Modeling the Scalability of the EuroExa Reconfigurable Accelerators - Preliminary Results*

Invited Paper

Panagiotis Miliadis, Panagiotis Mpakos, Nikela Papadopoulou, Georgios Goumas, and Dionisios Pnevmatikatos

National Technical University of Athens, Greece
{pmiliad,pmpakos,nikela,goumas,pnevmati}@cslab.ece.ntua.gr

Abstract. Current technology and application trends push for both performance and power efficiency. EuroEXA is a project that tries to achieve these goals and push its performance to exascale performance. Towards this objective, EuroEXA node integrate reconfigurable (FPGA) accelerators to offload computational intensive workloads. To fully utilize the FPGA's resource pool, multiple accelerators must be instantiated. System design and dimensioning requires an early performance estimation to evaluate different design options, including using larger FPGA devices, instantiating larger number of accelerator instances, etc.
In this paper, we present the preliminary results of modeling the scalability of EuroEXA reconfigurable accelerators in the FPGA fabric. We start by using simple equations to bound the total number of kernels that can work in parallel depending on the available memory channels and reconfigurable resources. Then, we use a 2^{nd} degree polynomial model to predict the performance benefits of instantiating multiple replicated kernels in a FPGA. The model suggests whether the switching to another larger FPGA is advantageous choice in terms of performance. We verify our results using micro-benchmarks on two state-of-the-art FPGAs; AlveoU50 and AlveoU280.

Keywords: FPGA · FPGA modeling · EuroEXA · reconfigurable accelerators · performance prediction .

1 Introduction

The HPC domain is well known for the gap between the theoretical peak performance of an actual platform and the achieved performance when running real applications. EuroEXA[1] is a project that attempts to reduce this disparity, by enabling -through co-design- an innovative solution that achieves both

* Georgios Goumas, Panagiotis Miliadis, Panagiotis Mpakos, Nikela Papadopoulou, Dionisios Pnevmatikatos

[1] https://euroexa.eu/

extreme data processing and extreme computing. EuroEXA pushes its nodes to exaflop-level performance by implementing a new system architecture that better balances the required computing resources compared to today's systems, supporting the acceleration of key applications. A compute node in EuroEXA assembles general purpose processors, graphic processors units and reconfigurable accelerators.

Current technology and application trends push for both computational performance and power efficiency. A very promising way to achieve both prerequisites is the development of specialized hardware functions. Field-Programmable Gate Arrays (FPGAs) are strong candidates for implementing custom design circuits, as they can be programmed to easily implement a computational datapath suited for a fixed application. The fact that FPGAs are re-programmable, as compared to their ASIC counterparts, offers great flexibility for their integration to larger systems, to support emerging workloads and computational intensive kernels.

State-of-the-art FPGAs are offering a large pool of re-programmable resources, e.g. 6-port LUTs, flip flops, block memories and DSPs, as well rich interconnection between the units. Large banks of memory and processor cores are paired with FPGAs, in order to increase the overall performance of applications. A bitstream with a hardware kernel is offloaded into the FPGA, and a host application sends data and requests to it like a co-processor. In a development environment such as Vitis[2] platform from Xilinx, host applications are usually written in a high-level programming language (e.g. OpenCL), while kernels are written in C++ with HLS primitives. A toolchain converts the high-level kernel into RTL code, and then produces the bitstream with the hardware accelerated design. In EuroEXA, multiple reconfigrable accelerators are instantiated into the same FPGA fabric, to exploit the large pool of resources offered by FPGAs. However, to design the system and the application deployment, the performance benefits of this approach should be gauged. In this paper, we will try to model the scalability of reconfigurable accelerators, and predict the performance benefits acquired by adopting a larger FPGA as compared to a current smaller platform.

Roofline [4] is a model that helps an application developer to classify his computational kernel into two different classes; compute-bound or memory-bound. While a plethora of optimizations can be applied to increase the kernel's performance, it is still bound to the computational capabilities of the processor unit and to the offered memory bandwidth. After a few years, Roofline for FPGAs[2] is introduced, where the authors extended the classic Roofline approach to reconfigurable accelerators. They introduced optimization guidelines to increase the performance of the accelerator and exploit the available resources of FPGA's fabric. However, most of hardware accelerators utilize a fraction of available resources, leaving a large part of fabric unused.

Summarizing, the primary objectives of this work are to:

1. Bound the maximum number of compute units that can be mapped on available reconfigurable resources and memory channels.

[2] `https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html`

2. Create a model that can predict the performance benefits from increasing the total number of compute units.
3. Verify our model on two different FPGA accelerator cards; a smaller AlveoU50 and a larger AlveoU280.

The rest of the paper is organized as follows. In section 2, we present our model for the scalability of reconfigurable accelerators. In this section we present two key parameters that strongly affects our model, FPGA area fabric and available memory channels. Furthermore, we will discuss how these parameters bound the number of kernels that can mapped in a FPGA, and how FPGA modeling can predict the performance benefits from implementing multiple instances of a kernel to a larger or newer FPGA. In section 3, we present our preliminary experimental evaluation of our model, followed by section 4 to finally conclude the paper.

2 FPGA modeling

In this section, we will discuss about our model regarding the scalability of the reconfigurable accelerators in a FPGA. The two key parameters of our discussion are area and memory. The scalability of our model is strongly affected from area, as the resources of a FPGA are limited and we will provide an upper bound of maximum number of kernels that can be mapped. Furthermore, the congestion of memory bandwidth between kernels is another significant issue that may lead to performance degradation. In this section we will discuss and provide an analysis of how congested memory bandwidth can be avoided.

2.1 Scalability limitations due to area congestion

The current generation of FGPAs includes a large pool of reconfigurable resources, which include BRAM, DSPs, LUTs and FFs. The computational kernels designed for FPGAs usually bind a small fraction of the available resources. Newer FPGAs achieve to contain even more reconfigurable resources into a die region, while FPGAs with multiple die regions (SLRs) into the same package are available by the vendors[5]. So, the transition to a newer FPGA leads that the same computational kernel will bind even fewer resources. One of the most straightforward ideas to take advantage of the computational capabilities that a FPGA can offer is to create multiple instances of the same kernel. By implementing multiple instances of a kernel, a host machine can either execute multiple times an algorithm in parallel, or it can dispatch the work items of a single algorithm into the accelerated instances. The replicated instances from now on will be referred as compute units.

A computational kernel needs a fraction of the available resources to instantiate it in a FPGA. The amount of resources that a kernel binds is dependent to optimization decisions of the designer, to maximize his kernel performance. While development decisions play a huge role on the performance capabilities of

a single kernel, they are out of the scope of this paper. In our model, a kernel is considered as a "black box" so as to limit the information that our model needs to evaluate. For area scaling the only piece of information that is needed is kernel's resources which will be referred from now on as {BRAM, DSP, FF, LUT}_design.

The development platforms used by designers, such as Vitis, restrict the utilization of the FPGA area. The suggested maximum resource utilization for a design is restricted to 80% for BRAMs and DSPs resources, while the corresponding ratio for LUTs and FFs is 70%, as reported from the vendors. In our model, we decided to bound the available resources to a more optimistic approach. So the total number of compute units that a FPGA can map is given from Equation 1.

$$\#CU_{area} = \min(\left\lfloor \frac{0.85 * BRAM_{total}}{BRAM_{design}} \right\rfloor, \left\lfloor \frac{0.85 * DSP_{total}}{DSP_{design}} \right\rfloor, \left\lfloor \frac{0.75 * FF_{total}}{FF_{design}} \right\rfloor, \left\lfloor \frac{0.75 * LUT_{total}}{LUT_{design}} \right\rfloor) \quad (1)$$

As shown from the equation, the total number of compute units is restricted by the most consuming resource of the computational kernel, while the floor in the equation offsets the optimistic approach that we took earlier on maximum resource utilization. If other designs occupy a fraction of the total available resources in the FPGA fabric, it is clear that the committed resources must be subtracted from each numerator in Equation 1.

As we will discuss in section 3, the scalability of reconfigurable accelerators in a FPGA platform may be restricted by HLS toolchains, especially when the number of compute units is large enough (i.e. 10-12 compute units). HLS toolchains consider each compute unit to be a distinct building block which consumes reconfigurable resources equal to the original one. So, compute resources increase linearly as more compute units are mapped into the FPGA fabric. When the number of compute units is high enough, the distinct blocks congest over the same wires into the FPGA fabric for routing. When there are not any available wires in the FPGA fabric, or timing requirements cannot be met, the toolchain rejects the design, even though there are available logic resources. This is a limitation in our current prediction model for the scalability of accelerators. In our future work, we will try to model the routing restrictions from HLS toolchains, and bound the total number of compute units in the FPGA fabric depending on the routing complexity as well available resources.

2.2 Scalability limitations due to memory congestion

Another key parameter that strongly affects the scalability of the reconfigurable accelerators is memory. Data are fetched from memory banks into compute units through memory channels. The management of memory channels from developers is the main reason for bottlenecks in a application performance. When multiple compute units try to access the same memory bank through the same memory

channel, they are competing for the same memory bandwidth. The congestion of memory bandwidth and the sharing of memory channels can significantly limit the performance and can convert an algorithm from compute-bound to memory-bound, as fewer data are fetched is second to each unit. In this subsection we discuss how memory congestion can be avoided and provide guidance for better memory management between multiple kernels.

The majority of FPGA boards contain large off-chip DDR memories (e.g. 32-48 GB for state-of-the-art devices), which are used for storing large sets of data. Data are initially stored on off-chip memory and then are streamed into compute units for processing. DDR memories usually are separated in 2 to 4 banks, and a same amount of memory channels are used for communication with compute units. As discussed in the previous subsection, the FPGA fabric can fit a large number of kernels, so memory channels must serve multiple compute units concurrently, decreasing the overall performance of the system due to sharing. With the advent of High Bandwidth Memory (HBM), FPGAs are offering a much higher number of memory channels and overall memory BW at the cost of smaller storage. Xilinx states that in state-of-the-art AlveoU50 and AlveoU280, 32 HBM channels are available for communication between memory banks and compute units. As more memory channels are available for data transmission, congestion can be avoided by statically partitioning memory channels to compute units.

By partitioning the memory channels, each memory bank will serve a single compute unit. Performance bottlenecks from sharing are prevented, and a compute unit can utilize the whole available bandwidth from a memory channel. So, a "one-to-many" communication type is suggested to avoid congestion over memory bandwidth, where a compute unit is atomically served either by a single memory bank or by multiple ones concurrently. An alternative solution to prevent the sharing of memory bandwidth is to enqueue work items into compute units in different time periods, but our scope in this paper is that compute units work in parallel to provide peak performance and maximum throughput.

Given the number of memory channels that a compute unit utilizes, the total number of kernels that can work in parallel without performance degradation due to sharing, is given in Equation 2.

$$\#CU_{mem} = \left\lfloor \frac{MemChannels_{Avail}}{MemChannels_{design}} \right\rfloor \tag{2}$$

2.3 FPGA performance modeling

In the previous subsections, we provided simple equations, and discussed how our model can extract the total number of compute units that a FPGA can map given a certain amount of logic and memory resources. Equation 1 provides an upper bound of compute Units due to limitation in the FPGA fabric, while Equation 2 an upper bound due to limited memory channels. The ideal number of compute units that can work in parallel without performance loss is given by

Equation 3. However, the performance benefits from the transition to a larger or newer FPGA are still unclear.

$$\#CUs_{ideal} = \min(\#CU_{area}, \#CU_{mem}) \tag{3}$$

Our approach for modeling the scalability of the reconfigurable accelerators is to consider the computational kernel as a "black box", where the development decisions are unknown, and minimal information about the kernel is needed for modeling. During the transition to a newer or larger FPGA, the computational kernel is not subject to modifications. If no architectural changes are made from generation to generation or from FPGA to FPGA, such as LUTs or DSPs, almost the same amount of resources are needed to implement and map the same computational kernel. So, as long as more compute units can fit in a FPGA, the performance have to keep increasing linearly. However, we expect reduced performance growth, because as it was mentioned the frequency of compute units is decreased when more compute units are mapped, while an extra software overhead is introduced in order to enqueue work items into the accelerated kernels.

The performance prediction model for FPGAs can be created by following a series of small steps. At first, a scattered graph is created by extracting performance results from an initial FPGA platform. At least two performance points are needed to create a simple model, when a single kernel is mapped in the FPGA fabric and the maximum number respectively. To further increase the accuracy of the prediction model, we recommend inserting more performance points from the initial platform, for different number of compute units. From the performance points of the initial used FPGA platform, a 2^{nd} degree polynomial model is exported. 2^{nd} degree polynomial models have been adopted by other similar works[1], to create prediction models for general purpose processors units. With the help of the prediction model, an application developer can find out the performance benefits from the transition to another FPGA. By integrating a newer FPGA in a system, a larger pool of reconfigurable resources or more memory channels are available. By using our models' equations, the total numbers of compute units can be extracted for the new FPGA platform, and from the 2^{nd} degree polynomial model, the performance benefits can be found out by implementing more compute units.

3 Preliminary Experimental Results

In this section, we will present our preliminary results regarding the FPGA modeling on reconfigurable accelerators. For our case study, we use two state-of-the-art FPGAs platforms: AlveoU50 and Alveo280, while their available resources are listed in Table 1. For the application development, Vitis 2020.2 unified software platform is used, the kernel was written in C++ with HLS primitives, and the host side uses OpenCL to enqueue work items to hardware kernels and to transfer data between the host machine and the FPGA. Three micro-benchmarks are used; Conv2D, MatrixMult and Sequential Read/Write. The first two are used to evaluate the performance capabilities of our platform, as compute units are keep

increasing, while the latter one is used to evaluate the communication between HBM channels and kernels.

FPGA	BRAM	DSP	LUTs(K)	Registers(K)	Mem BW(GB/s)
Alveo U50[3]	1344	5,952	872	1,743	316
Alveo U280[4]	2,016	9,024	1,304	2,607	460

Table 1. Available Resources of AlveoU50 and AlveoU280 and their maximum memory bandwidth.

3.1 Sequential read/write

To avoid memory bandwidth congestion between multiple compute units, we assumed to statically partition memory channels to kernels. At first, we need to evaluate our decision by finding out the potential drawbacks of this choice. In Table 2, we present our results for two communication patterns, by using Sequential Read and Write. In Sequential Read/Write, data are streamed into a compute unit, and then are streamed out to off-chip memory again. One-to-all communication is when a compute unit utilizes all available memory channels, while one-to-one communication is the worst case scenario where each compute unit utilizes only one memory channel. From our results, the static partitioning of memory channels does not introduce any significant overhead in our micro-benchmarks, and almost the entire memory bandwidth can be exploited. Our results come to an agreement with a recent paper that evaluates the HBM channels of Alveo devices, [3]. Our results confirm that congestion of memory bandwidth can be easily avoided by using partition, and the number of compute units can be bounded by the available memory channels.

FPGA	One-to-all	One-to-One	$Channels_{avail}$
AlveoU50	309.97 GB/s	307.31 GB/s	24
AlveoU280	388.82 GB/s	386.07 GB/s	30

Table 2. Available Memory channels and memory bandwidth for a) one-to-all communication and b) one-to-one communication.

3.2 Scalability of Accelerators

To model the scalability of reconfigurable accelerators in the FPGA fabric, we use two computational kernels with different kernel sizes, Conv2D and MatrixMmult.

[3] `https://www.xilinx.com/products/boards-and-kits/alveo/u50.html`
[4] `https://www.xilinx.com/products/boards-and-kits/alveo/u280.html`

As the original compute units are considered "black boxes", the only pieces of information that we need for modeling are the necessary design's resources and the number of channels that utilizes. Table 3 reports the information that our model needs to find out the total number of compute units.

Kernel	Kernel Size	BRAM	DSP	LUTs	Registers	$Channels_{design}$
Conv2D	120	131	43	18,766	23,423	1
	150	227	88	13,200	14,980	1
MatrixMult	80	31	459	15,782	23,129	1
	100	47	602	51,375	58,087	1

Table 3. Resource utilization for our micro-benchmarks when a single compute unit is mapped

We use two FPGA platforms, AlveoU50 as the initial FPGA where our model will be created, and AlveoU280 to verify our perfomance results. At first, our model exports the total number of compute units that AlveoU50 can map in its fabric based on designer's computational kernel. For Conv2D the most costly resource is BRAM while for MatrixMult is DSP. From Equation 1, our model calculates the number of compute units and in Figure 1 and Figure 2 we report our results. For Conv2D the total number of compute units is 8 and 4 for kernel sizes 120 and 150 respectively, while for MatrixMult is 11 and 8 for 80 and 100 kernel sizes. The results verify our equations, as we cannot map any more compute units. Meanwhile, we report the speedup as we increase the number of compute units, by using as baseline the execution time of a single kernel in AlveoU50.

For all kernels, the speedup (Blue Triangles and Red X Marks on all Figures) does not scale linearly as more compute units are instantiated in the FPGA fabric. The loss in performance is the result of the reduced frequency. As more compute units are implemented in the FPGA, the reduction in frequency is getting bigger, until the speedup is yielded around a value. This is the result of the increased latency in data transfers, as data have to cross a larger area until their destination. Fewer data are fetched each second in the compute units, which increase the overall execution time of a kernel. Our model captures the decreasing frequency from instantiating multiple compute units, and predicts the potential drawbacks in performance from implementing a high number of computational kernels.

To export the 2^{nd} degree polynomial model, the execution time is needed for multiple number of compute units. The minimum performance points needed are for a single compute unit and for the maximum numbers respectively. To further increase the accuracy of the model, more performance points can be included. The model for each type of kernel is printed with a light blue line in Figure 1 and Figure 2. The choice of the 2^{nd} degree polynomial model is made to capture the reduced frequency on compute units as they keep increasing, and it is more adapting based on our results. By using the exported model, the potential

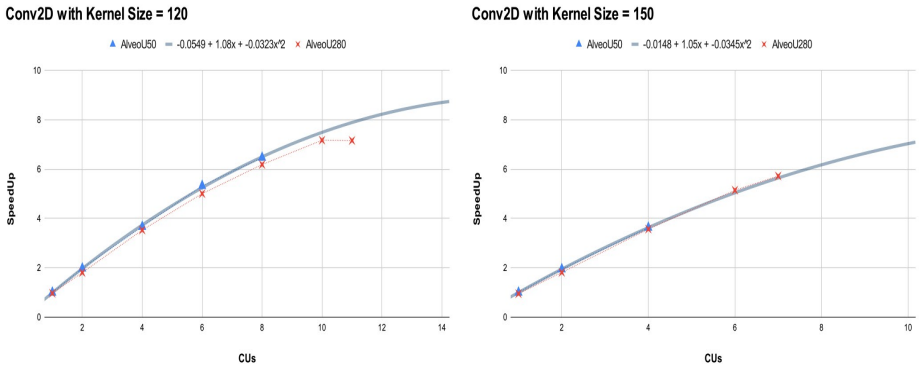

Fig. 1. Modeling the Scalability of Conv2D for Kernel Size= 120 (Left) and Kernel Size=150 (Right). From AlveoU50 performance points (Blue Triangle), we exported the performance model of the kernel (Light Blue Line) and we verified our model with AlveoU280 performance points (Red X Mark).

speedup can be predicted from using a larger FPGA with either more resources in its fabric or memory channels.

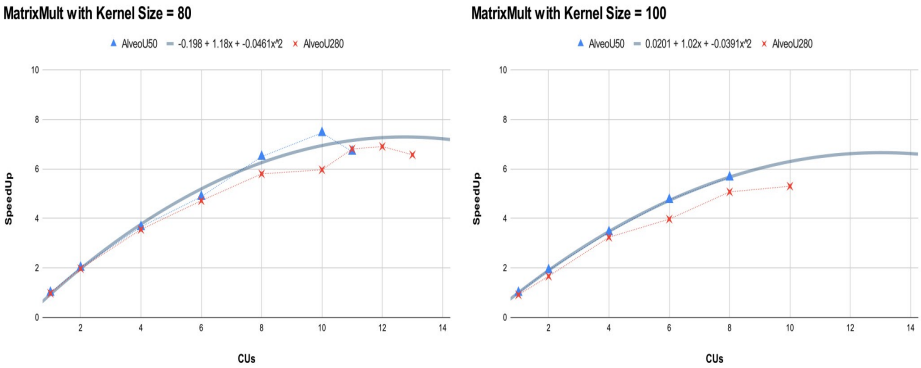

Fig. 2. Modeling the Scalability of MatrixMult for Kernel Size= 80 (Left) and Kernel Size=100 (Right). From AlveoU50 performance points (Blue Triangle), we exported the performance model of the kernel (Light Blue Line) and we verified our model with AlveoU280 performance points (Red X Mark).

One of our objectives in this paper is to predict the performance benefits from implementing the same computational kernel to a larger FPGA. For this case study AlveoU280 is used, which has almost 35% more available reconfigurable resources compared to AlveoU50. As Figure 1 and Figure 2 reports, the maximum number of compute kernels are increased on all micro-benchmarks we used. However, the increase differs from kernel to kernel. Conv2D indeed benefits

the most from the transition to a larger FPGA, as the number of compute units are capped in the FPGA fabric. However, MatrixMult cannot take full advantage of the more available resources, as the high routing complexity prevents the implementation of a high number of computational kernels due to congestion over the same wires. The wiring congestion prevents the full utilization of the FPGA fabric, and it is a limit in our current prediction model.

The final step is to verify the exported 2nd degree polynomial model from the initial platform. As more compute units can be mapped in our new FPGA, the speedup must increase non-linearly, and our model must be able to capture the increased performance. As reported Figure 1 and Figure 2 from the red scattered marks, AlveoU280 performs slight worse than the model reports. We figured out that the difference in performance is due to the reduced frequency from transferring the kernel from our initial FPGA to AlveoU280. The reduced frequency is observed regardless of the number of kernels, and is the result of the larger distance that data have to cross from memory banks into compute units, which further increases the latency as fewer data are fetched. In our future work, we will try to integrate the changes in frequency from transferring a computational kernel from a FPGA to a smaller or larger FPGA.

4 Conclusion and Future Work

In this paper, we present our preliminary results on modeling the scalability of EuroEXA reconfigurable accelerators. FPGA modeling is necessary to predict the performance benefits by utilizing a newer and larger FPGA. In our model, by using simple equations the total number of compute units can be calculated. The replicated kernels can work in parallel without degradation in performance either due to routing or memory congestion. We presented a performance 2nd degree polynomial model which can predict the speedup by increasing the number of compute units. We verified our results by using as initial FPGA platform the AlveoU50 acceleration card, and we tried to predict the speedup gains from using a larger FPGA platform, AlveoU280.

Our future work includes two points. The first one is to include the differences in frequency from implementing a kernel to another FPGA, as our current prediction model is proved slightly more optimistic about the performance benefits. Furthermore, high routing complexity limits toolchains to integrate a high number of compute units (i.e. 12-14) into the FPGA fabric, even though there are available logic resources. Consequently, the second point is to model the routing restrictions from congestion over FPGA wires in fabric to increase the accuracy of our equations.

5 Acknowledgments

This work is supported and funded by the European Commission under the H2020 Programme and the EuroEXA project (Grant Agreement no 754337).

The authors would like to thank Xilinx for their donation of FPGA Alveo development boards.

References

1. Alexandru Calotoiu, David Beckinsale, Christopher W Earl, Torsten Hoefler, Ian Karlin, Martin Schulz, and Felix Wolf. Fast multi-parameter performance modeling. In *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 172–181. IEEE, 2016.
2. Bruno da Silva, An Braeken, Erik H. D'Hollander, and Abdellah Touhafi. Performance modeling for fpgas: Extending the roofline model with high-level synthesis tools. *Int. J. Reconfig. Comput.*, 2013, January 2013.
3. Young kyu Choi, Yuze Chi, Jie Wang, Licheng Guo, and Jason Cong. When hls meets fpga hbm: Benchmarking and bandwidth optimization, 2020.
4. Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, April 2009.
5. Xilinx. Alveo u50 data center accelerator card data sheet. `https://www.xilinx.com/support/documentation/data_sheets/ds965-u50.pdf`.