Enabling An Isolated And Energy-Aware Deployment of Computationally Intensive Kernels on Multi-Tenant Environments *

Argyris Kokkinis[1[0000−0001−5242−0523]], Annastasios Nanos[2[0000−0002−1924−7025]], and Kostas Siozios[1[0000−0002−0285−2202]]

[1] Department of Physics, Aristotle University of Thessaloniki
Thessaloniki, Greece
{arkokkin, ksiop}@auth.gr
http://users.auth.gr/ksiop
[2] Nubificus LTD, Sheffield, United Kingdom
ananos@nubificus.co.uk

Abstract. Nowadays, hardware acceleration can be used as a service for maximizing the applications' performance and achieve significant speedup in time-critical scenarios. FPGA devices inherently consume less power than GPUs and HPC systems and are candidate solutions for performing low-energy yet high-performance computations. However, hardware acceleration services require a private, isolated and flexible execution of the accelerators in multi-tenant environments without compromising the platform's energy and performance efficiency. In this paper we aim to address this issue by proving an end-to-end methodology for the generation, virtualization and deployment of High-Level Synthesis accelerators in multi-tenant environments. We leverage approximate computing techniques and utilize the vAccel framework. Our proposed methodology was evaluated on the Xilinx Alveo U50 acceleration card, achieving energy savings up to 5.2x compared to the initial non energy optimized and non virtualized designs.

Keywords: FPGA · virtualization · High-Level Synthesis · Approximate Computing.

1 Introduction

The advancements in the cloud computing have led to the adoption of the Everything-as-a-Service "XaaS" paradigm [1]. In this computing era hardware resources are shared among users in a multi-tenant model with enhanced requirements for isolated execution and privacy.

Additionally, emerging technologies that target smart systems and time-critical designs require low-power and high-performance computations. FPGA

* This work has been supported by the E.C. funded program SERRANO under H2020 Grant Agreement No: 101017168 (https://ict-serrano.eu/).

devices are known for their design flexibility and can be used for the execution of high-performance accelerators in a small power envelope compared to GPGPUs and HPC platforms [2]. As a result, FPGAs have been recently introduced in cloud infrastructures enabling the Acceleration-as-a-Service "AXaaS" paradigm. Cloud providers such as Amazon and Alibaba have available high-end FPGA acceleration cards in their infrastructures allowing their on-demand usage by the end users [3], [4].

At the same time the design of performance optimized FPGA accelerators for complex applications such as Machine Learning (ML) algorithms has been improved significantly since the adoption of High-Level Synthesis (HLS) techniques [5]. Although, the traditional RTL design flow could deliver energy and performance optimized solutions the current trend that aims to shrink the design's development cycle without sacrificing its performance has led to the adoption of the HLS design flow [5]. The denelopments in HLS have been accompanied with the introduction of the (Multiprocessor System-on-Chip) MPSoC FPGA boards (e.g ZCU104 FPGA) that can deliver for the first time enough computational power at the edge to be utilized for multi-tenancy purposes [6].

In this paper, we propose a framework and a design methodology for the generation and virtualization of HLS accelerators in order to provide energy-aware solutions for both the edge and the cloud. Both of them are part of the SERRANO H2020 project's toolflow. In detail, the introduced solution leverages hardware approximations to minimize the design's energy expenses and utilizes the vAccel framework [7] to provide isolated, private and interoperable solutions without deterioration on the design's performance.

Experimental evaluation shows that we achieve performance and energy gains up to 13.5× and 5.2× respectively due to the hardware-aware approximations compared to the initial non approximate accelerators with a controllable decrease on the applications' quality. The decrease in those gains is negligible after the virtualization of the designs.

The key contributions of this work are:

- We propose to the best of our knowledge the first end-to-end methodology for the design, virtualization and deployment of energy-aware FPGA accelerators.
- We follow a platform-aware hardware optimization methodology to maximize the energy savings.

The rest of this paper is structured as follows. Section 2 introduces the SERRANO H2020 project and its main goals. In section 3 we discuss the hardware approximations, the virtualization technology and the vAccel framework. In section 4 our proposed framework and methodology is described. In section 5 the evaluation results are shown. Finally, section 6 concludes this paper.

2 Overview of the SERRANO project

SERRANO's overall ambition is to introduce a novel ecosystem of cloud-based technologies, spanning from specialized hardware resources up to software toolsets.

This will enable application-specific service instantiation and optimal customizations based on the workloads to be processed, in a holistic manner, thus supporting highly demanding, dynamic and security-critical applications. SERRANO is not only tuned and fully aligned with current trends in the cloud computing sector towards the expansion of cloud infrastructures so as to efficiently integrate edge resources, but it also integrates transparently HPC resources in order to provide an infrastructure that goes beyond the scope of the "normal" cloud and realizes a true computing continuum. SERRANO introduces an abstraction layer that transforms the distributed edge, cloud and HPC resources into a single borderless infrastructure, while it also facilitates their automated and cognitive orchestration. Moreover, SERRANO proposes the introduction and evolution of novel key concepts and approaches that aim to close existing technology gaps, towards the realization of advanced infrastructures, able to meet the stringent requirements of future applications and services. SERRANO will develop technologies and mechanisms related to security and privacy in distributed computing and storage infrastructures, hardware and software acceleration on cloud and edge, cognitive resource orchestration, dynamic data movement and task offloading between edge/cloud/HPC, transparent application deployment, energy-efficiency and real-time and zero-touch adaptability.

In detail, SERRANO provides advancements on several fronts, incorporating and improving several key technologies, in order to boost the development and deployment of novel applications. In particular, SERRANO platform will provide: (i) security and privacy by design in distributed computing and storage infrastructures, (ii) application security and low-latency in multi-tenant environments, (iii) hardware acceleration and energy efficiency in developing, deploying and managing data-intensive applications, (iv) transparent application deployment across seamlessly integrated heterogeneous computing resources in edge, cloud and HPC, (v) data-driven orchestration of network, computation and storage resources as well as of the applications themselves.

SERRANO (see 1) combines transparently and efficiently heterogeneous resources from: (i) edge and fog layers to bring adequate resources close to the end users, (ii) multiple clouds (federated operation) to increase robustness and scaling while reducing dependencies on a single cloud provider and vendor lock-in risks, and (iii) HPC infrastructures to provide enormous capacity for computationally intense and exascale data analysis tasks. By linking edge, fog, cloud and HPC resources, in an automated and self-managed approach, data and processing of extreme low latency services that require immediate action are kept close to where they are produced, while the rest of computationally- and data-intensive applications (e.g. exascale deep learning and analysis, numerical simulations) are intelligently assigned onto a diverse set of cloud and HPC platforms. Through SERRANO's abstraction mechanisms cloud-native application and services are supported towards the cloud continuum.

The main objectives of SERRANO project are summarized, as follows:

- Define an intent-driven paradigm of federated infrastructures consisting of edge, cloud and HPC resources

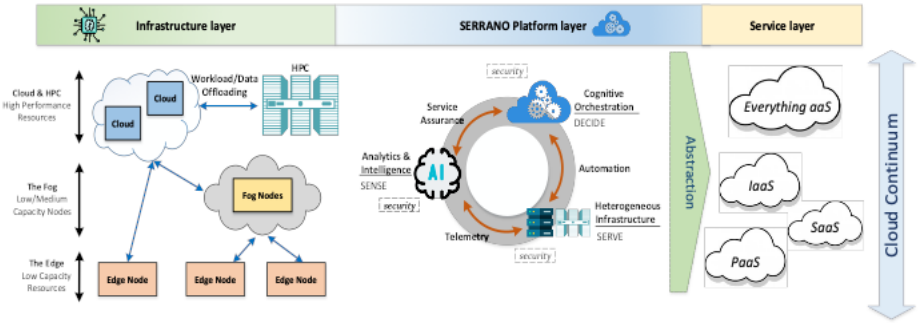

Fig. 1: The overall SERRANO framework.

- Develop security and privacy mechanisms for accelerated encrypted storage over heterogeneous and federated infrastructures
- Provide workload isolation and execution trust on untrusted physical tenders
- Provide acceleration and energy efficiency at the edge and cloud
- Cognitive resource orchestration and transparent application deployment over edge/fog-cloud/HPC infrastructures
- Demonstrate the capabilities of the secure, disaggregated and accelerated SERRANO platform in supporting highly-demanding, dynamic and safety-critical applications

2.1 Challenge for workload isolation and execution trust on untrusted physical tenders

Edge computing brings memory and computing power closer to where it is needed. Even though the cloud computing paradigm seems ideally suited for addressing the increased demand for computation power at the edge, this comes at the expense of an additional abstraction layer that imposes significant overhead to the software stack. Removing this layer to reduce the software overhead, as targeted by SERRANO, immediately exposes another significant challenge: security in multi-tenant environments. SERRANO attacks at the heart of this issue delivering a secure, lightweight, and efficient framework that embraces interoperable microservices in the cloud, the fog and at the edge and providing specific solutions for the low-level software stack, which will enable:

1. Multi-tenancy at the edge using novel virtualization concepts to ensure strict isolation and controlled access to data, while, keeping near-native execution times. This will be achieved by taking advantage of hardware features of the underlying processors.
2. Low-latency communication between applications (management- or compute-related) leveraging a lightweight Virtual Machine Monitor (VMM) that includes the bare minimum software stack needed to complete an I/O request in the virtualization software stack.

3. Near-instant spawn and tear-down times of short-lived applications, by using unikernels featuring ultra-fast boot times, thus breaking the barrier between serverless computing in the cloud and in the edge.

3 Background

3.1 Hardware-Aware Approximation

Approximate computing (AC) methods are used as an alternative to exact computing in the design of error-tolerant FPGA accelerators. AC methodologies employ techniques that trade-off the application's quality to performance and energy gains, aiming to reduce the FPGA platform's power consumption and utilized resources [8].

Typical error-tolerant applications that belong in the Digital Signal Processing (DSP) and ML domains, such as digital filters and Convolutional Neural Networks (CNNs) leverage the precision scaling AC technique to reduce the bit-width of the arithmetic operands and hence minimize the complexity of the implemented multiplication and addition blocks, achieving gains both in performance and in power [9]. In an HLS context precision scaling is applied using fixed precision data types of an arbitrary precision. Those data types are expressed in a format of $< W, I, Q >$ where W specifies the word's size in bits, I the bit-width for the word's integer part and Q the quantization mode.

Additional approximations are performed in the hardware realization of nonlinear functions. The implementation of nonlinear functions, such as trigonometric functions, on FPGAs traditionally leads to a significant utilization of re-configurable resources forming a bottleneck on the design of low-power accelerators [10]. Typical approximations of such functions employ Look-Up Tables (LUTs) for storing the precalculated results or use polynomial approximations. Other suggested methodologies implement nonlinear functions using Chebyshev approximations [11] or perform iterative algorithmic approximations [12]. Typically, the polynomial approximation of trigonometric functions is based on the Taylor series of e^x where the degree of the polynomial defines the level of the applied approximation.

3.2 Virtualization Technology

Hardware accelerators (GPUs, FPGAs, TPUs etc.) are being increasingly adopted in shared computing systems since they have shown to improve performance of diverse computationally intensive workloads, such as ML inference [13], big data analytics [14], data caching [15] and many others. Users running their software on shared infrastructure care about *performance*, *flexibility*, *security* and *interoperability*.

Unlike CPUs that are used in shared systems for over a decade, most of the recent hardware accelerators have not been designed to be shared by multiple, typically untrusted, users. Current solutions impose inflexible setups, add

significant performance overhead, suffer from security issues and lack wide applicability and software portability. The diverse - vendor specific - and complex software stack needed to program and use these devices hampers development of a unified solution [16]. Infrastructure providers ensure secure execution through virtualization techniques, but are still lacking in terms of flexibility and interoperability. True device sharing becomes important in serverless setups [17] where spawned functions are short-lived and response latency / boot times have to be minimal [18]. Deploying applications on virtualization environments at the edge dictates tenants with minimal footprint, removing software stack duplication, to achieve *resource efficiency* and meet hardware constraints.

To tackle secure hardware accelerator sharing, we use vAccel [7],a lightweight framework to run generic acceleration functions on multi-tenant environments. vAccel decouples the accelerated application code from the respective function call. vAccel comprises of the core library, vAccelRT, the user-facing API, and a set of hardware plugins. vAccelRT abstracts any hardware/vendor-specific code, by employing a modular design, where plugins implement bindings for popular acceleration frameworks and the User API exposes a function prototype for each available acceleration function. This way, vAccel-based applications are portable across different platforms without any source code modification. Additionally, using efficient data transport plugins that can be simply loaded at runtime, vAccelRT enables execution on different virtualized or remote targets without application rebuilding.

4 Proposed Framework

Our framework operates in two phases: **(i)** a hardware optimization phase where the accurate HLS accelerators are transformed into equivalent approximate versions that trade-off energy consumption and resource utilization to accuracy and **(ii)** a design virtualization phase where the generated hardware optimized HLS accelerator is virtualized and deployed through the vAccel framework. Figure 2 illustrates the proposed end-to-end methodology.

4.1 Hardware-driven optimization

In the hardware optimization phase our framework receives input an accurate HLS accelerator, a user-defined error margin (denoted as Π_σ at the rest of this paper) that specifies the accelerator's maximum acceptable deviation from the accurate version and a target FPGA platform. The HLS accelerator's full precision data types, the exponential and trigonometric functions are identified and precision scaling and polynomial approximation are performed.

The level of the applied approximations is determined by exploring the design space to find the design solutions that lead to an acceptable error and reduce the estimated resources. From the estimated Pareto solutions, the one that minimizes the resource utilization and satisfies the user-defined error margin is selected as the "best" design.

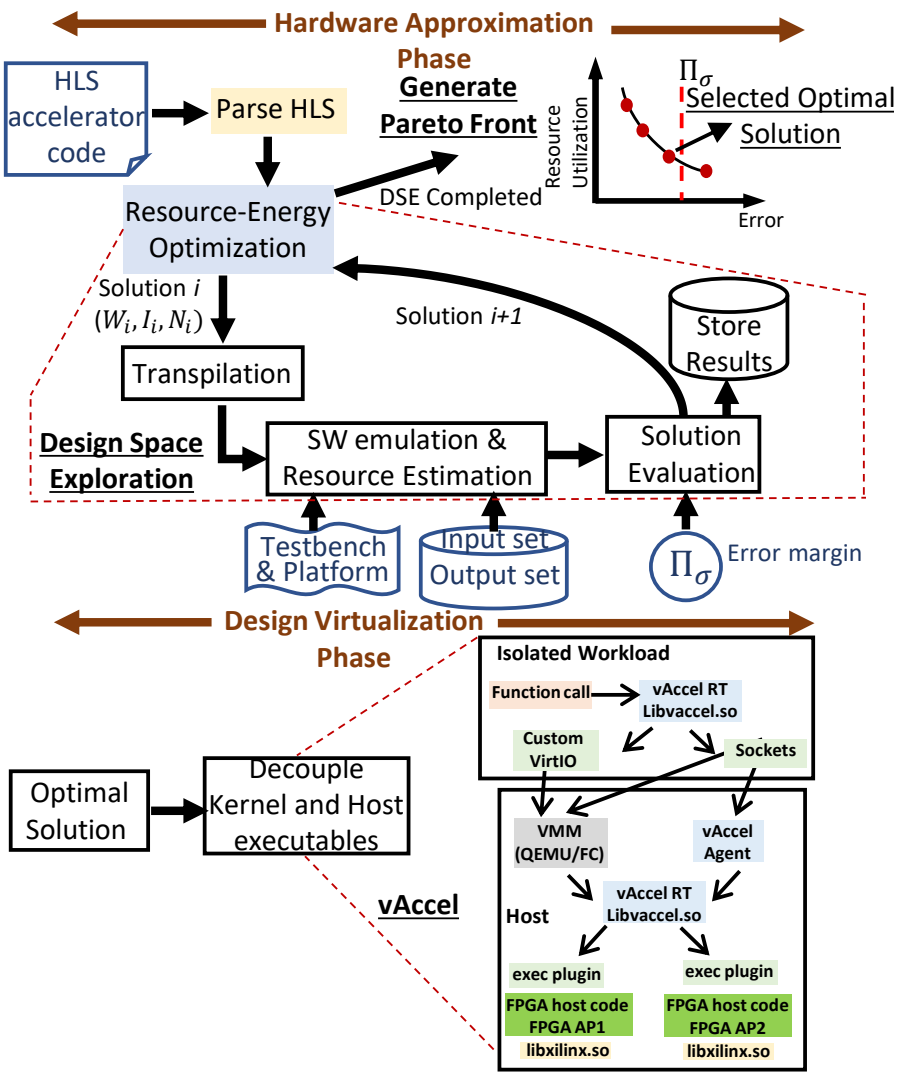

Fig. 2: Our framework for the energy-aware optimization and design virtualization based on the vAccel framework.

In FPGAs there is a positive correlation between the resource utilization and the platform's power consumption, meaning that a resource optimized design is also a design with minimal power expenses.

Design Space Exploration and HLS Transpilation A multi-objective optimization problem is defined to determine the Pareto frontier of the approximate

Table 1: Evaluated algorithms

Algorithm	**Domain**	**Error-Margin Π_σ**	**Energy (J)**
Savitzky-Golay Filter	Digital Signal Processing	50%	0.91
LeNet CNN	Machine Learning	1%	1.6
Black-Scholes	Financial Analysis	5%	0.2

Table 2: Optimal approximate solutions

Algorithm	**W,I,N**	**Error**
Savitzky-Golay Filter	$(7, 6, \text{n/a})$	46%
LeNet CNN	$(8, 3, 26)$	0%
Black-Scholes	$(23, 13, 12)$	4.33%

HLS solutions. Optimal are considered all the solutions that trade-off decrease in resource utilization to decrease in the application's error.

$$\begin{gathered} \min_{W,I,N} \left|\mathrm{Y}_{\text{accurate}} - \mathrm{Y}_{\text{approx.}}(\mathrm{W},\mathrm{I},\mathrm{N})\right| \\ \min_{W,I,N} \text{UtilizedResources}_{\text{approx.}}(\mathrm{W},\mathrm{I},\mathrm{N}), \\ \text{s.t.} \\ Error_{approx.} \leq \Pi_\sigma, \\ \mathrm{I} \leq \mathrm{W} \end{gathered} \tag{1}$$

Where Y in equation 1 denotes the accelerators' output. The problem's exploration parameters are the bit-width of the inputs and the intermediate operands (W), the bit-width for their integer part (I) and the degree of the polynomial that approximates the trigonometric and exponential functions. Noted that the exploration range for the W and I variables is between two to 32 bits, while for the N is between 1 to 100.

For the precision scaling approximation, the truncation quantization mode (Q) is selected in all cases. This is selected because circuits that perform bit truncation can be implemented in hardware with minimal resource overhead compared to rounding circuits and therefore the implemented quantization circuits do not impose significant overhead on the resource optimization task [19].

A linear search is performed on the design space and all the candidate design solutions are evaluated. For every evaluated solution (Q,I,N) HLS transpilation is used to transform the accurate accelerator to its equivalent approximate version. The transpiler receives input the accurate HLS accelerator and the evaluated solution (Q,I,N) and sets the bit precision for the inputs and for the intermediate operands to the corresponding values. Additionally, it replaces all the exponential based functions with a polynomial of N degree. To determine the estimated resource utilization and the approximation induced error, a software emulation is performed for all the approximate generated accelerators.

4.2 Isolated Execution

To achieve interoperable and isolated execution on the hardware accelerator, we employ the vAccel framework. The vAccel runtime library (vAccelRT) exposes two sets of functionalities: a) a primitive, abstract interface to implement acceleration functions (*vAccel plugins*) and b) an extensible, user-facing, user-definable *User API* to consume the available accelerator implementations. To simplify the addition of new functionality and provide a uniform way to write different kinds of plugins, vAccelRT also exposes a *Generic operation*. vAccelRT itself does not interact directly with the accelerator hardware; it provides an abstraction layer to exploit the different acceleration stack options using a common interface.

Abstract interface To efficiently support different platforms/accelerators and diverse multi-tenant setups, we implement vAccelRT's abstract interface as a plugin-based system: Any code interfacing with external libraries/software stacks is packaged as a plugin. This plugin can be written out-of-tree and is built as a shared library linking to any accelerator-specific dependencies, without the need to rebuild or alter the vAccelRT library. The user selects and loads any plugin implementing the required functionality, *at runtime*.

In addition to accelerator integration, vAccelRT plugins are also used for data transport through communication protocols like Unix or TCP sockets. A vAccelRT plugin can either map an "operation" to accelerator code or to a transport mechanism by packing/unpacking application data and using an external communication protocol. vAccel currently supports two plugins of this type for socket-based and virtio-based communication.

Finally, for ease of porting, vAccelRT includes a generic `exec` plugin for use with arbitrary functions packaged in a shared library. The `exec` operation, that is internally mapped to our `exec` plugin, relieves the user from the burden of using or defining a vAccel "operation". Instead, the user only needs a library with the code running on the accelerator and custom functions that call the plugin by passing the library location and handle data packing/unpacking.

5 Experimental Results

Our framework was evaluated on the acceleration of three algorithms that belong in the DSP and ML domains. Namely, those algorithms are: the Savitzky-Golay Filter, the inference of a LeNet CNN on the MNIST dataset and the Black-Scholes mathematical formula. For all three accelerators the selected target platform was the low-power Alveo U50 FPGA acceleration card on a Xeon 5218R server.

Table I summarizes the evaluation algorithms, the examined user defined Π_σ and the accelerators' energy consumption when their initial accurate HLS versions are executed on the target platform. Noted that the selected Π_σ values are typical approximate induced errors in those application domains [20].

To enable a time efficient exploration of the design space during our framework's hardware optimization phase, we initiate the exploration parameters to $(\mathrm{W}, \mathrm{I}, \mathrm{N}) = (32, 1, 100)$. After evaluating this initial solution we set the values of the exploration variables W and N constant and equal to their upper exploration boundaries (i.e 32 and 100) respectively. Following, we increase the value of the variable I by 1. (i.e $(32, \mathrm{I}+1, 100)$).

When the smallest value ($\mathrm{I_o}$) of the variable I that leads to a Π_σ equal to zero has been found, then the minimum number of bits than can represent the integer part without causing an arithmetic overflow are $\mathrm{I_o}$.

Next, we set the variable I constant and equal to $\mathrm{I_o}$, and continue with a linear search with only two variables instead of three. Table II shows the found optimal solutions for the three algorithms that satisfy the error margin constraint and minimize the platform's resource utilization and hence energy consumption.

The generated optimal approximate HLS designs are virtualized, ported through the `vsock` socket and executed on the Xeon server using the `exec` plugin of the vAccel framework.

The barplots in Figure 3 show the energy consumption (Fig. 3a) and execution time speedup (Fig. 3b) of the three HLS accelerators after the resource and energy aware approximation and after their virtualization compared to their initial accurate versions. The baseline for the speedup measurements was the execution time of those algorithms on an Intel Xeon Gold 5218R processor at 2.1GHz.

The approximate HLS designs that are produced from our framework lead to energy savings ranging from 1.7x up to 5.2x for the LeNet and the Black-Scholes accelerators correspondingly compared to the initial versions, while they satisfy the error constraint. Additionally, the accelerator of the LeNet CNN algorithm delivers those energy saving and performance improvements with zero accuracy drop. This is due to the inherent error-tolerance of the applications belonging in the ML domain.

The execution of the virtualized designs through the `vsock` socket cause a negligible decrease in the energy and performance gains compared to the non-virtualized approximate designs that ranges from 2% up to 4%.

6 Conclusion

The isolated execution of FPGA accelerators in a multi-tenant environment requires an efficient in terms of performance and energy consumption virtualization of the hardware designs to enable a private, interoperable and energy-aware deployment of the hardware accelerators. In this paper we propose an end-to-end methodology that utilizes the vAccel framework and leverages hardware approximations to deliver an energy-efficient virtualization of FPGA accelerators. Experimental results demonstrate energy and performance gains up to 5.2x and 13.5x respectively compared to non-approximate and non-virtualized designs.

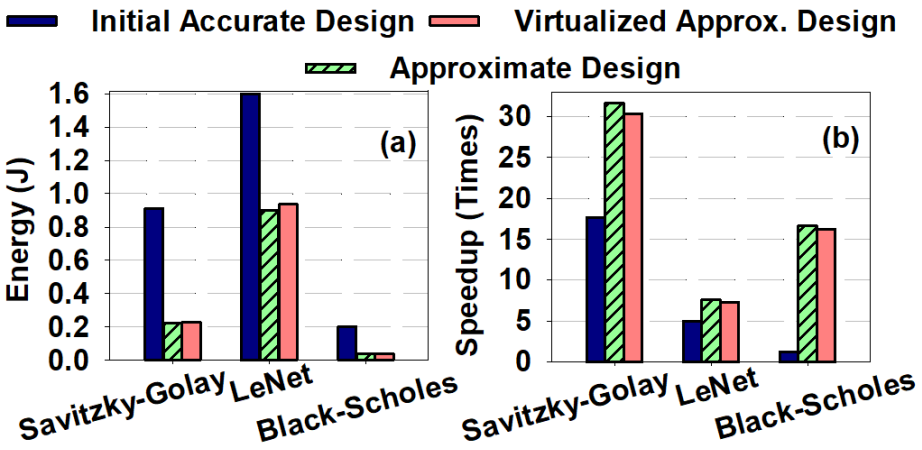

Fig. 3: Accelerators' energy consumption (a) and execution time speedup (b) among the initial, the optimal approximate and the optimal approximate virtualized designs

References

1. D. Yucong *et al.*, "Everything as a service (xaas) on the cloud: Origins, current and future trends," in *2015 IEEE 8th International Conference on Cloud Computing*, 2015.
2. M. Joel, Mandebi *et al.*, "Deploying multi-tenant fpgas within linux-based cloud infrastructure," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 15, no. 2, pp. 1–31, 2022.
3. D. Pellerin, "Fpga accelerated computing using aws f1 instances," 2017. [Online]. Available: https://www.slideshare.net/AmazonWebServices/fpga-accelerated-computing-using-amazon-ec2-f1-instances-cmp308-reinvent-2017
4. A. C. ECS, "Deep dive into alibaba cloud f3 fpga as a service instance," 2020. [Online]. Available: https://www.alibabacloud.com/blog/deep-dive-into-alibaba-cloud-f3-fpga-as-a-service-instances_594057
5. M. W *et al.*, "Towards automatic high-level code deployment on reconfigurable platforms: A survey of high-level synthesis tools and toolchains," *IEEE Access*, vol. 8, pp. 174 692 – 174 722, 2020.
6. S. Piyush *et al.*, "Ai multi-tenancy on edge: Concurrent deep learning model executions and dynamic model placements on edge devices," *arXiv:2107.12486*, 2021.
7. Nubificus LTD, "Interoperable Hardware acceleration for Serverless Computing," 2020. [Online]. Available: https://vaccel.org
8. M. S. Ansari *et al.*, "Improving the accuracy and hardware efficiency of neural networks using approximate multipliers," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 38, no. 2, pp. 317–328, 2019.
9. S. Hashemi *et al.*, "Understanding the impact of precision quantization on the accuracy and energy of neural networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017.

10. L. Yamin and W. Chu, "Implementation of single precision floating point square root on fpgas," in *Proceedings. The 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 1997.
11. P. Pennestrì *et al.*, "A novel approximation scheme for floating-point square root and inverse square root for fpgas," in *11th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, 2022.
12. P. A. Kumar, "Fpga implementation of the trigonometric functions using the cordic algorithm," in *5th International Conference on Advanced Computing Communication Systems (ICACCS)*, 2019.
13. A. Putnam *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014, pp. 13–24.
14. K. Neshatpour *et al.*, "Energy-efficient acceleration of big data analytics applications using fpgas," in *2015 IEEE International Conference on Big Data (Big Data)*, 2015, pp. 115–123.
15. M. Lavasani *et al.*, "An fpga-based in-line accelerator for memcached," *IEEE Computer Architecture Letters*, vol. 13, no. 2, pp. 57–60, 2013.
16. C.-H. Hong *et al.*, "Gpu virtualization and scheduling methods: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–37, 2017.
17. G. I. Amazon, Firecracker, "Offer support for hardware-accelerated inference in firecracker," 2020. [Online]. Available: https://github.com/firecracker-microvm/firecracker/issues/1179
18. E. Jonas *et al.*, "Cloud programming simplified: A berkeley view on serverless computing," *arXiv preprint arXiv:1902.03383*, 2019.
19. T. Lok, Kee *et al.*, "Virtex fpga implementation of a pipelined adaptive lms predictor for electronic support measures receivers," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 1, pp. 86–95, 2005.
20. V. Leon *et al.*, "Exploiting the potential of approximate arithmetic in dsp ai hardware accelerators," in *IEEE 31st International Conference on Field-Programmable Logic and Applications (FPL)*, 2021.