PROCEEDINGS OF SCIENCE

Scalable training on scalable infrastructures for programmable hardware

Dr. Marco Lorusso,[a,b,*] **Prof. Daniele Bonacorsi,**[a,b] **Dr. Riccardo Travaglini,**[b] **Dr. Davide Salomoni,**[d] **Dr. Paolo Veronesi,**[b] **Dr. Diego Michelotto,**[d] **Dr. Mirko Mariotti,**[b,c] **Dr. Giulio Bianchini,**[b,c] **Dr. Alessandro Costantini**[d] **and Dr. Doina Cristina Duma**[d]

[a] *Department of Physics and Astronomy "Augusto Righi", Alma Mater Studiorum - University of Bologna, Viale Berti-Pichat 6/2, Bologna, Italy*

[b] *INFN - National Institute for Nuclear Physics, Viale Berti-Pichat 6/2, Bologna, Italy*

[c] *Department of Physics and Geology, Alma Mater Studiorum - University of Perugia, Via Alessandro Pascoli, Perugia, Italy*

[d] *INFN - CNAF Bologna, Viale Berti-Pichat 6/2, Bologna, Italy*

E-mail: marco.lorusso11@unibo.it, daniele.bonacorsi@unibo.it, riccardo.travaglini@bo.infn.it, davide@infn.it, paolo.veronesi@bo.infn.it, diego.michelotto@cnaf.infn.it, mirko.mariotti@unipg.it, giulio.bianchini@studenti.unipg.it, alessandro.costantini@cnaf.infn.it, cristina.aiftimiei@cnaf.infn.it

Machine learning (ML) and deep learning (DL) techniques are playing an increasingly pervasive and dominant role in High Energy Physics, but this is posing several challenges. Effective computing infrastructures are required for executing AI workflows, and there is a growing demand for training opportunities to upskill users and developers in exploiting programmable hardware such as FPGAs. While there are many training opportunities on generic ML/DL concepts, there is a gap in hands-on tutorials on ML/DL on FPGAs that can cater to a large number of attendants and provide access to a diverse set of hardware with varying specs. This highlights the need for the development of scalable and inclusive training tools to bridge this gap.

INFN-Bologna, the University of Bologna, and INFN-CNAF collaborated on a pilot course on ML/DL on FPGAs, which was successful in paving the way for the creation of a scalable toolkit for future courses. The course used virtual machines, in-house cloud platforms equipped with AMD/Xilinx Alveo FPGA, and Amazon AWS instances for project deployment on FPGAs. Docker containers with full environments for DL frameworks and Jupyter Notebooks were used for interactive exercises.

Finally, the Bond Machine, a software ecosystem that can dynamically generate computer architectures synthesizable in FPGA, is being explored as an alternative for teaching FPGA programming. It offers hardware abstraction, which simplifies interaction with FPGAs and avoids the need to delve into low-level details.

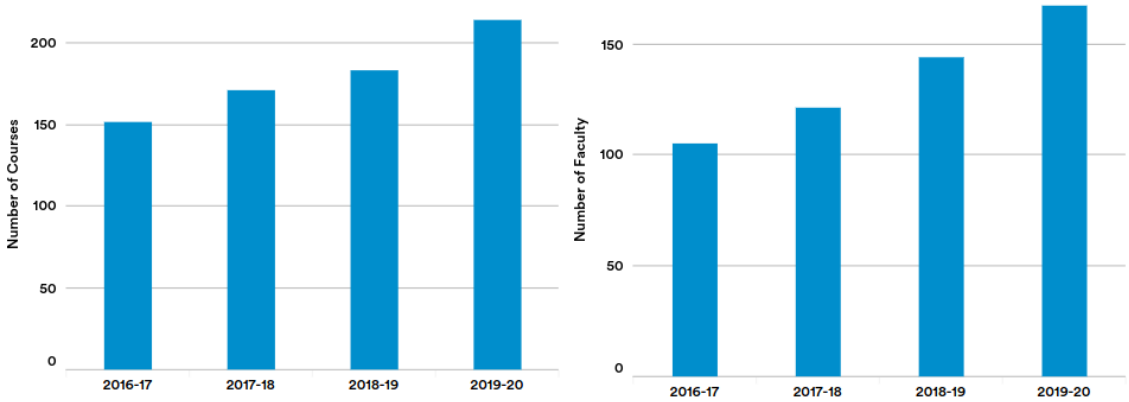
International Symposium on Grids & Clouds (ISGC) 2023 in conjunction with HEPiX Spring 2023 Workshop, ISGC&HEPiX2023
19 - 31 March 2023
Academia Sinica Taipei, Taiwan

[*] Speaker

PoS(ISGC&HEPiX2023)022

1. Status of Machine Learning education

In the last few years Machine Learning has become one of the most popular topic in computer science. This is reflected in the number of educational programs, workshops and courses available to upskill in this sector. In 2020, the AI Index [1], an independent initiative at the Stanford Institute for Human-Centered Artificial Intelligence (HAI), developed a survey that asked computer science departments or schools of computing and informatics at top-ranking universities around the world about four aspects of their AI education: undergraduate program offerings, graduate program offerings, offerings on AI ethics, and faculty expertise and diversity. The survey was completed by 18 universities from 9 countries. Results from the AI Index survey indicate that universities have increased both the number of AI courses they offer that teach students how to build and deploy a practical AI model and the number of AI-focused faculty.



(a) Number of graduate courses that focus on instructing students in the requisite abilities for constructing or deploying a functional AI model, AY 2016-20.

(b) Number of tenure-track faculty who primarily focus their research on AI, AY 2016-20.

Figure 1: Results from a survey that asked top-ranking universities around the world about their AI education offerings

The survey also looks at course offerings at the graduate or advanced degree level, specifically at graduate courses that teach students the skills necessary to build or deploy a practical AI model. These have increased by 41.7% in the last four academic years, from 151 courses in AY 2016–17 to 214 in AY 2019–20 (Figure 1a).

As shown in Figure 1b, the number of tenure-track faculty with a primary research focus on AI at the surveyed universities grew significantly over the past four academic years, in keeping with the rising demand for AI classes and degree programs. The number of AI-focused faculty grew by 59.1%, from 105 in AY 2016–17 to 167 in AY 2019–20.

The Joint Research Center (JRC) at the European Commission assessed the academic offerings of advanced digital skills in 27 European Union member states as well as six other countries: the United Kingdom, Norway, Switzerland, Canada, the United States, and Australia. Figure 2 shows the total number of 1,680 specialized AI programs in all countries considered in the 2019–20 academic year. The United States appears to have offered more programs specialized in AI than

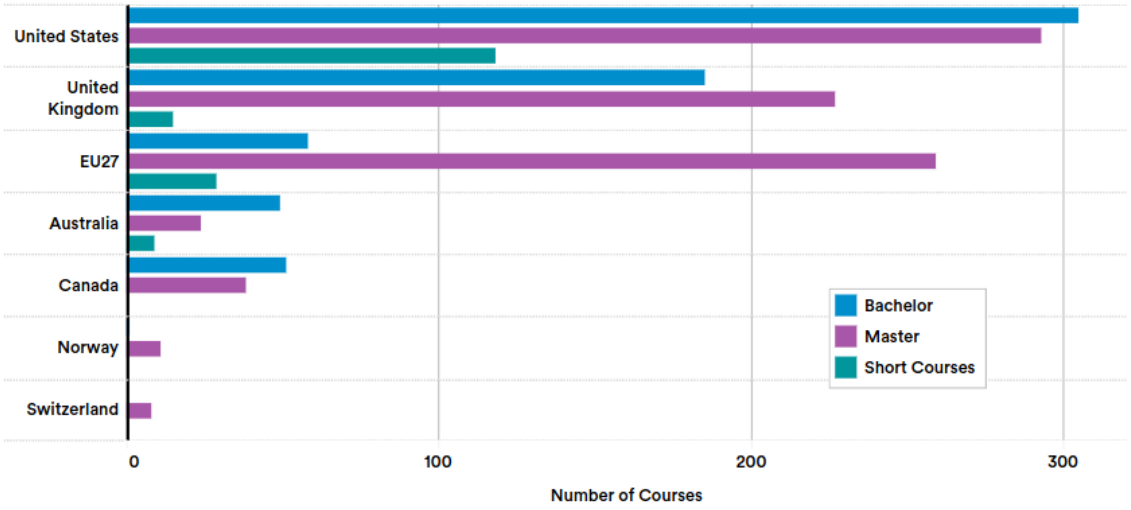


Figure 2: Number of specialized AI programs by geographic area and level, AY 2019-20.

any other geographic area, although the EU comes in a close second in terms of the number of AI-specialized master's programs.

2. FPGAs: Combining Performance and Flexibility through Software and Hardware

In the few decades since field-programmable gate arrays (FPGAs) were introduced, they have radically changed the way digital logic is designed and deployed [2]. By combining the high performance of application-specific integrated circuits (ASICs) and the flexibility of microprocessors, FPGAs have made possible entirely new types of applications. This has helped FPGAs supplanting both ASICs and digital signal processors (DSPs) in some traditional roles. To make the most of this unique combination of performance and flexibility, designers need to be aware of both hardware and software issues. Thus, FPGA users must think not only about the gates needed to perform a computation but also about the software flow that supports the design process. Indeed, FPGAs provide nearly all of the benefits of software flexibility and development models, and nearly all of the benefits of hardware efficiency, but compared to a microprocessor, even though they are typically several orders of magnitude faster and more power efficient, creating efficient programs for them is more complex.

To effectively use FPGAs, it is important for the user to have a foundational understanding of both software and hardware technology. Specifically, on the hardware side, the user should be familiar with digital logic design, including concepts such as gates, multiplexers, flip-flops, and RAM. They should also have a basic understanding of binary number systems and simple logic optimization. Knowledge of hardware description languages such as Verilog or VHDL is also helpful. On the software side, the user should have a basic understanding of computer programming, including simple data structures and algorithms. Overall, the ideal user would have a background that combines electrical engineering, computer science, and computer engineering.

3. The course: *Machine learning techniques with FPGA devices for particle physics experiments*

In Section 1 the sheer amount of new courses about Machine Learning and Artificial Intelligence is evident. However, the same cannot be said when putting together AI and FPGAs. Even though the pros of fusing these two cutting-edge technologies, namely latency and energy consumption, could be very useful in a lot of fields, especially in High Energy Physics [3], there have been few efforts for educating new people on this subject.

Indeed, as can be seen in Figure 3, the range of skills a course should be able to provide to the students is very wide. From Python and very high-level programming to HDL and electronics, and it is a challenge to compress such a vast landscape of notions in a single course or workshop.

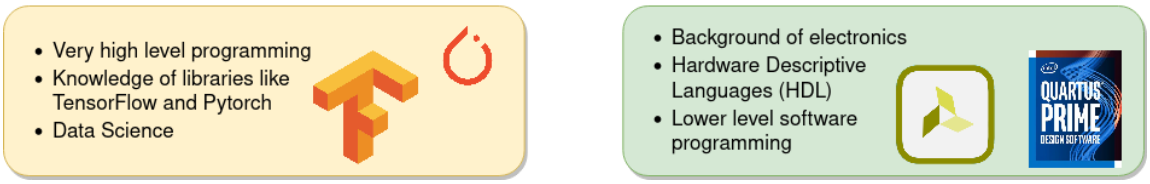


Figure 3: Different set of skills needed to be proficient in both the world of AI and FPGAs.

From the gap in tutorials on ML on FPGAs, the idea of a course called *Machine learning techniques with FPGA devices for particle physics experiments* [4] came up, in order to give a start in understanding and experimenting the various tools that allow the connection between the world of AI and FPGAs.

The course took place from 2^{nd} to 4^{th} November 2022 and it was organized by the Bologna division of the Italian National institute for Nuclear Physics (INFN) with the technical support of CNAF, the main data processing and computing technology research center of INFN. This effort was funded by the INFN Training program. It represented a first step towards a greater focus on education in this field in Italy. The course featured leading international lecturers who are involved in the development of tools to make hardware more approachable at a higher level. The program also received support from the AMD/Xilinx University Program (XUP).

A lot of topics were addressed in the dense two days of lectures and more than half of the duration of the course was spent on tutorials:

- Introduction to efficient use of Machine Learning in HEP;
- Crash course on what FPGAs are;
- *HLS4ML* and how to translate Python to something implementable in hardware (see Section 3.1)
- *Vitis-AI*, the AMD/Xilinx solution to Artifical Intelligence on programmable hardware;
- A new kind of computer architecture (multi-core and heterogeneous) which dynamically adapt to the specific computational problem rather than be static: the *BondMachine* (see Section 3.2)

- How Quartus and Intel make ML on FPGA possible;

In the next two Sections a small description of *hls4ml* and the *BondMachine* is given, as they were two topics of major interest for the high-energy physics community.

3.1 HLS4ML

High-level Synthesis (HLS) [5] is the process of automatic generation of hardware circuit from "behavioral descriptions" contained in a C or C++ program. The target hardware circuit consists of a structural composition of data path, control and memory elements. HLS acts as a bridge between hardware and software domains [6], providing an improvement in productivity for hardware designers who can work at a higher level of abstraction while creating high performance hardware as well as an improvement in system performance for software designers who can accelerate the computationally intensive parts of their algorithms on a new compilation target, i.e. the FPGA.

Using HLS design methodology allows to develop algorithms at the C-level (in programming languages like C and C++) with typically shorter development time. Moreover, it is more easier to validate functional correctness at this level than with traditional HDLs.

The *hls4ml* package [3, 7] was developed by members of the High Energy Physics (HEP) community to translate ML algorithm, built using frameworks like TensorFlow, into HLS code. In this way a trained Neural Network (NN), defined by its architecture, weights, and biases, can be made ready for hardware synthesis with few lines of code. A schematic of a typical workflow is illustrated in Figure 4.

The part of the workflow illustrated in red indicates the usual software workflow required to design a NN for a specific task. The blue section of the workflow is the task of hls4ml, which translates a model into an HLS project that can be synthesized and implemented to run on an FPGA. Some code snippets are shown in the following to explain how an already trained model can be converted into an HLS project using the hls4ml Python API.

Firstly, the model must be loaded:

```
import hls4ml
import tensorflow as tf

model = tf.keras.model.load("model.h5")
```

Then, a *configuration* has to be created:

```
config = hls4ml.utils.config_from_keras_model(model,
                                               granularity = 'name')
```

The `config_from_keras_model()` function returns a Python *dictionary* and takes the following compulsory arguments:

- The Python object containing the NN;
- The *granularity* (`name`, `type` or `model`). This flag sets the level of fineness wanted for the parameter tuning. By using `name`, it is possible to configure each layer and activation function individually. While, `type` is used if the developer wants to share the configuration between all layers of the same type. And finally, with `model` a single configuration is used for the entire model.

By modifying the configuration dictionary it is possible to change the arithmetic precision used for weights, biases and results. After the configuration, the model can be converted by specifying,

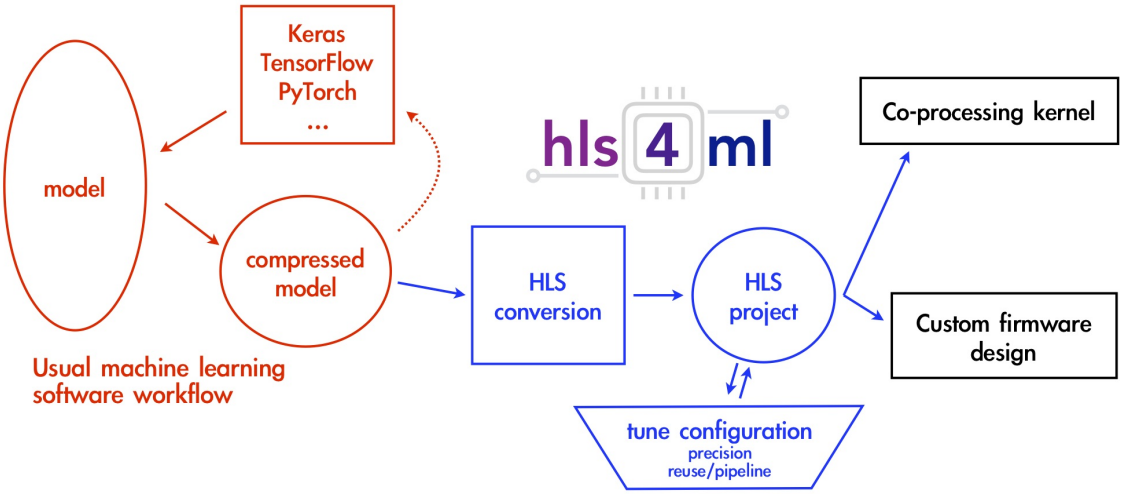


Figure 4: A typical workflow to translate a model into an FPGA implementation using hls4ml.

other than the model object and configuration, the output folder and the FPGA model where the project will be implemented:

```
hls_model = hls4ml.converters.convert_from_keras_model(model, hls_config=config
    , output_dir='HLS_Project', fpga_part='xczu9eg-ffvb1156-2-e')
```

Now, typing `hls_model.compile()`, the `hls_model` can be compiled, i.e. scripts for Vivado HLS [6] are generated containing the instructions for synthesizing the model with the provided device as target hardware. It is possible to synthesize the project inside a Python session with the `hls_model.build()` function.

It is clear by the couple of lines of code shown, how it is easy to create the HLS project, making it feasible also for people who are not expert in FPGAs or hardware in general. Indeed, the goal of the hls4ml package is to empower a HEP physicist to accelerate ML algorithms using FPGAs, thanks to its tools for ML models conversion into HLS. Indeed, hls4ml makes the translation of Python objects into HLS, and its synthesis, parts of an automatic workflow, allowing fast deployment times also for those who know how to write software, yet are not experts on FPGAs.

3.2 The BondMachine

BondMachine (BM) [8] is an open-source framework that enables the creation of computational systems with co-designed hardware and software. This approach maximizes the use of existing resources in terms of concurrency and heterogeneity. The unique feature of BM is the creation of a dynamic architecture that adapts to the specific problem, rather than being static. The hardware is customized to meet the software requirements, implementing only the necessary processing units, resulting in significant advantages in terms of energy consumption and performance. Furthermore, BM is vendor and board independent, allowing for the creation of clusters of heterogeneous FPGAs to solve one or multiple tasks, following the cloud paradigm.

Compared to the use of Hardware Description Language (HDL) code, BM introduces an architecture abstraction layer with minimal overhead, allowing for the use of a standard computational model. This toolkit makes full use of the main features of Field Programmable Gate Arrays (FPGAs) and can be used as an High-Level tool to generate custom firmware for accelerated computation.

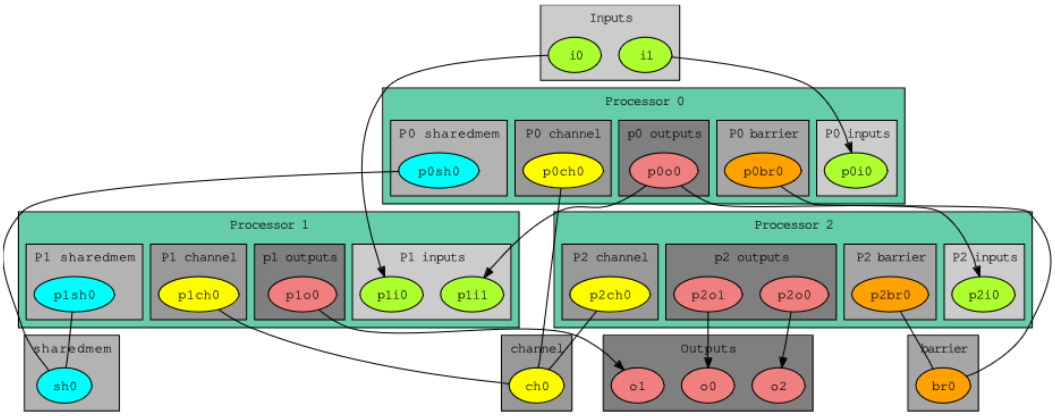


Figure 5: An example of a BondMachine architecture. This specific BM is made of two inputs and tree outputs interconnected between the input/output registers of the processors. Shared objects, such as memory, Channel and Barrier, are connected among the processors.

The BM architecture is particularly suitable for computational models and graphs. The project's flagship activity involves generating firmware with the aim of developing accelerated systems on FPGA to solve different computational problems with a particular focus on machine learning inference [9]. The firmware for accelerated inference generated starting from an high-level trained model with standard machine learning libraries, is highly customizable according to the needs of the specific problem. Different hardware and software optimization techniques have been implemented, starting from the choice of the numerical precision, up to the collapse and pruning of the processors, in order to reduce the resource usage and the energy consumption while improving the inference speed at the same time.

4. A scalable classroom using Cloud Computing

The course aimed to provide an avenue for participants to gain hands-on experience with FPGA technology and the workflows that will be used to create a functional ML design. However, the development of ML algorithms and FPGA firmware requires specific software and libraries, which means a dedicated development machine must be available to attendees. On the other hand, despite the desire to use actual hardware to test the firmware, it is typically not possible for multiple individuals to access FPGAs simultaneously for programming. At the same time it is evident that providing a board for each attendee would be cost-prohibitive and impractical. As a result, the solution was to utilize FPGAs in the cloud.

A system with two different machines was set up (Figure 6): a *Development machine* and a *Deployment machine*.

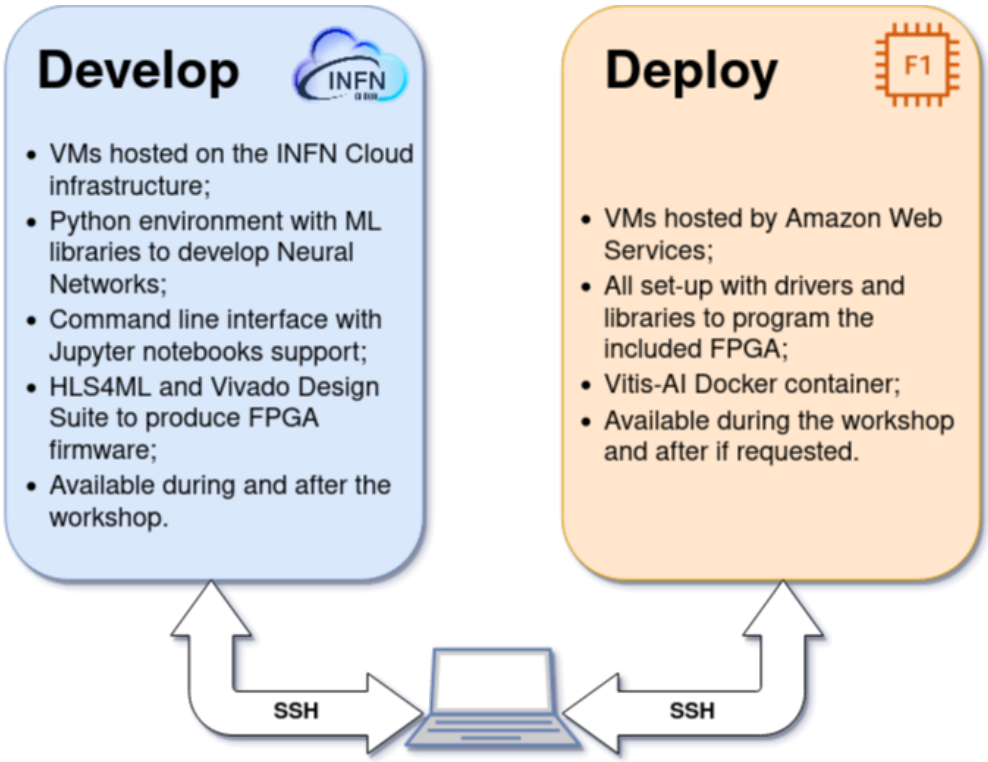


Figure 6: Layout of the two virtual machines made available to each attendee of the course.

The *Development machines* consist in CentOS 7 Virtual Machines (VM) created in the INFN Cloud infrastructure. By utilizing Anaconda [10], a Python environment was made accessible which contained all the necessary tools to manipulate data and construct Neural Networks such as TensorFlow, Keras, QKeras [11] for quantization and optimization, and HLS4ML. To access the machines, SSH with X11 support has been used. The Vivado Design Suite was installed to enable the creation of FPGA firmware, equipped with the relevant libraries to target the available board in the deployment machine. To guarantee remote access to the machines, a public floating IP (FIP) address has been assigned to each VM. In order to let the users play with the available resources and services after the working period of the workshop, they have been maintained for few weeks after the workshop end.

The *Deployment machines* deployed on AWS are EC2 F1 instances [12], equipped with Xilinx FPGA acceleration cards. F1 instances are equipped with tools to develop, simulate, debug, and compile a hardware acceleration code, including an FPGA Developer Amazon Machine Image (AMI) and supporting hardware level development on the cloud. In order to test the Vitis-AI toolkit [13], the Docker Daemon was added to the AMI.

Using F1 instances to deploy hardware accelerations can be useful in many applications to solve complex science, engineering, and business problems that require high bandwidth, enhanced networking, and very high compute capabilities. A variety of target applications can benefit from F1 instance acceleration, including but not limited to genomics, search/analytics, image and video processing, network security, electronic design automation (EDA), image and file compression, and

big data analytics.

F1 instances provide diverse development environments: from low-level hardware developers to software developers who are more comfortable with C/C++ and OpenCL environments. Once an FPGA design is complete, it can be registered as an Amazon FPGA Image (AFI), and deployed to every F1 instance needed.

The course has been used as a test to exploit the potential benefits of a seamless integration between INFN Cloud and a cloud provider like AWS. The proposed sketch of how this integration could work are listed hereafter:

- The user would authenticate themselves on INFN Cloud using a federated authentication system;
- They would then select the type of resource they need, even FPGAs from various vendors;
- If the desired FPGA resource is not available on INFN Cloud, it could be instantiated on AWS transparently;
- The user would be provided with an endpoint to connect to, without the need for a different authentication or interface.

This proof of concept is part of the effort by the people behind INFN Cloud to continuously expand the services that they can offer and keep up with the ever-growing interest in heterogeneous computing.

5. Conclusions

In conclusion, machine learning and deep learning techniques are becoming increasingly important in High Energy Physics, which presents several challenges. To effectively implement AI workflows, there is a need for computing infrastructures, as well as training opportunities to upskill users and developers in using programmable hardware like FPGAs. While there are many training opportunities available, there is a gap in hands-on tutorials for ML/DL on FPGAs that can cater to a large number of attendees and provide access to a diverse set of hardware with varying specifications. To bridge this gap, INFN-Bologna, the University of Bologna, and INFN-CNAF collaborated on a pilot course on ML/DL on FPGAs using virtual machines, in-house cloud platforms, and Amazon AWS instances.

While the course was successful and garnered significant interest, there is still room for improvement. For example, attendees could benefit from more tutorial time and access to the machines before the training begins to better prepare. The lack of established teaching methods for this topic presents an opportunity to test new and more effective teaching techniques, such as inverted learning.

Finally, creating a VM template that includes all the necessary tools for this type of development and publishing an AMI for deployment could streamline the setup process and increase productivity for both educational purposes and research work.

Acknowledgments

We would like to express our gratitude to Dr. Thea Aarrestad (ETH), Dr. Vladimir Loncar (CERN), Dr. Jennifer Ngadiuba (CALTECH), and Dr. Sioni Summers (CERN) for their invaluable support and constructive feedback. Additionally, we would like to acknowledge the financial support provided by the INFN Training Commission and the technical assistance offered by the AMD/Xilinx University Program. Furthermore, we extend our appreciation to Mariella Gangi and Antonella Monducci for their organizational support.

References

[1] D. Zhang, S. Mishra, E. Brynjolfsson, J. Etchemendy, D. Ganguli, B. Grosz et al., *The AI Index 2021 Annual Report*, AI Index Steering Committee, Human-Centered AI Institute, Stanford University (March, 2021).

[2] S. Hauck and A. DeHon, *Reconfigurable computing: the theory and practice of FPGA-based computation*, Systems on Silicon, Morgan Kaufmann (2008).

[3] J. Duarte et al., *Fast inference of deep neural networks in FPGAs for particle physics*, *JINST* **13** (2018) P07027 [1804.06913].

[4] *Machine learning techniques with FPGA devices for particle physics experiments* https://agenda.infn.it/event/15116/.

[5] P. Coussy and A. Morawiec, *High-Level Synthesis: From Algorithm to Digital Circuits* (06, 2008).

[6] Xilinx Inc., *Vivado Design Suite User Guide - High-Level Synthesis*, 2020.

[7] FastML Team, *fastmachinelearning/hls4ml*, Zenodo (2021), 10.5281/zenodo.1201549.

[8] M. Mariotti, D. Magalotti, D. Spiga and L. Storchi, *The bondmachine, a moldable computer architecture*, *Parallel Computing* **109** (2022) 102873.

[9] M. Mariotti, L. Storchi, D. Spiga, D. Salomonie, T. Boccalif and D. Bonacorsid, *The bondmachine toolkit: Enabling machine learning on fpga*, in *International Symposium on Grids & Clouds 2019*, p. 20, 2019.

[10] Anaconda Inc., *Anaconda Data Science Platform* https://www.anaconda.com/.

[11] *QKeras library* https://github.com/google/qkeras.

[12] M. Lorusso, D. Bonacorsi, D. Salomoni and R. Travaglini, *Machine Learning inference using PYNQ environment in a AWS EC2 F1 Instance*, *PoS* **ISGC2022** (2022) 001.

[13] *Xilinx® Vitis™ AI Integrated Development Environment* https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html.