.

A cost-effective solution for FPGA as a Service was proposed. This architecture can be recommended for so called input-oriented tasks, where the requirements for the input and output dataflow are not so strict. To verify the proposed approach, a high performance computational task was carried out, used for solving scientific problem based on brute force search of polynomials for nonlinear feedback shift registers of second degree. It allowed saving the time and resources to get final results.

References

1. Chen, F., Shan, Y., Zhang, Y., Wang, Y.: Enabling FPGAs in the Cloud. In: Proceedings of the 11th ACM Conference on Computing Frontiers Article No. 3, pp. 1–10. Cagliari, Italy (2014). doi: 10.1145/2597917.2597929
2. Gupta, P.: Xeon+FPGA platform for the data center, https://www.ece.cmu.edu/
3. Fahmy, S. A., Vipin, K., Shreejith, S.: Virtualized FPGA accelerators for efficient cloud computing. In: IEEE International Conference on Cloud Computing Technology and Science, pp. 430–435. Vancouver, Canada (2015). doi: 10.1109/CloudCom.2015.60
4. Yanovskaya, O., Yanovsky, M., Kharchenko, V.: The concept of green Cloud infrastructure based on distributed computing and hardware accelerator within FPGA as a Service. In: Proceedings of the IEEE East-West Design & Test Symposium (EWDTS), pp. 45–48. Kiev, Ukraine (2014). doi: 10.1109/EWDTS.2014.7027089
5. Neshatpour, K., Malik, M., Ghodrat, M. A., Sasan, A., Homayoun, H.: Energy-efficient acceleration of Big Data analytics applications using FPGAs. In: BIG DATA '15 Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), pp. 115–123. Washington, DC, USA (2015). doi: 10.1109/BigData.2015.7363748
6. Kolesnyk, I. N., Kulanov, V. O., Perepelitsyn, A. E.: Analysis of FPGA Technologies Application as a Part of Cloud Infrastructure. In: Radioelectronic and computer systems, 6 (80), pp. 130–135 (2016).
7. A Reusable Integration Framework For FPGA Accelerators, http://riffa.ucsd.edu
8. Marcus, G., Gao, W., Kugel, A., Manner, R.: The MPRACE framework: An open source stack for communication with custom FPGA-based accelerators. In: Programmable Logic, VII Southern Conference, Cordoba, Argentina (2011). doi: 10.1109/SPL.2011.5782641
9. Bus Master Performance Demonstration Reference Design for the Xilinx Endpoint PCI Express Solutions, https://www.xilinx.com
10. Lund, K., Naylor, D., Trynosky, S.: Virtex-5 FPGA Integrated Endpoint Block for PCI Express Designs, https://www.xilinx.com
11. EZDMA2 IP for Altera Devices, http://www.plda.com
12. XpressRICH3-AXI for ASIC, https://www.plda.com/
13. Fowers, J., Brown, G., Cooke, P., Stitt, G.: A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications. In: FPGA '12 Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pp. 47–56. Monterey, California, USA (2012). doi: 10.1145/2145694.2145704
14. Poluyanenko, N.: Development of the search method for non-linear shift registers using hardware, implemented on field programmable gate arrays. In: EUREKA: Physics and Engineering, pp. 53-60 (2017). doi: 10.21303/2461-4262.2017.00271