Scalable training on scalable infrastructures for programmable hardware

Marco Lorusso[1,2,*], *Daniele* Bonacorsi[1,2], *Riccardo* Travaglini[2], *Davide* Salomoni[4], *Paolo* Veronesi[2], *Diego* Michelotto[4], *Mirko* Mariotti[2,3], *Giulio* Bianchini[2,3], *Alessandro* Costantini[4], and *Doina Cristina* Duma[4]

[1]Department of Physics and Astronomy "Augusto Righi", Alma Mater Studiorum - University of Bologna, Viale Berti-Pichat 6/2, Bologna, Italy
[2]INFN - National Institute for Nuclear Physics, Viale Berti-Pichat 6/2, Bologna, Italy
[3]Department of Physics and Geology, Alma Mater Studiorum - University of Perugia, Via Alessandro Pascoli, Perugia, Italy
[4]INFN - CNAF Bologna, Viale Berti-Pichat 6/2, Bologna, Italy

Abstract. Machine learning (ML) and deep learning (DL) techniques are increasingly influential in High Energy Physics, necessitating effective computing infrastructures and training opportunities for users and developers, particularly concerning programmable hardware like FPGAs. A gap exists in accessible ML/DL on FPGA tutorials catering to diverse hardware specifications. To bridge this gap, collaborative efforts by INFN-Bologna, the University of Bologna, and INFN-CNAF produced a pilot course using virtual machines, in-house cloud platforms, and AWS instances, utilizing Docker containers for interactive exercises. Additionally, the Bond Machine software ecosystem, capable of generating FPGA-synthesizable computer architectures, is explored as a simplified approach for teaching FPGA programming.

1 Status of Machine Learning education

Machine Learning has gained significant prominence in recent years within the field of computer science. This is evident through the proliferation of educational initiatives, workshops, and courses aimed at enhancing skills in this domain. In 2020, the AI Index [1], an independent effort associated with the Stanford Institute for Human-Centered Artificial Intelligence (HAI), conducted a survey targeting top-ranked universities across the globe. This survey focused on four key aspects of AI education: undergraduate and graduate program offerings, education in AI ethics, and faculty diversity and expertise. The survey received responses from 18 universities spanning 9 countries. The results of the survey indicate a noteworthy increase in the quantity of AI courses offered, concentrating on the practical development and deployment of AI models, as well as an uptick in AI-specialized faculty. The survey also delved into advanced-level courses, specifically those intended for graduate students seeking to acquire the necessary skills for constructing and implementing practical AI models. Over the course of the last four academic years, these offerings saw a substantial 41.7% surge, rising from 151 courses in the 2016–17 academic year to 214 in the 2019–20 academic year.

*e-mail: marco.lorusso11@unibo.it

In response to the escalating demand for AI courses and degree programs, there was a significant rise in the number of tenure-track faculty members with a primary research emphasis on AI at the surveyed universities. The count of AI-focused faculty increased by 59.1%, expanding from 105 individuals in the 2016–17 academic year to 167 in the 2019–20 academic year. The European Commission's Joint Research Center (JRC) evaluated advanced digital skills education across 27 European Union member states and an additional six countries: the United Kingdom, Norway, Switzerland, Canada, the United States, and Australia. The Commission counted a total of 1,680 specialized AI programs across all considered countries during the 2019–20 academic year. Notably, the United States boasted a higher count of specialized AI programs compared to other regions, although the EU closely followed, particularly in terms of AI-specialized master's programs.

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

The introduction of field-programmable gate arrays (FPGAs) has significantly transformed the landscape of digital logic design and deployment [2]. By blending the performance attributes of application-specific integrated circuits (ASICs) with the adaptability of microprocessors, FPGAs have enabled novel applications and even replaced ASICs and digital signal processors (DSPs) in some conventional roles. However, harnessing the potential of FPGAs requires a comprehensive grasp of both hardware and software considerations. This entails not only accounting for the hardware components required for computations but also incorporating the software workflow that facilitates the design process. Although FPGAs offer the advantages of software flexibility and hardware efficiency, achieving optimal results demands a more intricate programming approach compared to microprocessors, despite FPGAs' superior speed and energy efficiency. Effectively leveraging FPGAs necessitates a foundational comprehension of both software and hardware principles. This includes, as depicted in Figure 1, familiarity with digital logic design, hardware description languages like Verilog or VHDL, as well as basic computer programming knowledge encompassing data structures and algorithms. An ideal user profile would integrate expertise in electrical engineering, computer science, and computer engineering. Condensing such a vast array of concepts within a single course or workshop presents a significant challenge.

In Section 1, the abundance of new courses focused on Machine Learning and Artificial Intelligence is evident, yet the same is not observed when considering the integration of AI and FPGAs. Despite the potential benefits of combining these advanced technologies, such as reduced latency and energy consumption, particularly in fields like High Energy Physics [3], there have been limited endeavors to educate individuals in this intersection.

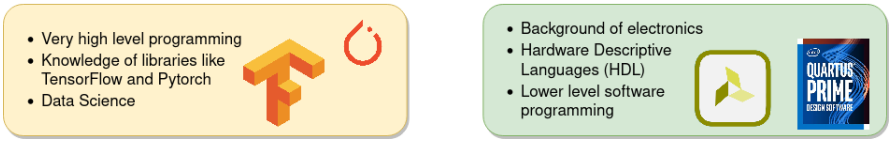

Figure 1. Different set of skills needed to be proficient in both the world of AI and FPGAs.

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 2.1)
- *Vitis-AI*, the AMD/Xilinx solution to Artifical Intelligence on programmable hardware;
- A new kind of computer architecture (multi-core and heterogeneous) which dynamically adapts to the specific computational problem rather than being static: the *BondMachine* (see Section 2.2)
- 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.

2.1 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. 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 typically associated to a shorter development time. Moreover, it is 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 algorithms, 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 2.

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`) determines the desired level of detail for parameter tuning. Opting for `name` enables the configuration of each layer and activation function independently. Conversely, texttttype is employed when a shared configuration is desired for all layers of the same type. Lastly, `model` involves utilizing a single configuration 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

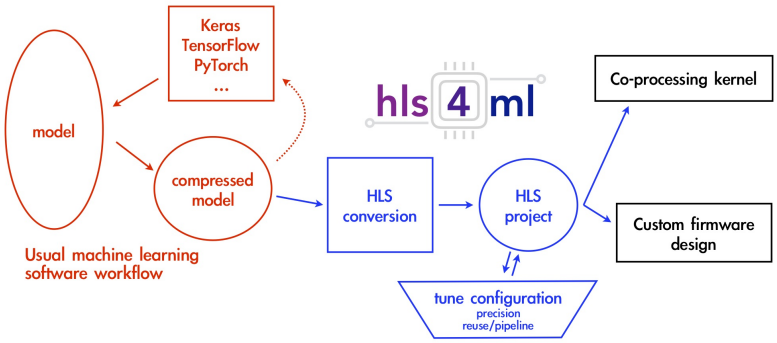

Figure 2. A typical workflow to translate a model into an FPGA implementation using hls4ml.

specifying, 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 from the couple of lines of code shown, how easy it is to create the HLS project, making it feasible also for people who are not experts in FPGAs or hardware. 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.

2.2 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.

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.

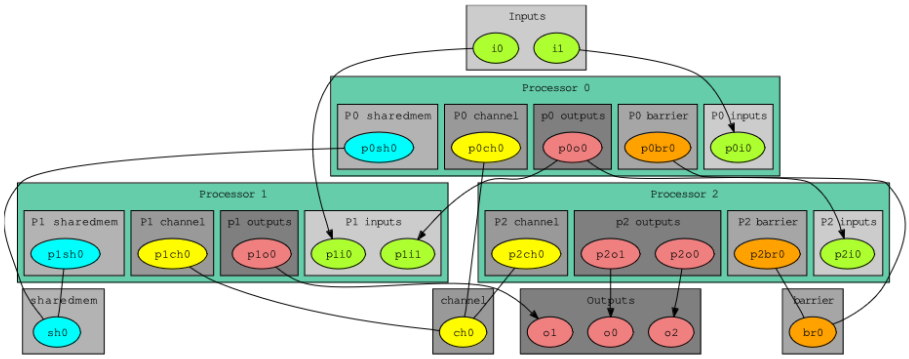

Figure 3. An example of a BondMachine architecture. This specific BM is made of two inputs and three outputs interconnected between the input/output registers of the processors. Shared objects, such as memory, Channel and Barrier, are connected among the processors.

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.

3 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.

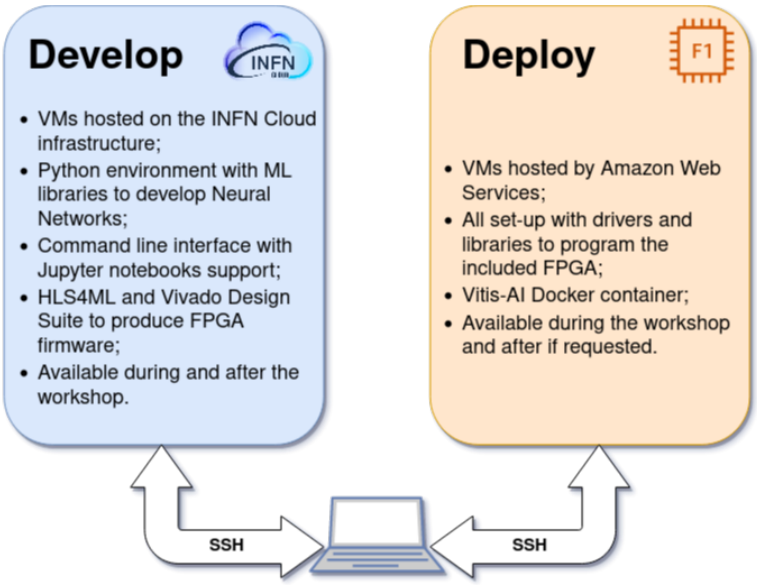

Figure 4. Layout of the two virtual machines made available to each attendee of the course.

A system with two different machines was set up (Figure 4): a *Development machine* and a *Deployment machine*.

The *Development machines* consist in CentOS 7 Virtual Machines (VM) created in the INFN Cloud infrastructure. By utilizing Anaconda[1], a Python environment was made accessible which contained all the necessary tools to manipulate data and construct Neural Networks such as TensorFlow, Keras, QKeras[2] 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 were maintained for a few weeks after the workshop ended.

The *Deployment machines* deployed on AWS are EC2 F1 instances [10], 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[3], 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.

[1] https://www.anaconda.com
[2] https://github.com/google/qkeras
[3] https://www.xilinx.com/products/design-tools/vitis/vitis-ai

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 users would authenticate themselves on the 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.

4 Expanding INFN Cloud Services with HPC Bubbles

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.

Indeed, INFN spearheaded the *Terabit network for Research and Academic Big data in Italy* (TeRABIT) initiative, which is backed by the Italian National Recovery and Resilience Plan (NRRP) [11]. The project's objective revolves around establishing a distributed, highly interconnected hybrid Cloud-HPC computing environment. This entails the integration of the distributed INFN infrastructure with PRACE-Italy's HPC resources, facilitated by a high-speed network provided by the National computer network for universities and research (GARR)

Within this framework, INFN is expanding its INFN Cloud infrastructure by deploying distributed HPC infrastructures known as "HPC Bubbles" These HPC Bubbles encompass various clusters featuring CPUs, CPUs + GPUs, and CPUs + FPGAs, along with swift storage capabilities. The plan encompasses achieving integration both among the distributed HPC bubbles themselves and between these bubbles and the INFN Cloud infrastructure. Moreover, integration is sought between the HPC bubbles and conventional HPC systems, with a notable focus on the Leonardo@CINECA system.

The overarching aim is to establish a scalable "Edge-Cloud Continuum" that leverages AI technologies. This continuum is designed to empower users to flexibly process substantial volumes of big data, offering a dynamic and adaptable approach to data processing.

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, T. Lyons, J. Manyika, J.C. Niebles, M. Sellitto et al., *The AI Index 2021 Annual Report* (AI Index Steering Committee, Human-Centered AI Institute, Stanford University, 2021)

[2] S. Hauck, A. DeHon, *Reconfigurable computing: the theory and practice of FPGA-based computation*, Systems on Silicon (Morgan Kaufmann, 2008), ISBN 9780123705228

[3] J. Duarte et al., JINST **13**, P07027 (2018), 1804.06913

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

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

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

[7] FastML Team, *hls4ml*, https://doi.org/10.5281/zenodo.7933047

[8] M. Mariotti, D. Magalotti, D. Spiga, L. Storchi, Parallel Computing **109**, 102873 (2022)

[9] M. Mariotti, L. Storchi, D. Spiga, D. Salomoni, T. Boccali, D. Bonacorsi, *The BondMachine toolkit: Enabling Machine Learning on FPGA*, in *International Symposium on Grids & Clouds 2019* (2019), p. 20

[10] M. Lorusso, D. Bonacorsi, D. Salomoni, R. Travaglini, PoS **ISGC2022**, 001 (2022)

[11] F. Fanzago, *INFN and the evolution of distributed scientific computing in Italy* https://indico.jlab.org/event/459/contributions/11802/ (CHEP, 2023)