Modular FPGA Acceleration of Data Analytics in Heterogenous Computing

Elias Koromilas[i]
NTUA, Athens, Greece
LenXcale, Spain
elias.koromilas@gmail.com

Christoforos Kachris
ICCS-NTUA
Athens, Greece
kachris@microlab.ntua.gr

Dimitrios Soudris
ICCS-NTUA
Athens, Greece
dsoudris@microlab.ntua.gr

Francisco J. Ballesteros[ii]
Univ. Rey Juan Carlos
Spain

Patricio Martinez, Ricardo Jimenez-Peris
LeanXcale, Spain

[i] Currently with InAccel, Inc.
[ii] Work performed in the context of a contract with URJC funded by LeanXcale.

***Abstract*—**Emerging cloud applications like machine learning, AI and big data analytics require high performance computing systems that can sustain the increased amount of data processing without consuming excessive power. Towards this end, many cloud operators have started deploying hardware accelerators, like FPGAs, to increase the performance of computationally intensive tasks but increasing the programming complexity to utilize these accelerators. VINEYARD has developed an efficient framework that allows the seamless deployment and utilization of hardware accelerators in the cloud without increasing the programming complexity and offering the flexibility of software packages. This paper presents a modular approach for the acceleration of data analytics using FPGAs. The modular approach allows the automatic development of integrated hardware designs for the acceleration of data analytics. The proposed framework shows the data analytics modules can be used to achieve up to 3.5x speedup compared to high performance general-purpose processors.

Keywords*—*data analytics, databases, cloud computing, FPGAs, heterogeneous computing

I. Introduction

As the traffic in the data centers, continues to increase rapidly, data center operators are looking for novel systems that can provide much higher performance than the typical processors without consuming excessive amounts of power. In the domain of embedded systems, vendors and designers have embraced the heterogeneity paradigm in order to provide high performance systems that are at the same time are energy efficient. Typical processor (high performance and low power) are used to provide the flexibility and support the typical software stack, while specialized co-processors are used to offload the processor for the most widely used tasks such as encryption, compression, and signal processing.

Currently, it seems that cloud computing and data center operators are embracing the advantages of heterogeneity in order to provide high performance and energy-efficient systems. Data center operators in the last few years started deploying GPGPUs in hyperscale data center in order to provide to the cloud users high performance systems and keep the flexibility of the processors. In the last couple of years, data center operators are also looking on how to allow the cloud users to utilize the performance of specialized systems by deploying FPGAs in the data centers.

Although FPGAs can provide much higher performance than GPUs in several applications, FPGAs were not widely deployed so far due to the high programming complexity. FPGAs had to be programmed (configured) using hardware programming languages such as VHDL and Verilog that describe the circuits that are used to perform a specific task. Hardware programming languages are much more difficult than C/C++ and Java and therefore , FPGA were mainly used for embedded systems and for modules that are based on a specific standard (compression or encryption) or for application in which you need high performance but the volume of the systems is low and you cannot afford to develop and ASIC.

In the past, there were several efforts (both from academia and industry) to program the FPGAs using high level languages like C/C++, Java and Python. However, most of these efforts lack the efficiency and the performance of the VHDL or Verilog diminishing the advantages for the FPGAs. However, in the last few years, the larger FPGA vendors (Altera which bought by Intel) and Xilinx started supporting the programming of FPGAs using OpenCL [1][2][3]. In cases that the application is mainly data-driven, OpenCL can provide similar performance with the hand-written modules developed in VHDL or Verilog. The use of OpenCL for programming the FPGAs fostered the development of a new market. The market of utilization to FPGAs in cloud computing where the cloud users can program the FPGA without getting the design complexity of hardware description languages.

Although that programming the FPGAs with OpenCL is much easier than hardware description languages, still cloud users prefer to use more widely used languages like python, Scala and Java. Therefore, the most promising approach for the widely deployment of FPGAs in the cloud is the decoupling of the cloud users from the FPGA developers through the use of modules. The use of a marketplace is the most efficient way to allow cloud developers to utilize 3rd party FPGA modules in order to speedup their applications without any prior knowledge of FPGAs. The use of a marketplace for the FPGA in the cloud has sparked a new ecosystem and a new emerging market; acceleration as a service. Amazon AWS has already started providing into its market place 3rd party FPGA modules for their FPGAs. Accelize is another company that provided a marketplace for FPGAs that allows these modules to be deployed to any data center that utilizes FPGAs. In the domain of the research efforts, VINEYARD [5] has initiated a marketplace for opensource modules for the research community. Companies like InAccel, rENIAC and Falcon Computing develop FPGA

modules that can deployed in FPGAs that are hosted in the cloud operators like Amazon AWS, Alibaba Cloud and Huawei. Accelize and Amazon already provide marketplaces for these FPGA modules. Cloud developers can browse the marketplaces and rent the FPGA modules in the form of IP core. Then the cloud users can select in which data center to deploy the accelerators based on the cost, availability etc. All of these companies provide a wide range of IP cores. For example, InAccel provides ready to use FPGA modules for machine learning applications like logistic regression, k-means clustering and recommendation engines. The use of a library-based approach can be used to foster the widespread adoption of hardware accelerators in the cloud. The use of easy-to-use modules that can offload the typical processor without changes in the original code can be the preferred way to help the utilization of accelerators.

In this paper we first give an overview of the high-level framework of the VINEYARD and then we present a use-case on the acceleration of data analytic application. We present the development of a modular approach for the acceleration of data analytic application based on acceleration modules for the most widely used tasks in data analytic applications.

II. THE VINEYARD ACCELERATED FRAMEWORK

VINEYARD, a EC-funded project, aims to allow cloud users to easily utilize accelerators (FPGAs, and dataflow engines) in heterogeneous data centers in the same way as software packages and with the same flexibility as any other cloud services. VINEYARD provides an integrated framework that abstract the main hardware accelerators drawbacks such as resourcing, scheduling, programming and utilization of the accelerators, thus making easier the utilization of the FPGAs.

VINEYARD provides the required APIs that enables the utilization of the heterogeneous infrastructures without any other modifications on the applications. Some of the tasks such as sorting of data, encryption, compression, pattern matching, etc. are extremely computationally intensive. These tasks have been implemented in hardware as customized intellectual-property (IP) accelerators that can achieve much higher performance with lower power consumption. These hardware accelerators are stored in an IP repository (VineStore) that interface with the VINEYARD resource manager and scheduler. For each application there are several versions of the accelerators based on the available platform (FPGA, DFE, and Xeon Phi).

To interface with hardware accelerators, vendor specific libraries are used for the low-level communication with the hardware resources. (e.g. Xilinx's SDAccel and Intel's OPAE library [8]). On top of these interfaces, VINEYARD has developed FPGA drivers that are required for the communication with the vendor specific libraries. On top of the accelerator drivers, VINEYARD has developed the VINEYARD controller (VineController) that allows the abstraction of the accelerator drivers from vendor-specific libraries and the utilization of the accelerators from high level programming frameworks.

For the cloud computing applications, the Software stack of each node contains the VMs that are running on the processor, the Local scheduler that dispatches the job to the local accelerators, the VineTalk that allows the virtualization of the underlying hardware resources, and the VineController that serializes the jobs to the hardware resources.

Applications can either use directly the VineTalk [7] and VineController for the utilization of the resources or through the use of a central scheduler. In the first case, multiple applications can share the resources of a single accelerator through the virtualization of the resources. The Scheduler and the resource manager are used when these applications want to access several heterogeneous infrastructures. In that case, VineTalk can be used optionally if several applications want to share the hardware resources.

In this paper, we present the architecture and the performance evaluation for the data analytics accelerations. A modular approach has been developed that allows the acceleration of data analytic applications based on the most widely-used tasks in data analytics. The main building blocks have been developed as IP modules for FPGAs that can be used to speedup any data analytic application.

III. DATA ANALYTICS

A. Data analytics

Current trends in big data analytics, with constantly increasing data sizes, demand incredibly high computational performance. Moving from traditional database systems to the Big Data environment, FPGA, as a representative of reconfigurable devices, provides a unique opportunity to build an efficient query processing platform, by constructing high-throughput execution units with the additional aim of minimizing reconfiguration overheads and data movements.

IV. DATA ANALYTIC MODULES

This section presents the FPGA modules that have been developed, so far, but also provides guidance on leveraging the functionalities of the Intel FPGA Software Development Kit (SDK) for OpenCL.

A. Sorting

Sorting an array, or a column, of numbers is a continuously active area of research as it is an essential primitive operation in many application domains, including databases. Quicksort based algorithms have traditionally been considered to have the best average case performance among software sorting algorithms. However, recent advances in CPU, GPU and FPGA architectures have brought merge sort based algorithms, like Batcher's ones, to the forefront of performance as they are able to exploit new parallel architectures more effectively and better utilize the limited amount of bandwidth.

After several experiments with different possible approaches, we concluded that Bitonic sort was the best algorithm choice for the FPGA sorter implementation, as it could utilize more efficiently the available hardware resources, maximizing the performance.

One of the first challenges that we had to face was the vectorization of the FPGA kernel, in order to exploit the memory locality and the inherit parallelism. The complexity of the algorithm and the sequence of the comparisons (at the different passes of different stages), however, did not allow the Intel tools to perform the vectorization automatically. So we performed this vectorization manually, along with several

special optimizations, eliminating unnecessary kernel invocations and leading to a more sophisticated design with over 250 lines of additional OpenCL kernel code.

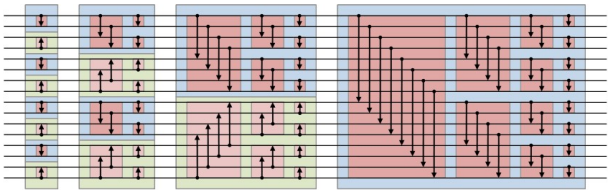

Figure 1 Architecture of Bitonic sorting network with 16 inputs.

A second important challenge was the modification of the algorithm to support arbitrary array sizes (not only power of two) without degrading performance and requiring extra memory allocation. For that reason, we used a technique called 'virtual padding'. Bitonic sort algorithm was modified in a way that in each phase the 'extra' values (up to the next power of 2) are never moved and thus do not have to exist physically.

B. Hashing

Hashing is an essential part in several database operations, such as joins. The disadvantage of simple hash functions is that they produce imperfect data distributions, leading to an increased number of collisions. On the other hand, robust hash functions produce balanced distribution, but they are computationally expensive. As a starting point, we focused on MurmurHash algorithms which, along with Google's CityHash and FarmHash families, are state of the art hash functions used in the modern computer world.

The MurmurHash3 FPGA module that we created, takes as an input an array of 64-bit keys and outputs an array with their 32-bit hashes. The hashing algorithm is basically a series of bitshifts, XORs and multiplications and thus is very suitable for hardware implementation. Our hardware design is fully pipelined, while also 8 parallel hash function units are used to exploit the maximum SIMD kernel vectorization. Finally, the kernel is manually replicated, with 2 instances of the kernel being available for concurrent execution.

C. Arithmetic and Comparison Operators

Comparison operators have a key importance in an SQL query as they limit (filter) the amount of data for further processing, according to the specified condition. Arithmetic operators, are also very useful and common, as they perform mathematical calculations between two expressions in the query. Both types of operators are mostly used in the WHERE clause of the SQL statement, while such operations are normally performed on a large amount of data (whole or partial columns).

In order to meet the requirements for fast and flexible processing, we developed different FPGA modules, one for every operation. At a filtering module, for example, an input column can either be compared with another input column (from the same or another table), or it can be compared with a constant. Equivalent functionality is provided by the arithmetic modules.

D. MIN(), MAX() and SUM() Functions

Aggregation operations, like MIN, MAX and SUM, are used to reduce an input set to a single value. Similar to the previous units, our FPGA aggregation units are designed to take columns as inputs, however their inherit loop-carried dependency does not allow the kernel to be pipelined at thread level. For the integer aggregations, this dependency does not affect the loop pipeline (inside the kernel) and successive iterations are launched every cycle. On the other hand, this data dependency, combined with the latency of the FPGA floating point compare and addition operators, leads to an increased loop Initiation Interval (II) and thus to poor performance. To enable the compiler to handle such kernels that carry out double precision floating-point operations efficiently, we used a technique that removes the loop-carried dependencies by inferring a shift register.

V. Data Analytic Modules

A. Information on Kernel Types

ND-Range kernels are built to accept multiple work-items simultaneously. Kernel throughput is usually reduced by the largest total number of iterations of nested loops. A large number of threads is usually required to efficiently utilize ND-Range kernels.

For task kernels, the compiler will attempt to pipeline every loop in the kernel to allow multiple iterations of the loop to execute concurrently. If some loops are not pipelined, or not pipelined well, you may not get good performance.

B. Manual and Automatic ND-Range Kernel Replication

ND-Range kernels that are not automatically replicated, using "*num_compute_units*" attribute, and that do not use any work-group information, are by default optimized by the Intel compiler, for higher flexibility and performance. In such cases, the compiler performs hardware optimizations that allow the automatic modification of the local work size to match the global work size on launch. In other words, the compiler creates a kernel capable of processing an arbitrary number of 'single work-item work-groups', in a pipelined manner.

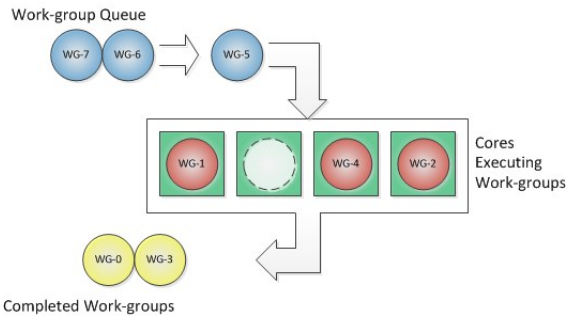

Figure 2. Automatically replicated ND-Range Kernel. Cores represent the multiple compute units.

On the other hand, kernels with multiple compute units, are compiled to behave like an heterogeneous thread pool for work-groups that are created by a kernel call on the host. Each core will pull a work-group off the work-group queue (like a thread pool queue). It will execute the work-group to completion and will then pull another work-group from the

queue. This will continue until all the work-groups for an ND-Range kernel submission are complete.

A disadvantage of this scheme is that the work-group size has to be big enough to better utilize the hardware pipeline and avoid wasted cycles, caused by the pipeline latency, each time a work-group starts/ends its execution, but, at the same time, small enough to avoid unnecessary memory paddings and better utilize the multiple compute units, since Intel notes that "*You should have at least three times as many work-groups as the number of compute units to efficiently utilize all compute units.*" [6]. In other words, it seems that the manual kernel replication is a more straightforward solution, as it comes with less restrictions and makes optimal use of the hardware pipelined resources, however, as we will see in a different section, requires more complex handling and manipulation at the host software side.

C. Comments on Vectorization

Manual vectorization of all the kernels was performed, in order to allow optimal usage of the memory bandwidth through memory coalescing, but also to perform more computations simultaneously, in a SIMD-like manner (Single Instruction Multiple Data). It is important to note that, the algorithmic complexity of some kernels, like BitonicSort, does not allow automatic vectorization, using "*num_simd_work_items*" attribute. In any case manual maximum vectorization of the kernels (int16, float16, etc.) should be considered, although it may increase the kernel programming effort and complexity.

VI. Performance Evaluation

The following picture shows the performance evaluation of the data analytic modules used in the Arria10 integrated FPGA platform from Intel [1]. The figure shows the total execution time in 2 processors (SW only solution) and the total execution time in the HW-accelerated platform using the Arria 10 FPGA and Intel processor integrated into the same chip using the UPI interface. The table shows also the size of the input and the output dataset that it was used for each application. As it is shown in the figure the total speedup ranges from 1.2x to 3.5x depending on the application and the dataset size. The speedup that is measured is the total end-to-end speedup including the communication with the software module that is used to feed the kernels with the data.

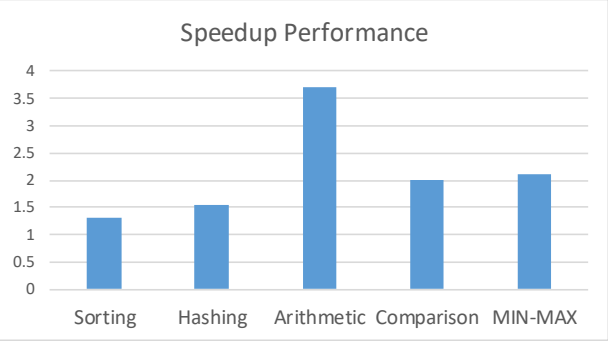

Figure 3. Average speedup depending on the category for the acceleration of the data analytics suite implemented on the HARP.

VII. Conclusions

Data analytics are widely used in several applications such as scientific application, financial applications and marketing. However, as the size of the data that need to be processed keeps increasing significantly, contemporary general-purpose processors cannot process these data in real-time. Hardware accelerators based on FPGAs can be used to speedup significantly data analytics applications and reduce the total execution time. In this paper we have shown a modular approach in which we have developed the basic components that are used in data analytic applications and can be used to construct integrated frameworks for the complete acceleration of data analytics. The speedup for the data analytics varies from 1x to 3.5x for an integrated platform that host both the processor and the FPGA in the same device. The most important conclusion is that for data analytic applications a tight communication mechanism is required between the processor and the accelerators in order to overcome the overhead of the data transfer for these applications. Intel HARP platform and IBM Power8 server with CAPI-FPGA interface such as the ones we have evaluated offer significant speedup. Accelerators platforms that are not tightly coupled to the processors cannot offer significant advantage over other platforms due to the communication overhead.

Acknowledgment

This project has received funding from the European Union's VINEYARD Horizon 2020 research and innovation programme under grant agreement No 687628.

References

[1] S. Windh, X. Ma, R. J. Halstead, P. Budhkar, Z. Luna, O. Hussaini, and W. A. Najjar.2015. High-Level Language Tools for Reconfigurable Computing. Proc. IEEE 103,3 (March 2015), 390–408. https://doi.org/10.1109/JPROC.2015.2399275

[2] David Bacon, Rodric Rabbah, and Sunil Shukla. 2013. FPGA Programming for the Masses. Queue 11, 2, Article 40 (Feb. 2013), 13 pages. https://doi.org/10.1145/2436696.2443836

[3] O. Segal, M. Margala, S. R. Chalamalasetti, and M. Wright. 2014. High level programming framework for FPGAs in the data center. In Field Programmable Logic and Applications (FPL), 2014 24th International Conference on. 1–4. https://doi.org/10.1109/FPL.2014.6927442

[4] Xeon+FPGA Platform for the Data Center, Intel, Inc.

[5] "The VINEYARD approach: Versatile, Integrated, Accelerator-based, Heterogeneous Data Centres" C. Kachris, D. Soudris, G. Gaydadjiev, H. Nguyen, D. S. Nikolopoulos, A. Bilas, C. Strydis, C. Tsalidis, R. Jimenez-Peris, and A. Almeida. International Symposium on Applied Reconfigurable Computing (ARC 2016), March 22-24, 2016, Rio de Janeiro, Brazil

[6] IvyTown Xeon + FPGA: The HARP Program, Intel HARP Inc, 2016 VineTalk: Simplifying Software Access and Sharing of FPGAs in Datacenters

[7] S. Mavridis, M. Pavlidakis, C. Symeonidou, C. Kozanitis, N. Chrysos, A. Bilas, I. Stamoulias, C. Kachris, D. Soudris, VineTalk: Simplifying Software Access and Sharing of FPGAs in Datacenters, IEEE nternational Conference on Field-Programmable Logic and Applications, September, 2017. Ghent Belgium

[8] Intel. [n. d.]. Intel Open Programmable Acceleration Engine (OPAE),. https://01.org/OPAE