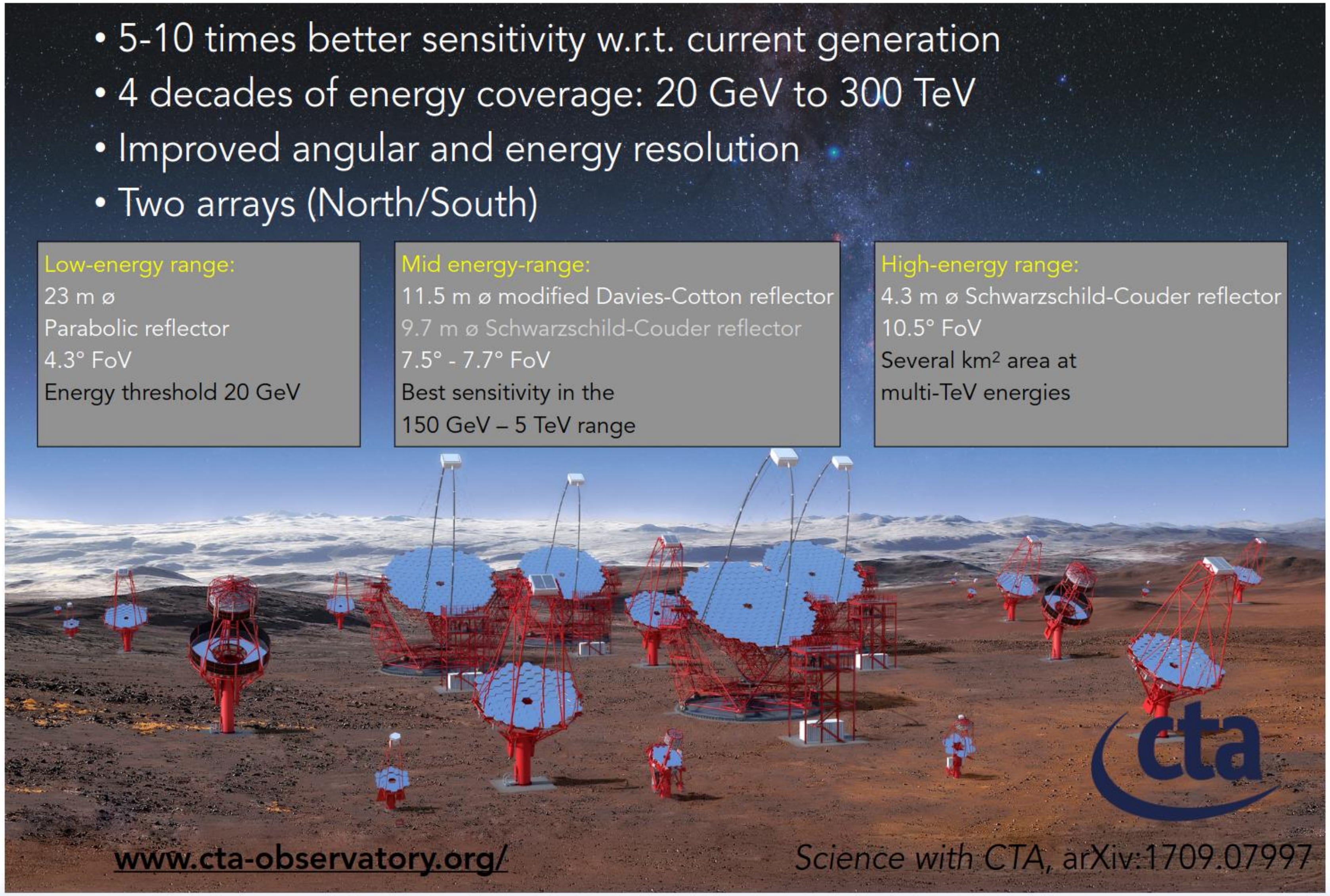
Deep Learning-Based Data Processing in Large-Sized Telescopes of the Cherenkov Telescope Array Observatory: FPGA Implementation

Carlos Abellán Beteta[1], Iaroslava Bezshyiko[1], Nicola Serra[1]

1. University of Zurich

Cherenkov Telescope Array Observatory (CTAO)



- 5-10 times better sensitivity w.r.t. current generation
- 4 decades of energy coverage: 20 GeV to 300 TeV
- Improved angular and energy resolution
- Two arrays (North/South)

Low-energy range:
23 m ø
Parabolic reflector
4.3° FoV
Energy threshold 20 GeV

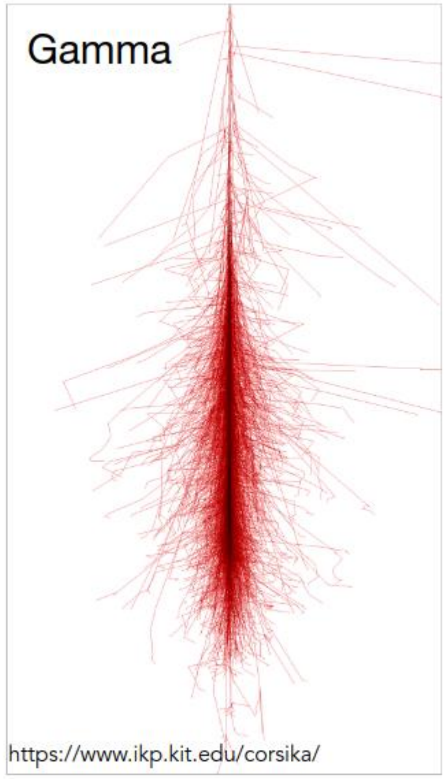
Mid energy-range:
11.5 m ø modified Davies-Cotton reflector
9.7 m ø Schwarzschild-Couder reflector
7.5° - 7.7° FoV
Best sensitivity in the
150 GeV – 5 TeV range

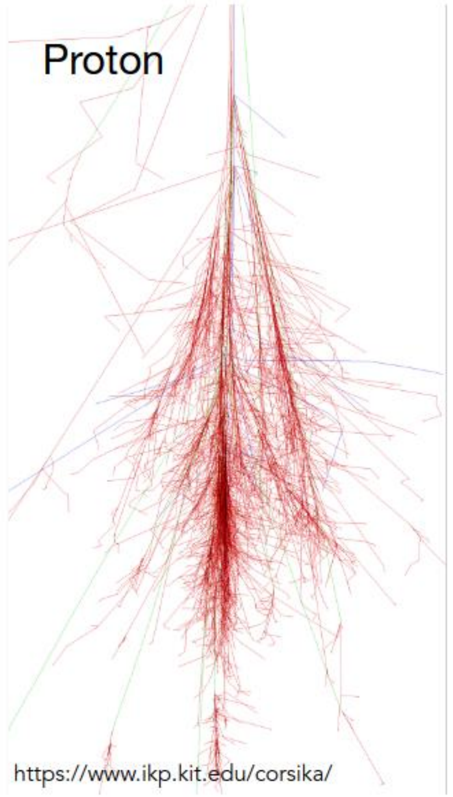
High-energy range:
4.3 m ø Schwarzschild-Couder reflector
10.5° FoV
Several km^2 area at
multi-TeV energies

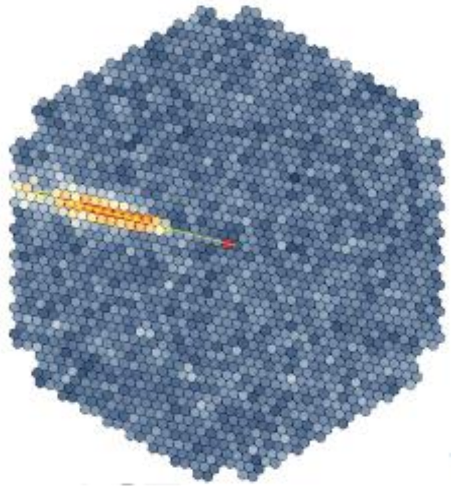
www.cta-observatory.org/

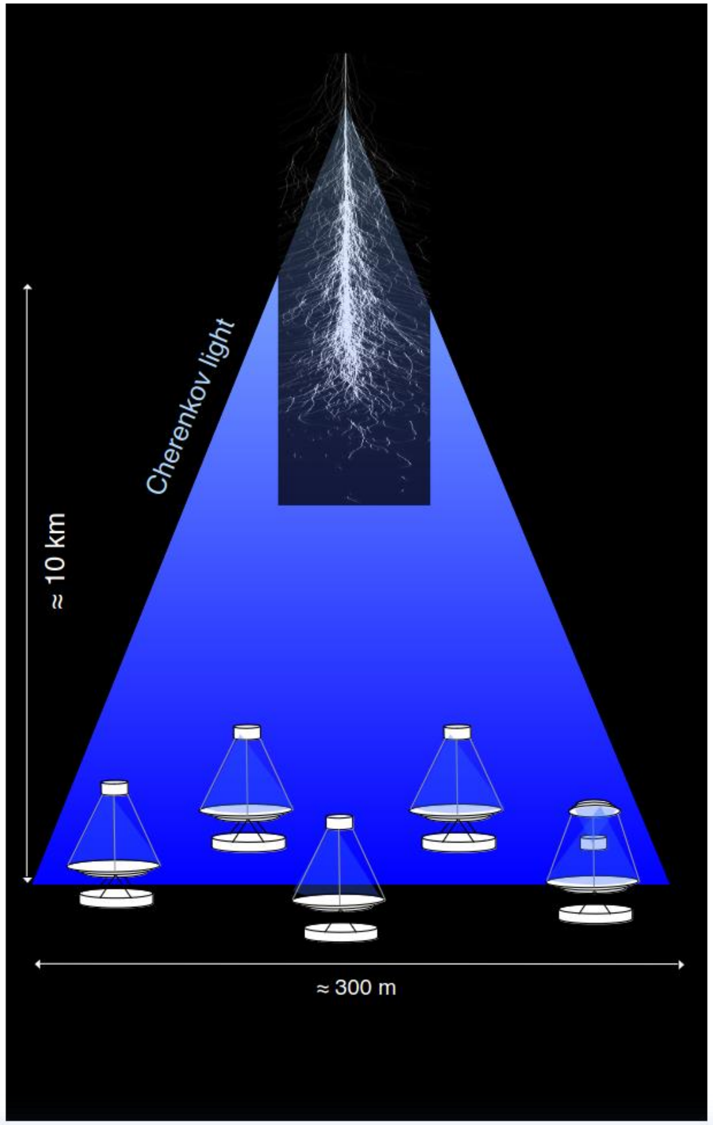
Science with CTA, arXiv:1709.07997

Imaging atmospheric Cherenkov telescope

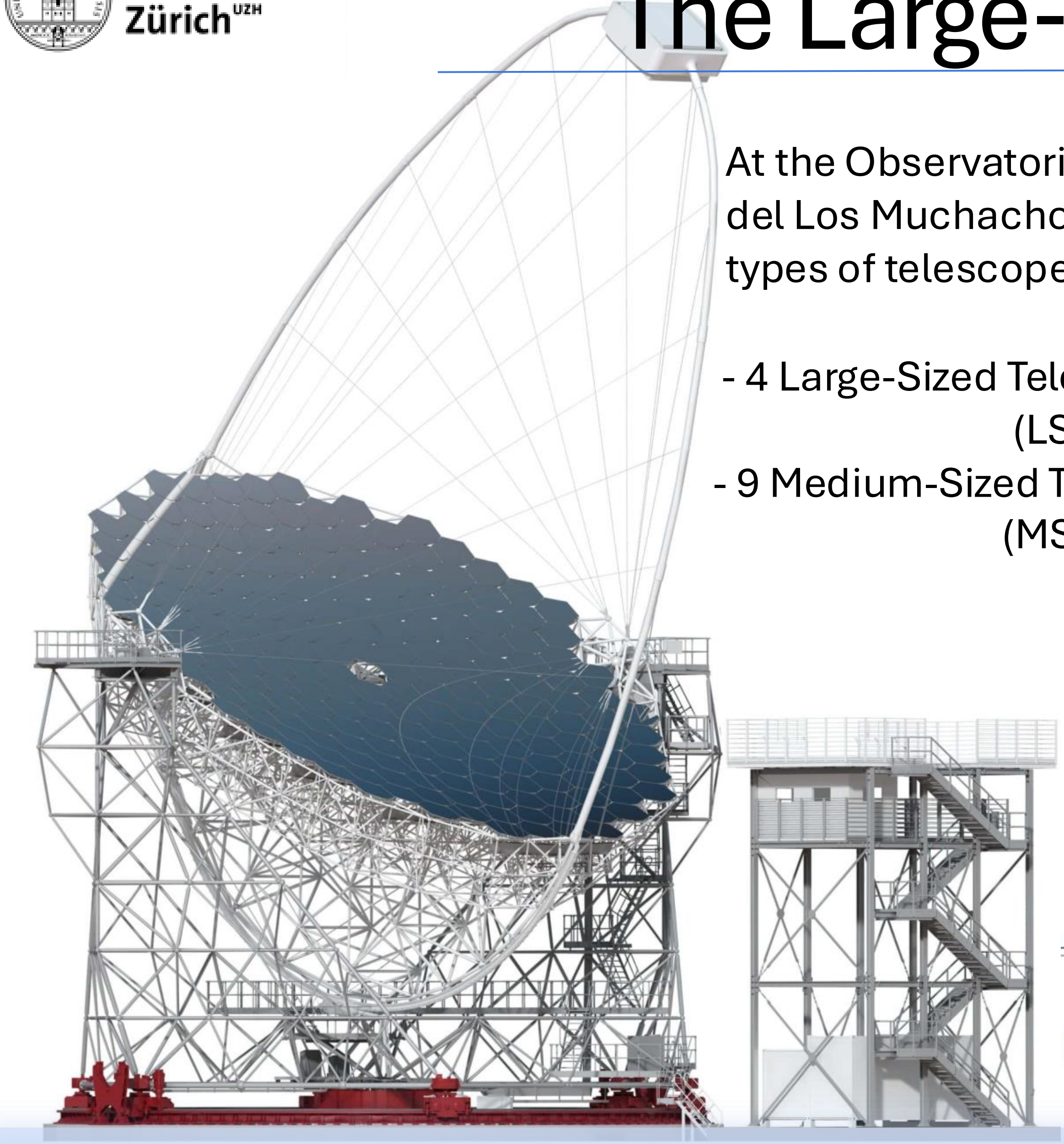






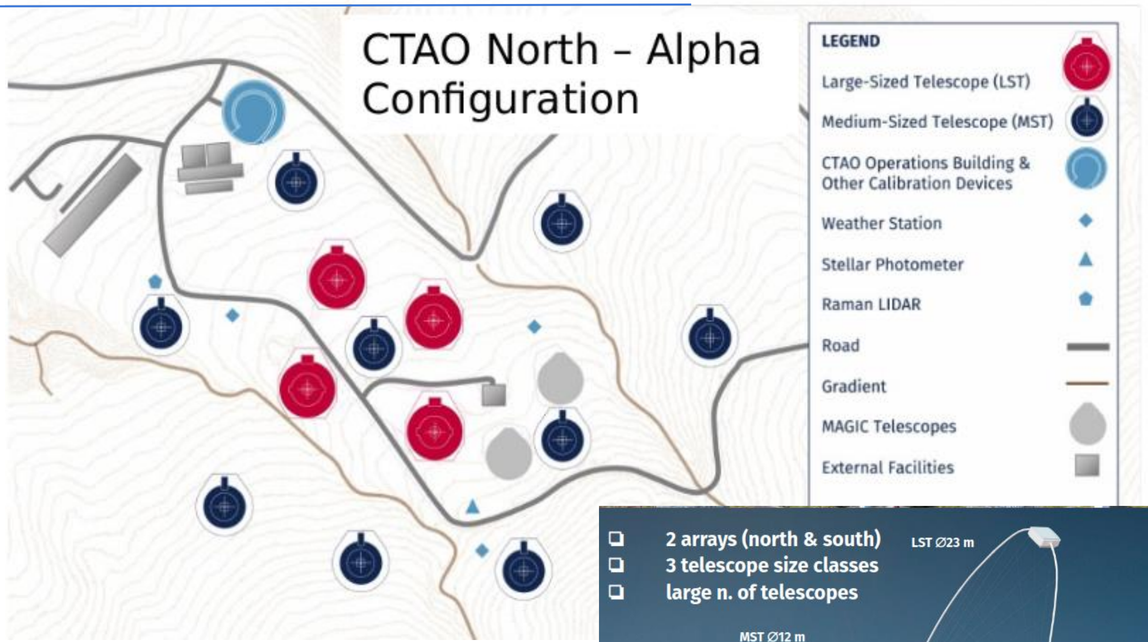


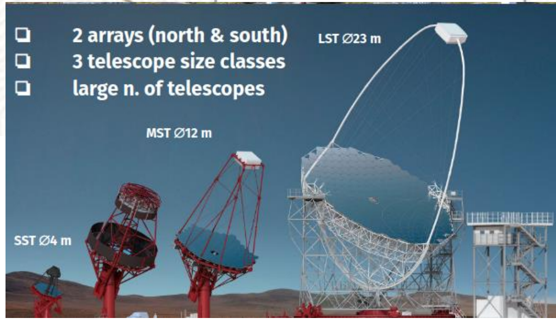
The Large-Sized Telescope



At the Observatorio Roque del Los Muchachos two types of telescopes:

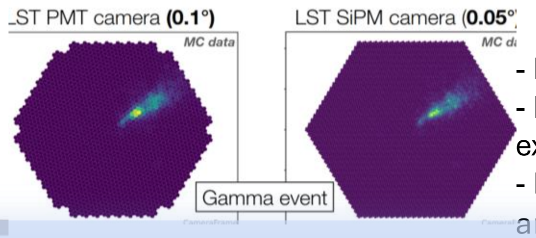
- 4 Large-Sized Telescops (LSTs)
- 9 Medium-Sized Telescopes (MSTs)


CTAO North – Alpha Configuration
LEGEND
Large-Sized Telescope (LST)
Medium-Sized Telescope (MST)
CTAO Operations Building & Other Calibration Devices
Weather Station
Stellar Photometer
Raman LIDAR
Road
Gradient
MAGIC Telescopes
External Facilities



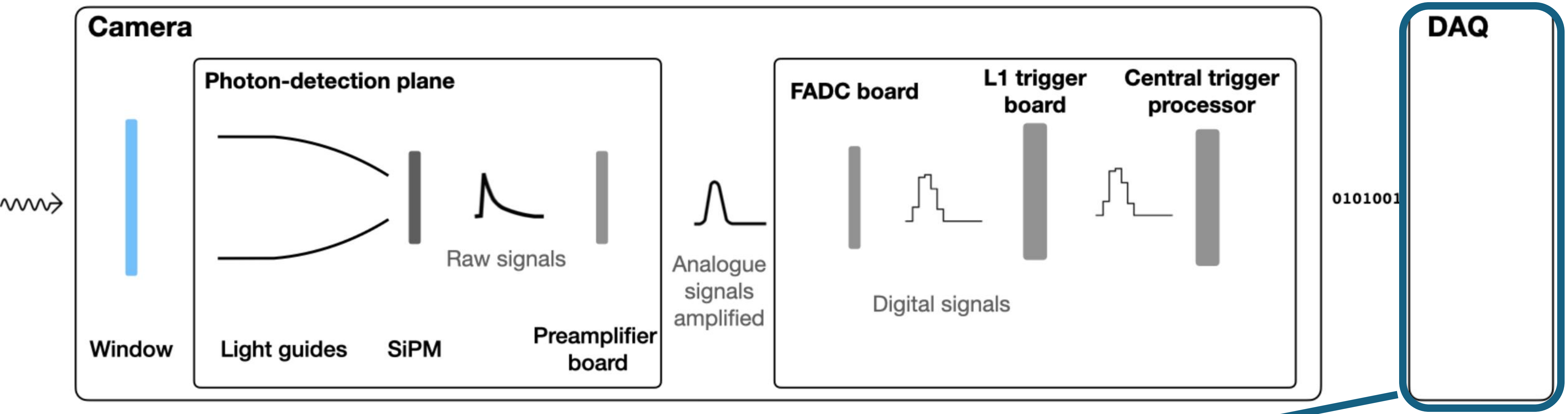
LST-1 first telescope at north site:
- Telescope inaugurated in 2018
- Fully takes data since November 2019

LST-2, LST-3, and LST-4: under construction → LST Advanced SiPM Camera



- Improve duty cycle, robustness, stability using SiPMs
- Increase image granularity for better image feature extraction
- Fully digital readout for better upgradability and use of artificial intelligence at earliest stage of the readout chain

Simplified camera architecture



1 GHz sampling rate ← 72 Tbps
After Level-1 trigger:
300 kHz ← 24 Gbps
After Level-2 trigger :
30 kHz
After stereo software trigger:
10 kHZ

Data Acquisition and Advanced Trigger

- Assemble events from all telescopes
- Perform stereo software trigger and potential data volume reduction (gamma/hadron separation)
- Neural Network algorithm for the high-level trigger → FPGA ?
- Altera®-based FPGA network card for DAQ (PCIe400)

Biggest FPGA Manufactures

- Altera® 30% share
- Xilinx 50% share

→ 80% share

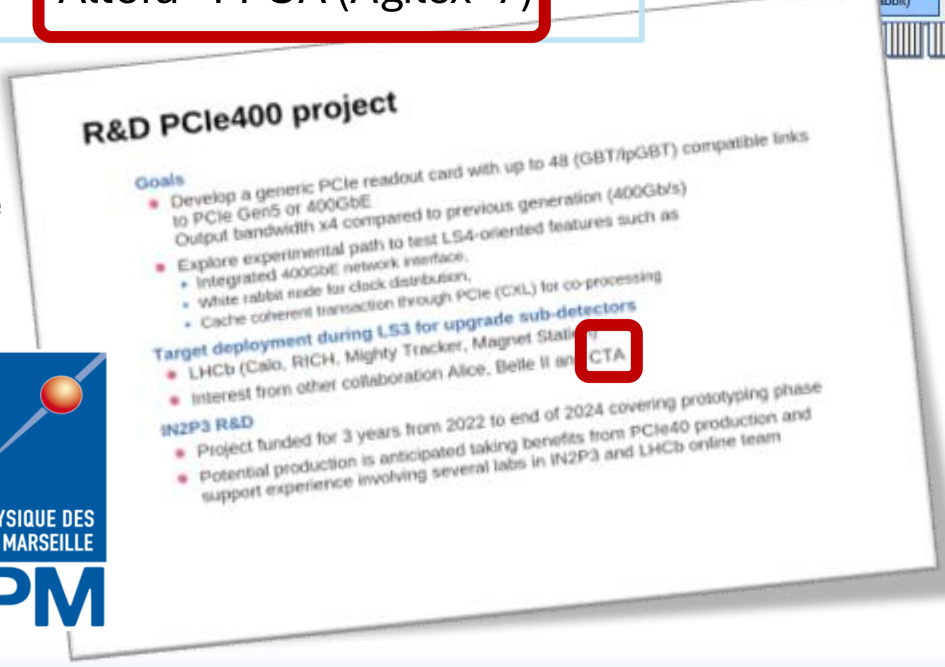
- Microsemi
- Lattice Semiconductor
- Achronix +
- Flex Logix +
- GOWIN Semiconductor
- Microchip Technology +
- Efinix +
- QuickLogic +

} 15% share

At the current design, the DAQ of LST Advanced camera is carried out by PCIe400 board.

↓

High-level DNN trigger algorithm must be run on Altera® FPGA (Agilex®7)



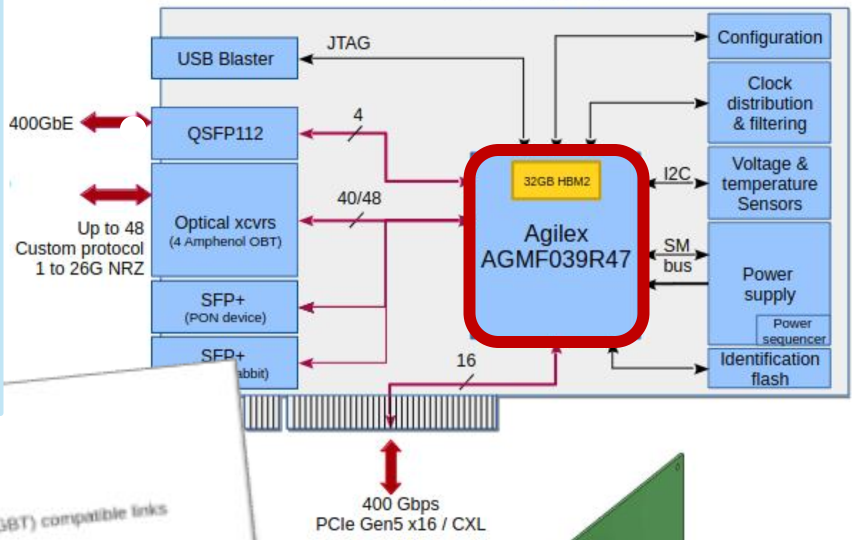
R&D PCIe400 project

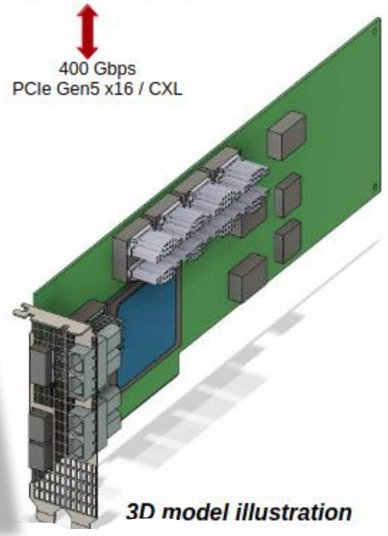
Goals
- Develop a generic PCIe readout card with up to 48 (GBT/lpGBT) compatible links to PCIe Gen5 or 400GbE
  Output bandwidth x4 compared to previous generation (400Gb/s)
- Explore experimental path to test LS4-oriented features such as
  - Integrated 400GbE network interface,
  - White rabbit node for clock distribution,
  - Cache coherent transaction through PCIe (CXL) for co-processing

Target deployment during LS3 for upgrade sub-detectors
- LHCb (Calo, RICH, Mighty Tracker, Magnet Station)
- Interest from other collaboration Alice, Belle II and CTA

IN2P3 R&D
- Project funded for 3 years from 2022 to end of 2024 covering prototyping phase
- Potential production is anticipated taking benefits from PCIe40 production and support experience involving several labs in IN2P3 and LHCb online team

PCIe400 synoptic





3D model illustration

Deep Learning Inference on FPGA

ADVANTAGES	**Predictable low latency and high throughput** FPGAs give low latency for real-time applications, bypassing CPU	**Low power consumption** FPGAs allow to modify the hardware architecture to adjust power consumption. They parts of a chip can be used without involving the entire chip to reduce power consumption	**Massively parallel data processing, customer data precision and data paths** allows programming power to scale as much as needed.
DISADVANTAGES	**Sophisticated programming** FPGAs require specific engineering expertise to map custom circuits and the architecture of the hardware.	**High initial cost** This one follows from the previous disadvantage, because greater expertise results in higher cost per unit.	**Complication with the code** Most code samples won’t easily migrate between GPUs and FPGAs.

Deep Learning on FPGA

- Deep learning models are trained on PCs with GPUs
- To maximise throughput and minimise latency for inference, it is advantageous to implement deep learning models in FPGAs for triggering.
- One way - write VHDL code
- Simpler way - use deep learning compilers for FPGA.

Altera® FPGAs

$$\begin{tikzcd} & \text{Altera® FPGAs} \arrow[dl] \arrow[dr] & \\ \text{Intel® FPGA AI Suite} & & \text{hls4ml} \end{tikzcd}$$

Intel® FPGA AI Suite

- various model sizes to fit
- provided directly by Altera® experts
- lower throughput depending on the model size

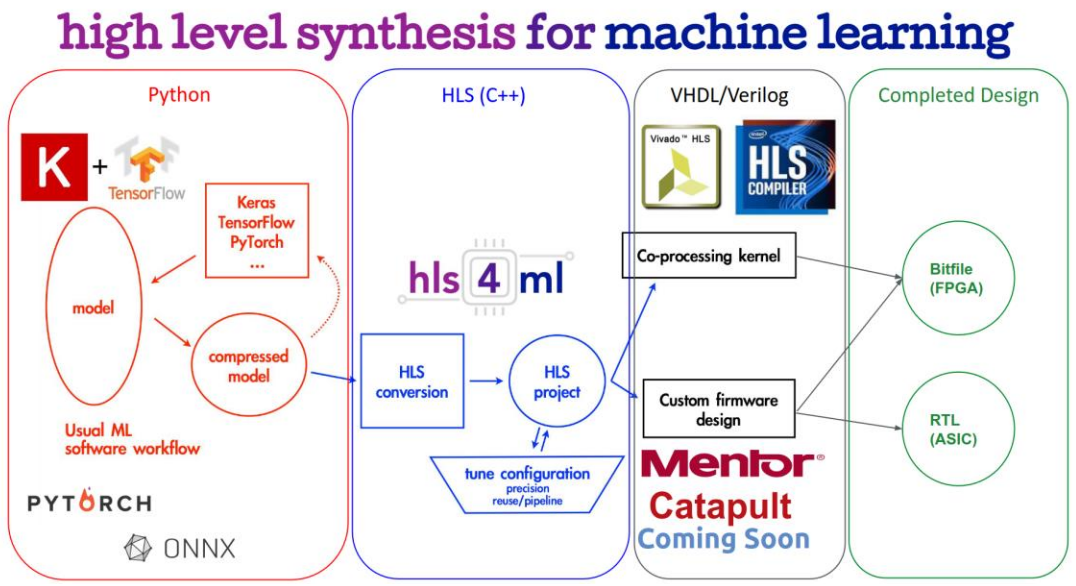
hls4ml

- potentially higher possible throughput
- smaller model sizes to fit

We study possibilities and performances of both packages for implementing trigger DNN for LST Advanced Camera.

- User-friendly tool for the automatic build and optimization of DL models for FPGAs
- Reads as input models that have been trained with standard DL libraries
- Uses various high-level syntesis compilers as backend, depending on requirements.

- No loading weights from external sources (e.g. DDR, PCIe).
- Much **faster access times** (on-chip weights).



- Hls4ml was originally developed to process extremely high data rates at the (HL-)LHC
- Therefore, **support** for the **Xilinx** boards, commonly used in the ATLAS and CMS experiments, is much **more advanced** at the moment.
- Hls4ml support for Altera® devices is being implemented by Fermilab.

Models

The final goal: to port the deep convolutional neural networks (CNNs) for LST triggering created using a dedicated framework for IACT event reconstruction and data management of deep-learning-based image and waveform analysis techniques for IACT data.

Details about the model → CNN-based models on calibrated waveforms for the Large-Sized Telescope prototype of the Cherenkov Telescope Array

Our studies:

TensorFlow (Keras) trained models with a CNN block, a few dense layers and a softmax activation layer.

The size and number of CNN blocks vary in order to find an optimal compromise between throughput and physics performance.

Benchmark models for evaluation:

- ~ 6M parameters
- ~ 200k parameters
- ~ 50k parameters
- ~ 2k parameters

All tested models can be found at:
https://github.com/yabezsh/LST-AI-trigger-FPGA

For the initial studies with hls4ml, a simple model with 3 hidden layers of 64, then 32, then 32 neurons was also used. Each layer use relu activation. One output layer with 5 neurons, finish with softmax activation.

Achievements & Discoveries: hls4ml

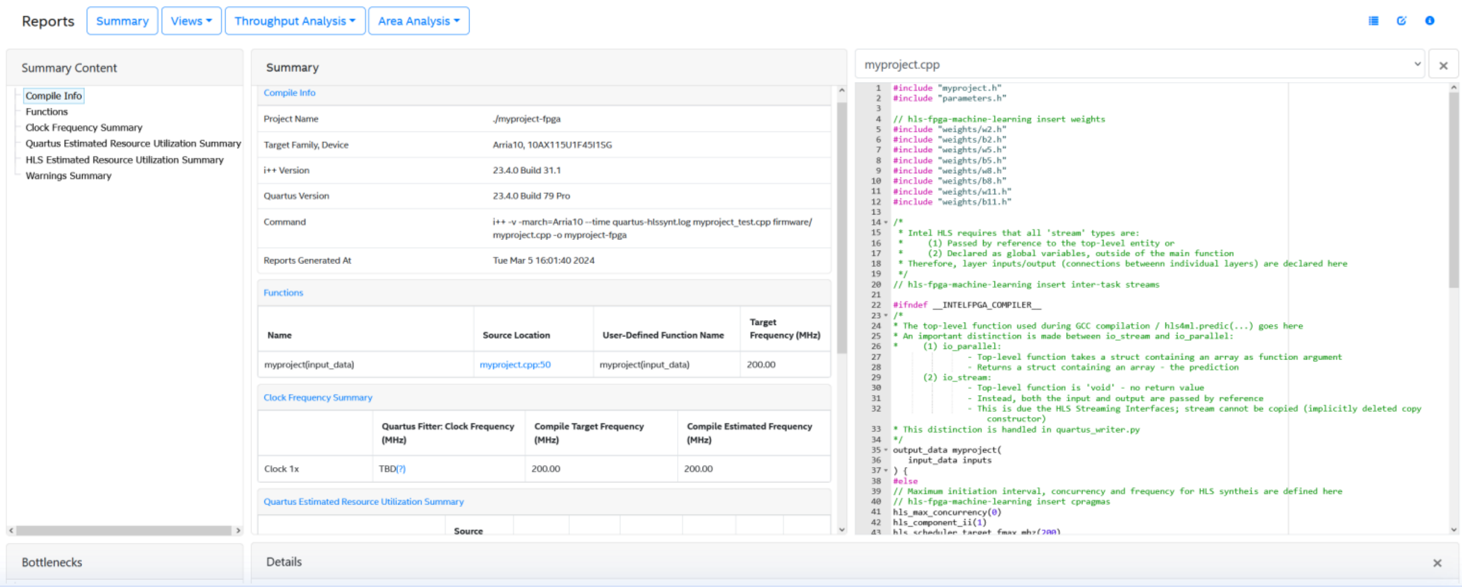
Inputs:

- hls4ml was originally developed by/for Xilinx users.
- The majority of hls4ml users use Xilinx
- Support for Altera® has been implemented, but may not be fully mature yet

- All examples/documentation are intended for the Xilinx backend, it takes time to find the right configuration.
- The simple model was successfully executed with Quartus® (=Altera® backend).
- hls4ml uses the Intel® HLS compiler to convert the code to RTL and create a testbench.

Managed to run the Quartus® compilation on the RTL files created by hls4ml with Intel® HLS Compiler backend for the simple DNN model. As results achieved to get QoR like fmax and resource utilisation

(200MHz on Intel® Arria® 10 PAC)



Summary

Compile Info

Project Name	./myproject-fpga
Target Family, Device	Arria10, 10AX115U1F45I1SG
i++ Version	23.4.0 Build 31.1
Quartus Version	23.4.0 Build 79 Pro
Command	i++ -v -march=Arria10 --time quartus-hlssynt.log myproject_test.cpp firmware/ myproject.cpp -o myproject-fpga
Reports Generated At	Tue Mar 5 16:01:40 2024

Functions

Name	Source Location	User-Defined Function Name	Target Frequency (MHz)
myproject(input_data)	myproject.cpp:50	myproject(input_data)	200.00

Clock Frequency Summary

	Quartus Fitter: Clock Frequency (MHz)	Compile Target Frequency (MHz)	Compile Estimated Frequency (MHz)
Clock 1x	TBD(?)	200.00	200.00

Quartus Estimated Resource Utilization Summary

Source

```
#include "myproject.h"
#include "parameters.h"

// hls-fpga-machine-learning insert weights
#include "weights/w2.h"
#include "weights/b2.h"
#include "weights/w5.h"
#include "weights/b5.h"
#include "weights/w8.h"
#include "weights/b8.h"
#include "weights/w11.h"
#include "weights/b11.h"

/*
 * Intel HLS requires that all 'stream' types are:
 *     (1) Passed by reference to the top-level entity or
 *     (2) Declared as global variables, outside of the main function
 * Therefore, layer inputs/output (connections betweenn individual layers) are declared here
 */
// hls-fpga-machine-learning insert inter-task streams

#ifndef __INTELFPGA_COMPILER__
/*
* The top-level function used during GCC compilation / hls4ml.predic(...) goes here
* An important distinction is made between io_stream and io_parallel:
*     (1) io_parallel:
                - Top-level function takes a struct containing an array as function argument
                - Returns a struct containing an array - the prediction
      (2) io_stream:
                - Top-level function is 'void' - no return value
                - Instead, both the input and output are passed by reference
                - This is due the HLS Streaming Interfaces; stream cannot be copied (implicitly deleted copy
                  constructor)
* This distinction is handled in quartus_writer.py
*/
output_data myproject(
   input_data inputs
) {
#else
// Maximum initiation interval, concurrency and frequency for HLS syntheis are defined here
// hls-fpga-machine-learning insert cpragmas
hls_max_concurrency(0)
hls_component_ii(1)
hls_scheduler_target_fmax_mhz(200)
```

Achievements & Discoveries: hls4ml

- When trying to compile the trigger model, the error reported in the past by developers for hls4ml with Intel® HLS compiler backend was received.

- The developers tell us that there was a **problem** with **CNN** support for the **Altera® backend** in the past due to:

1) The fact that you could not specify the buffer size of streams
2) The padding function seemed to be broken

- The Intel® High Level Synthesis (**HLS**) **compiler** is said to be **deprecated** in favour of the Intel® oneAPI IP Authoring Flow.

- For this reason, **hls4ml** has **stopped** the development for the Altera® (= **Intel® HLS Compiler**) backend and **started** to work on the **implementation** of the new Intel Intel® **oneAPI** backend.

- The Intel® **oneAPI** backend **isn't** yet **available** for public use.

→ At the moment there is no possibility to run our CNN on an Altera® FPGA with hls4ml. We are looking on how to patch the needed layers in the Intel® HLS Compiler support.

- **Altera®** took an interest in the **hls4ml** project and made its **experts** available to **help** with the development of Intel® **oneAPI** support. We are looking forward to the release of hls4ml with Intel® oneAPI support for Altera® FPGAs soon! :)

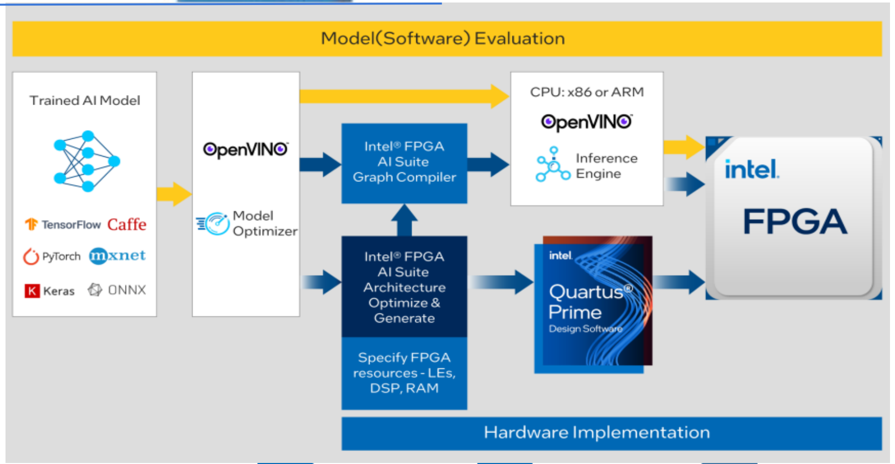
Intel® FPGA AI Suite

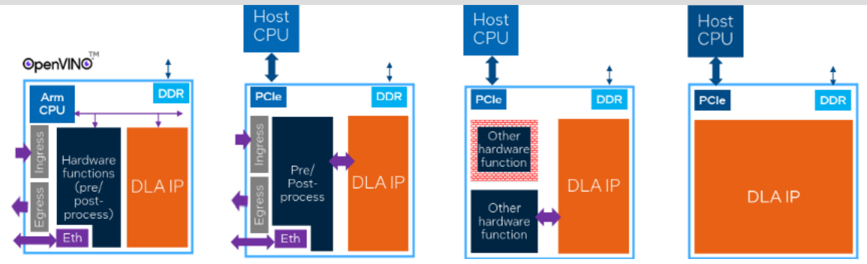
The FPGA AI Suite is a powerful toolset provided (= supported) by Altera®

- High performance
- 3 679 resnet-50 frames per second at 90% FPGA utilisation with Intel® Agilex® 7 FPGA M-Series

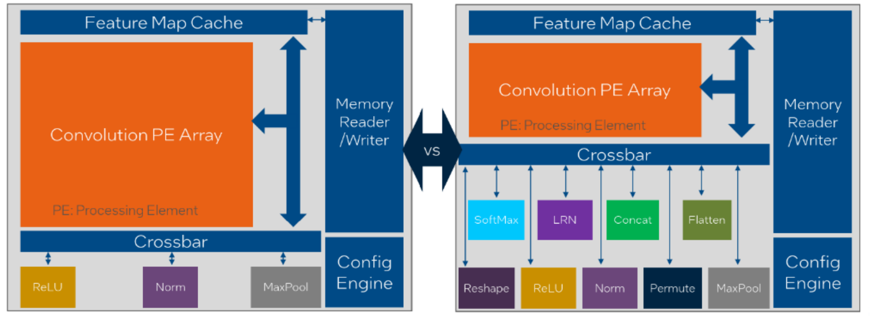
- Model optimizer for creating network files (.xml) and files with weights and biases (.bin) for intermediate representation.
- DLA compiler to provide estimated area or performance metrics for a given architecture file or to create an optimized architecture file and compile the network.
- The compiled file is imported at runtime (Inference Engine API; FPGA AI)

- Allows mixed heterogeneous execution
- Enables different use cases of FPGA resources
- The architecture optimizer can be used to optimise the implementation for the specific network and achieve the best performance.
- Model optimizers configure the network for the best performance on Altera® hardware.





▪ Architecture is adaptable to support new or evolving networks



Achievements & Discoveries: Intel® FPGA AI Suite

- We have been working with the Altera® group since 2021, using the Intel® FPGA AI Suite to implement algorithms needed for particle physics (first official Intel® FPGA AI Suite release - 2023).

- Strong interest and support from Altera® in understanding our requirements, regular meetings, good feedback on the status of the package and perspective developments.

- The package requires combined use with Intel® OpenVINO™ for model optimization.

- There is a certain time delay in supporting the latest Intel® OpenVINO™ versions (= latest Tensorflow versions)

- Not all architectures/layers are supported, but new ones are constantly being added. Huge progress in support since 2021.

- We have experience with running inferences for various models on the server installed with the Intel® Arria® 10 PAC at the University of Zurich.

- We are not the typical customers with small "images" at high rates, the software is developed with image/video processing in mind.

- Intended for milliseconds latency in complex networks, not microseconds in simple networks like we intend

Achievements & Discoveries: Intel® FPGA AI Suite

- Initially, only **200** inferences/s were achieved on the **Intel® Arria® 10 PAC** card for the original model for high-level triggers in the LST Advanced Camera (**6M** parameters).

- The Altera® group advised increasing the clock rate to 600 MHz (standard 400 MHz) to determine the maximum achievable performance.

→ **732** inferences/s on **Agilex® 7**. Assuming an implementation with 4 instances of the inference IP in Agilex® 7, this would result in 2928 inferences/s.

- The model sizes were reduced to almost two orders of magnitude.

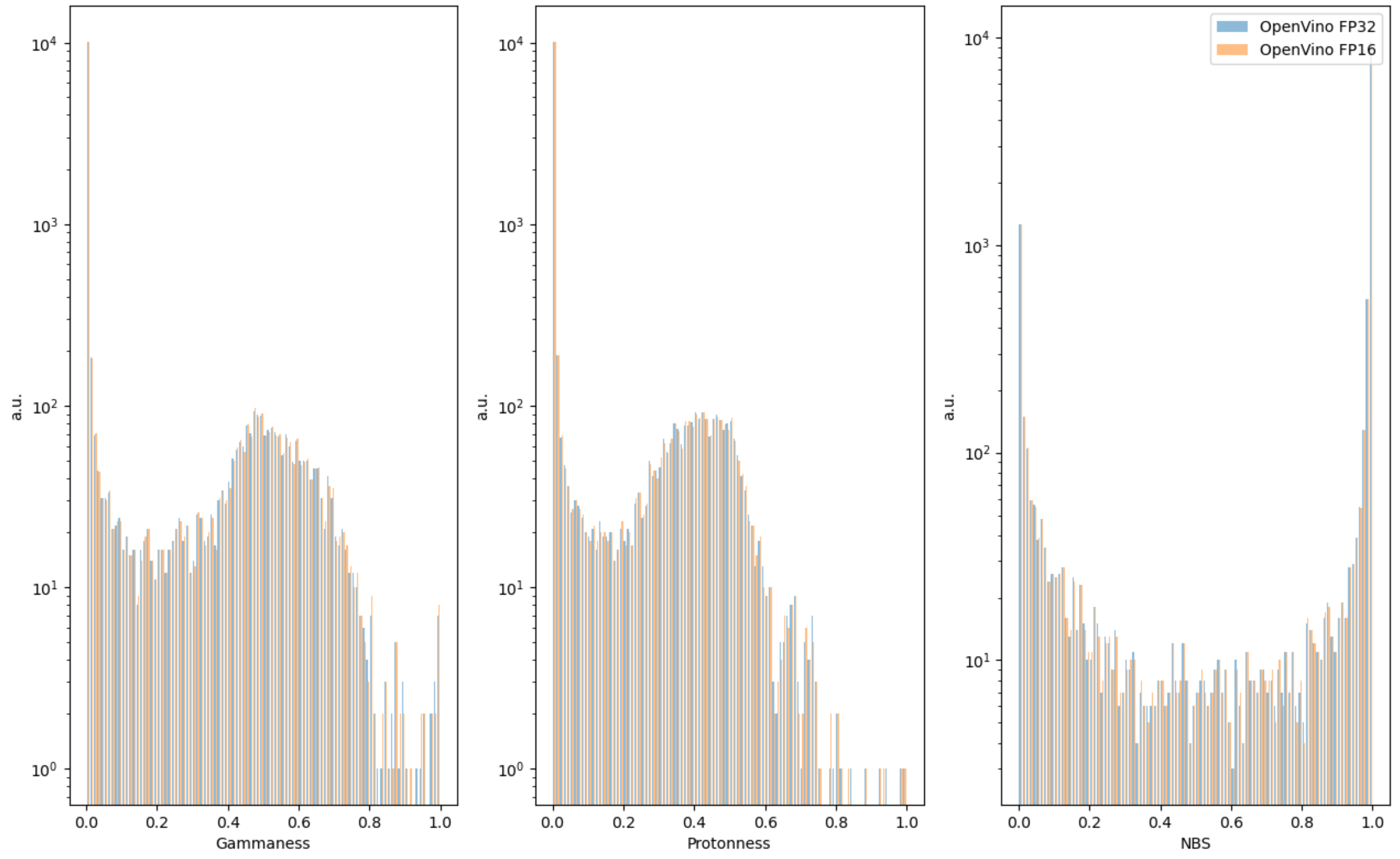
- Problems occurred with the new models due to the different version support of TensorFlow/Intel® OpenVINO™/Intel® FPGA AI Suite.

- Altera® offered us the solution for version incompatibilities (should not be generally used, but the problem will be fixed with the new version of AI Suite).

$N_{parameters}$	6M	200k	50k	2k
Throughput	732 fps	20 839 fps	22 131 fps	22 202 fps

- By optimising the architecture based on the graph of the network, ~ **40k fps** could be achieved.
- These results are sufficient for the CTAO trigger rate.
- We are now working on evaluating the impact of the switch to lower precisions on the physical performance of the model.
- Running algorithm on the installed Arria10 FPGA using c++ libraries.

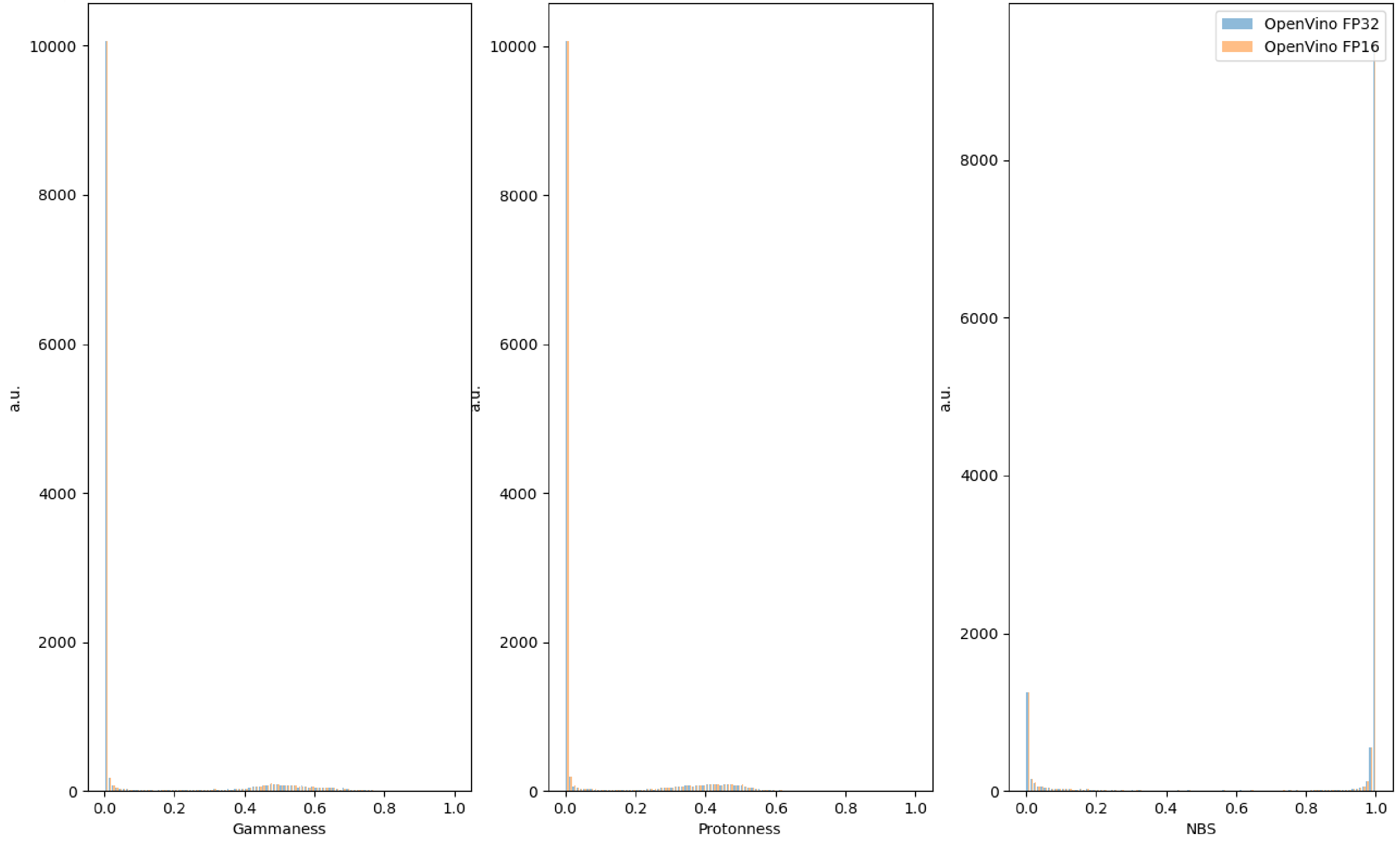
Preliminary results on the physics performance



Summary

- The new advanced camera being developed for the large-sized telescope at CTAO North must be able to perform gamma/hadron separation at the trigger level at high rates
- DNN algorithms is developed to perform efficient triggering at high level
- DAQ of the trigger is planned to be done via PCIe400 network card equipped with an Altera® Agilex®7 FPGA card
- We have investigated two approaches to port the DNN trigger algorithm to an Altera® FPGA board: hls4ml and Intel® FPGA AI Suite
- Hls4ml provides better results in terms of maximum achievable throughput in perspective, but currently is very limited support for Altera® boards due to deprecation of the Intel® HLS compiler. Good potential with the release of the new Intel® oneAPI backend support.
- The Intel® FPGA AI Suite doesn't work with the latest Tensorflow models, but can easily be patched to do so. The problem should be fixed with the new version, most likely later this year
- We managed to reach 40k fps with one core on Agilex7 with AI Suite.
- Investigating the option of using HBM2e (High Bandwidth Memory) instead of onboard DDR4 memory, which would be available on the Agilex®7 M Series board.
- We're now working on investigating the effect of precision reduction on physics performance and using the better space of FPGA with architecture optimization.
- Precision error of FP16 for the current model ~O(10^{-3}), ongoing studies on the further precision reduction effects (FP12, INT9, INT8). Quantisation aware training may improve the precision drop.

Backup



Preliminary results on the physics performance

- Used two models from Tjark to compare the physics performance of the model for Temsorflow/OpenVino

Barvinok 3: - finaltrigger_cnn_8_16_fch_32_GlobalMaxPool outputs: gammanness, protonnes, nsb

Barvinok 2: - finaltrigger_cnn_8_16_fch_32_GlobalMaxPool_noNSB_10epochs outputs: gammanness, protonnes

Input data : nsb

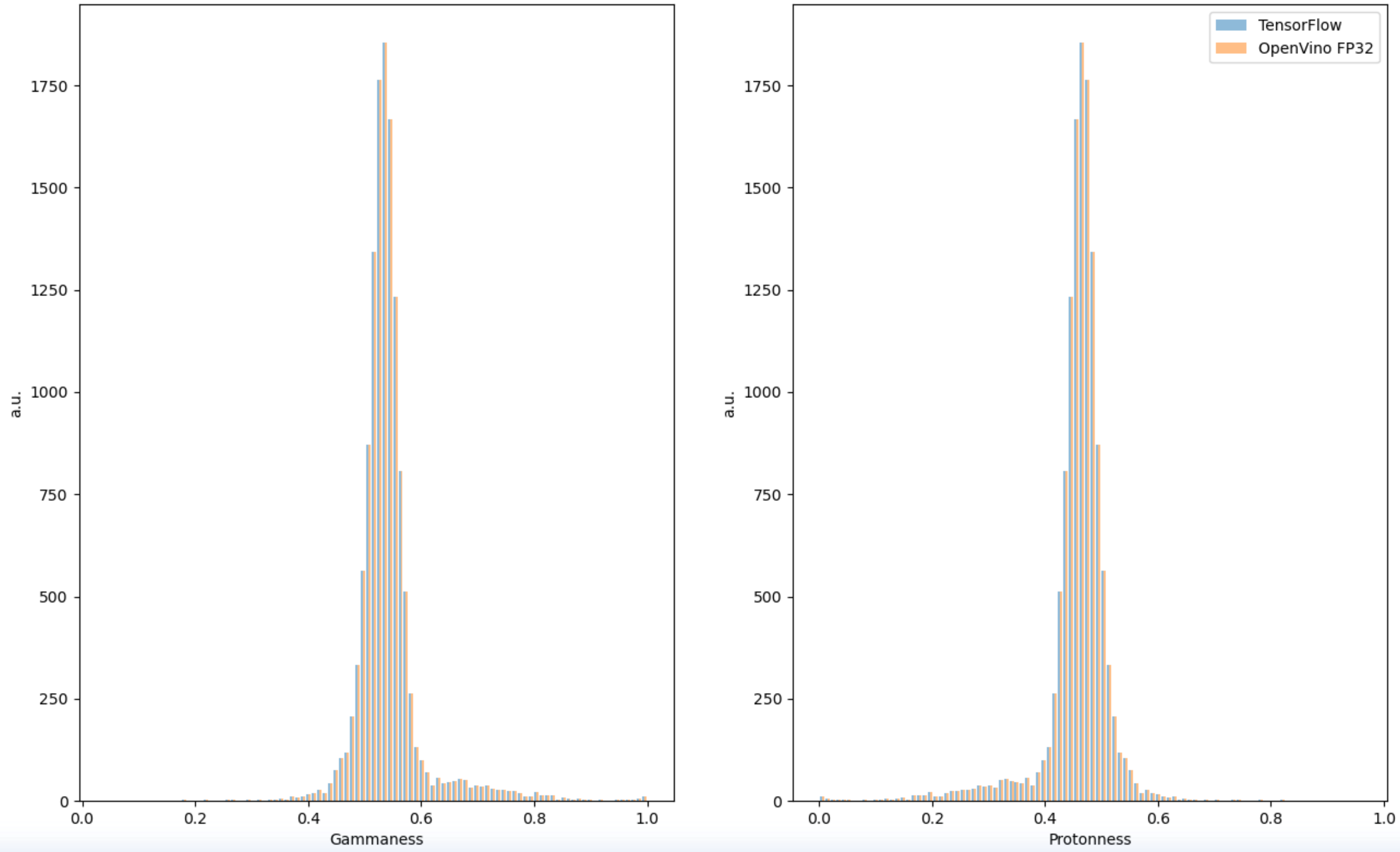
Barvinok 2

finaltrigger_cnn_8_16_fch_32_GlobalMaxPool_noNSB_10epochs

outputs: gammanness, protonnes

Input data : nsb

Comparison of the **Tensorflow** (Tjark's) and OpenVino **FP32** outputs



Universität Zürich^UZH

CTAO

Preliminary results on the physics performance

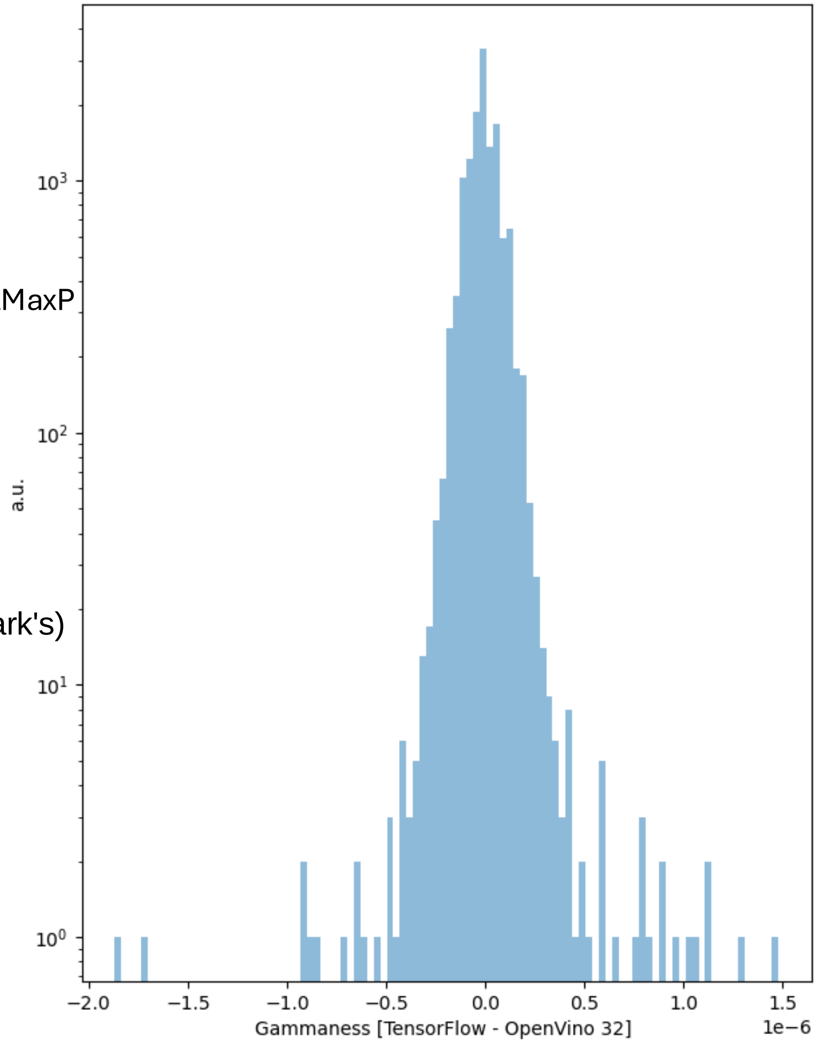
finaltrigger_cnn_8_16_fch_32_GlobalMaxPool_noNSB_10epochs

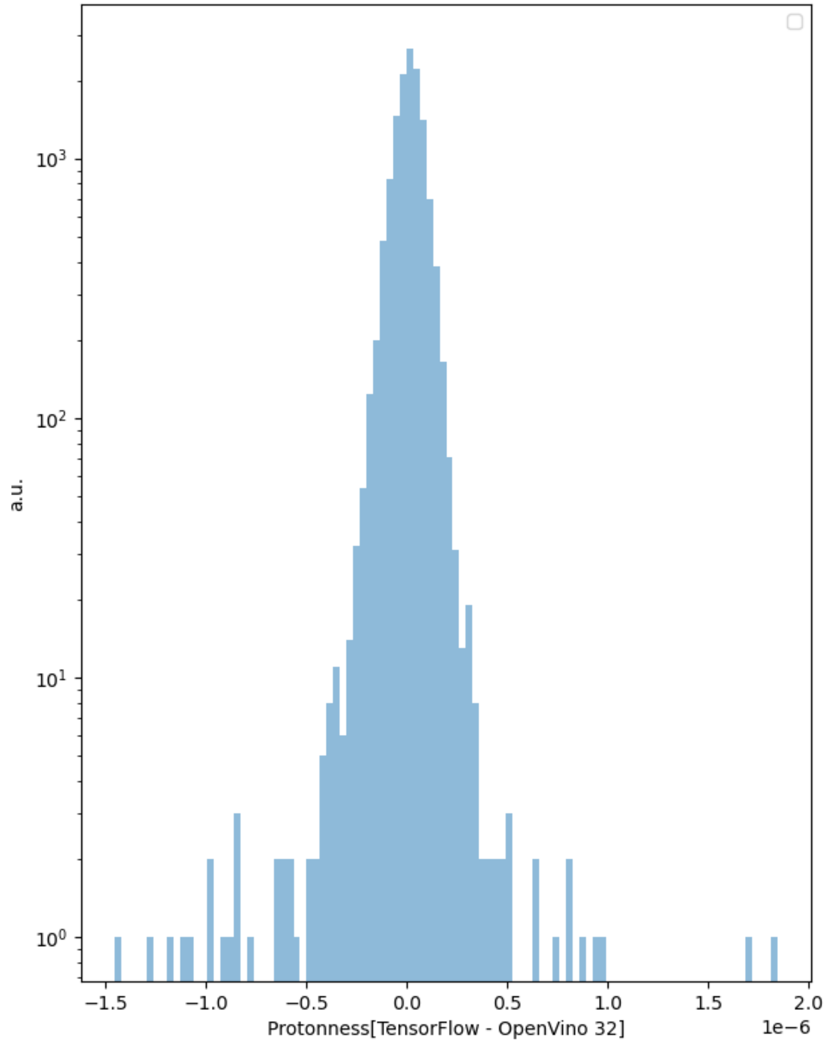
outputs: gammanness, protonnes

Input data : nsb

Barvinok 2

- Differences on the **Tensorflow** (Tjark's) and OpenVino **FP32** outputs





Preliminary results on the physics performance

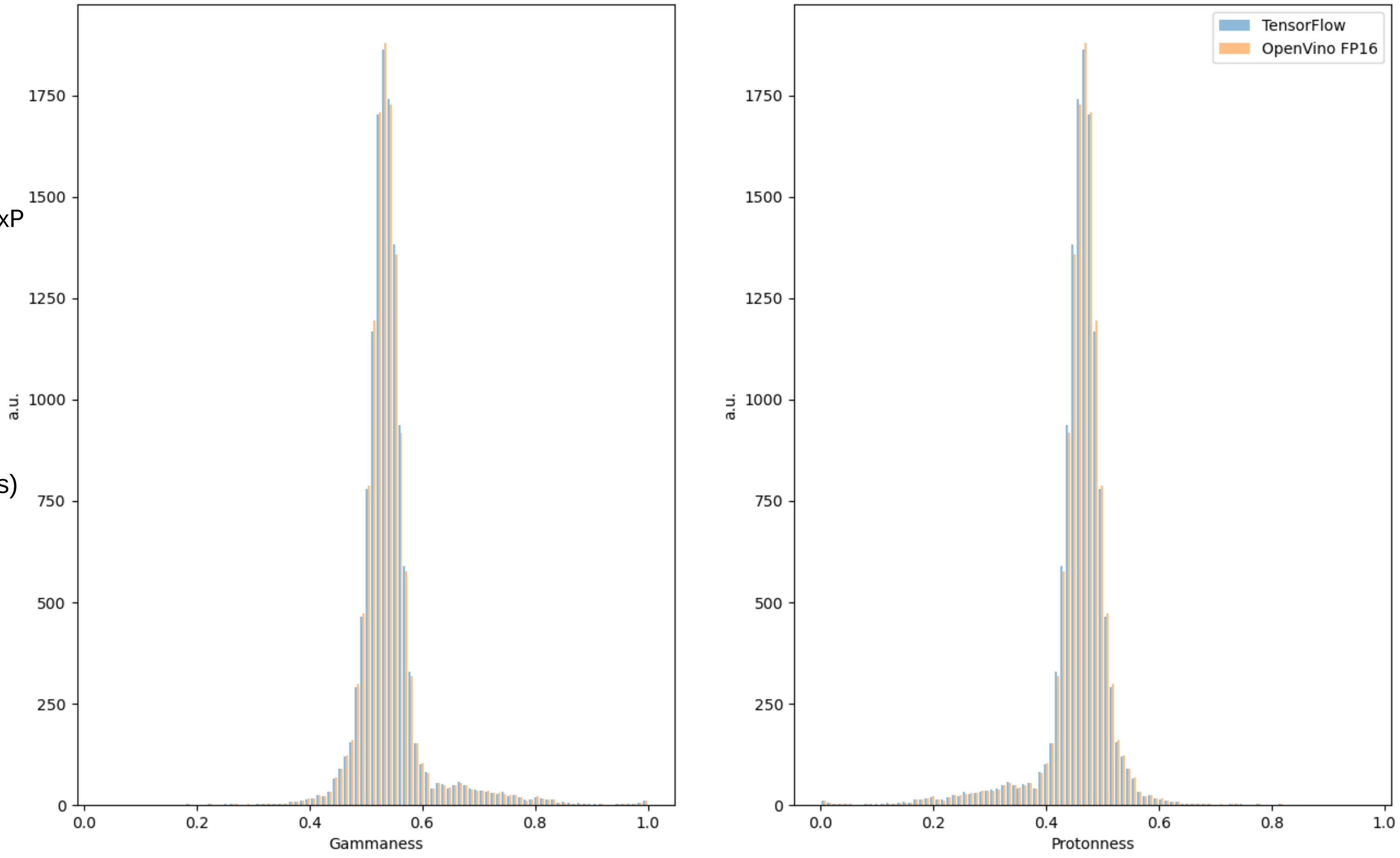
finaltrigger_cnn_8_16_fch_32_GlobalMaxPool_noNSB_10epochs

outputs: gammanness, protonnes

Input data : nsb

Barvinok 2

- Comparison of the **Tensorflow** (Tjark's) and OpenVino **FP16** outputs



Preliminary results on the physics performance

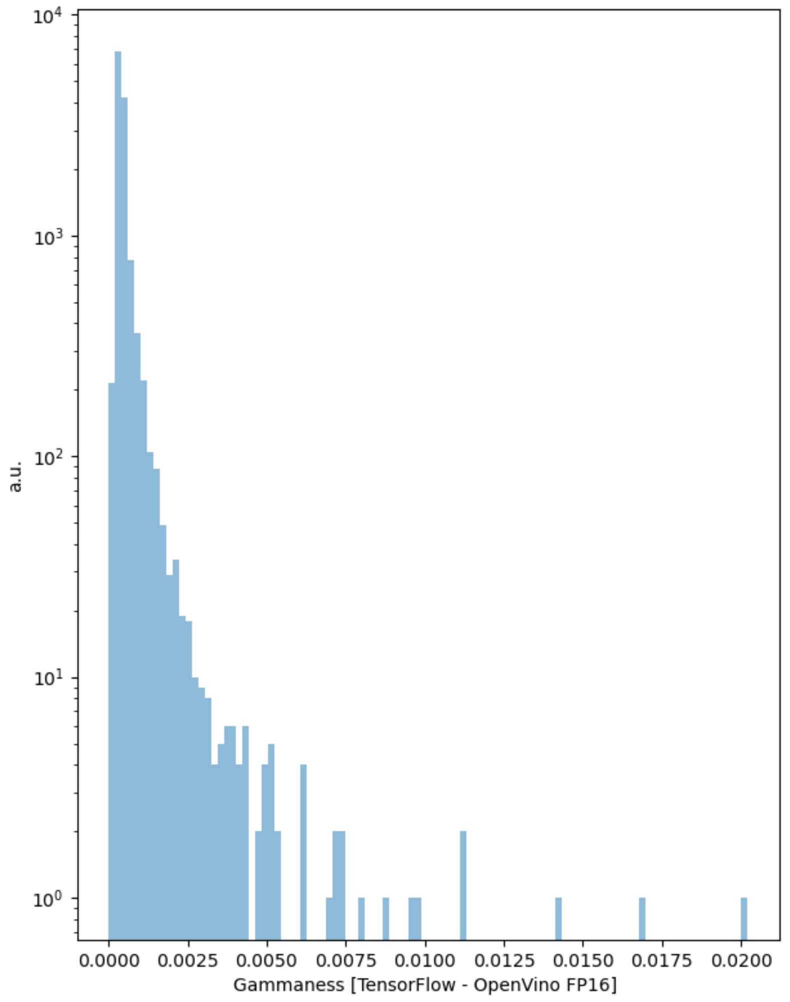
finaltrigger_cnn_8_16_fch_32_GlobalMaxPool_noNSB_10epochs

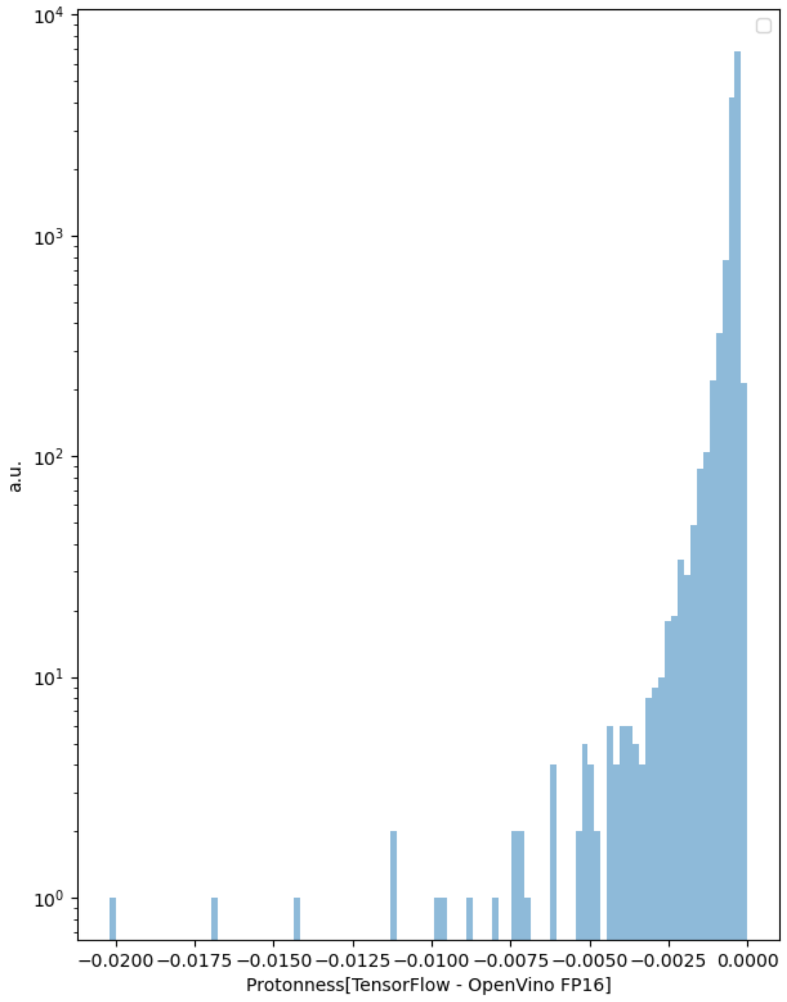
outputs: gammanness, protonnes

Input data : nsb

Barvinok 2

- Differences on the **Tensorflow** (Tjark's) and OpenVino **FP16** outputs





Universität Zürich UZH

Preliminary results on the physics performance

CTAO

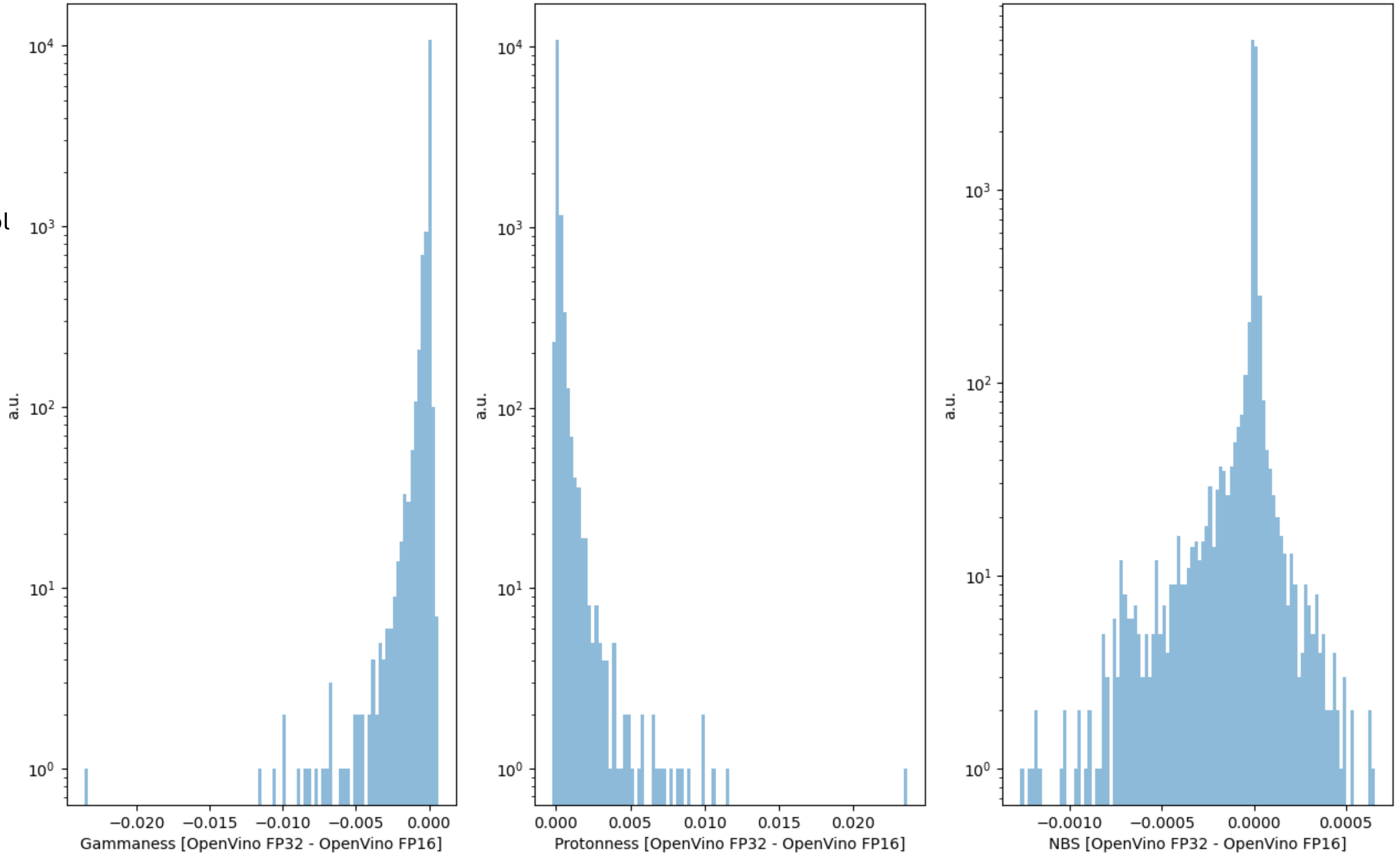
finaltrigger_cnn_8_16_fch_32_GlobalMaxPool

outputs: gammanness, protonnes, nsb

Input data : nsb

Barvinok 3

- Differences on the OpenVino **FP32** and OpenVino **FP16** outputs



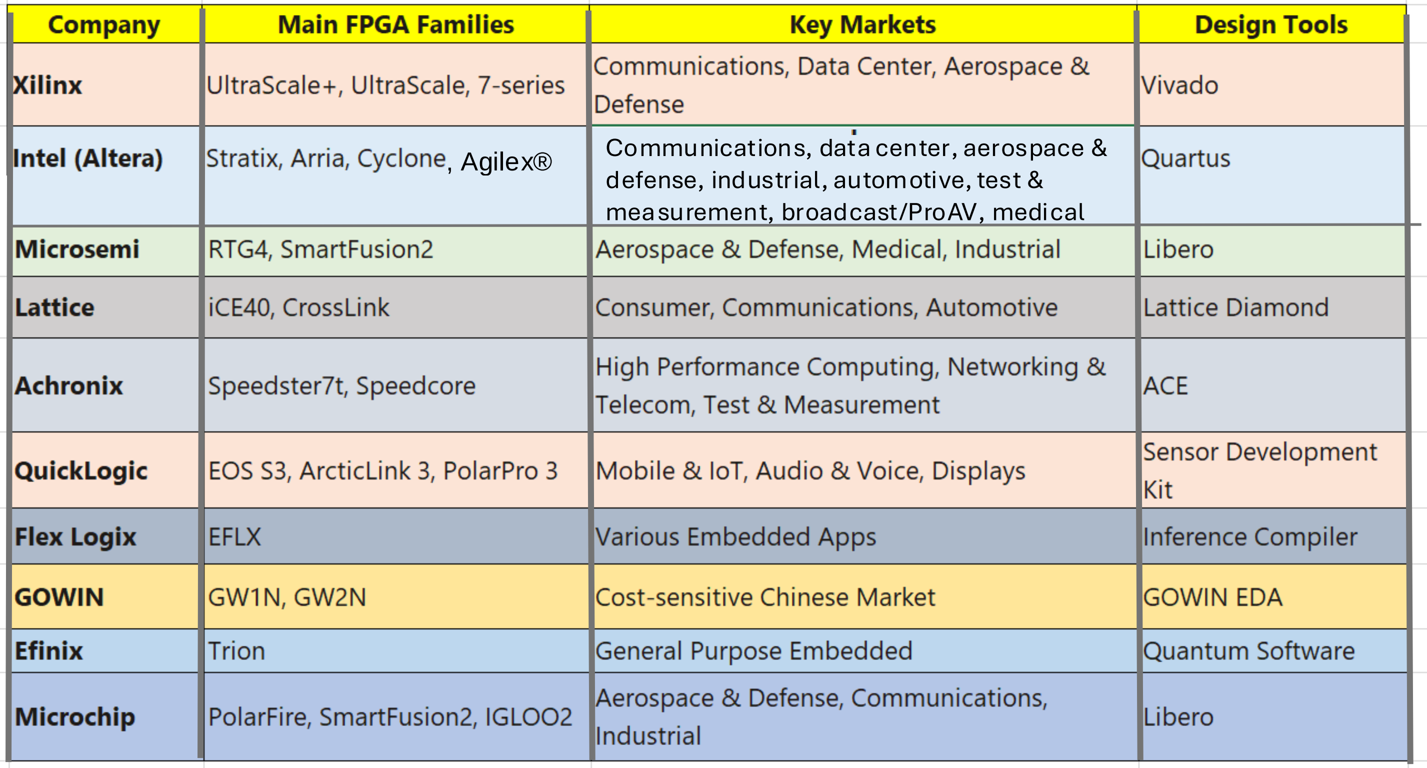
Biggest FPGA Manufactures

- Altera® 30% share
- Xilinx 50% share

80% share

- Microsemi
- Lattice Semiconductor
- Achronix +
- Flex Logix +
- GOWIN Semiconductor
- Microchip Technology +
- Efinix +
- QuickLogic +

15% share



Company	Main FPGA Families	Key Markets	Design Tools
Xilinx	UltraScale+, UltraScale, 7-series	Communications, Data Center, Aerospace & Defense	Vivado
Intel (Altera)	Stratix, Arria, Cyclone, Agilex®	Communications, data center, aerospace & defense, industrial, automotive, test & measurement, broadcast/ProAV, medical	Quartus
Microsemi	RTG4, SmartFusion2	Aerospace & Defense, Medical, Industrial	Libero
Lattice	iCE40, CrossLink	Consumer, Communications, Automotive	Lattice Diamond
Achronix	Speedster7t, Speedcore	High Performance Computing, Networking & Telecom, Test & Measurement	ACE
QuickLogic	EOS S3, ArcticLink 3, PolarPro 3	Mobile & IoT, Audio & Voice, Displays	Sensor Development Kit
Flex Logix	EFLX	Various Embedded Apps	Inference Compiler
GOWIN	GW1N, GW2N	Cost-sensitive Chinese Market	GOWIN EDA
Efinix	Trion	General Purpose Embedded	Quantum Software
Microchip	PolarFire, SmartFusion2, IGLOO2	Aerospace & Defense, Communications, Industrial	Libero

* taken from Top 10 FPGA Manufacturers in The World

hls4ml

- User-friendly tool for the automatic build and optimization of DL models for FPGAs
- Reads as input models that have been trained with standard DL libraries
- Uses various high-level syntesis compilers as backend, depending on requirements.

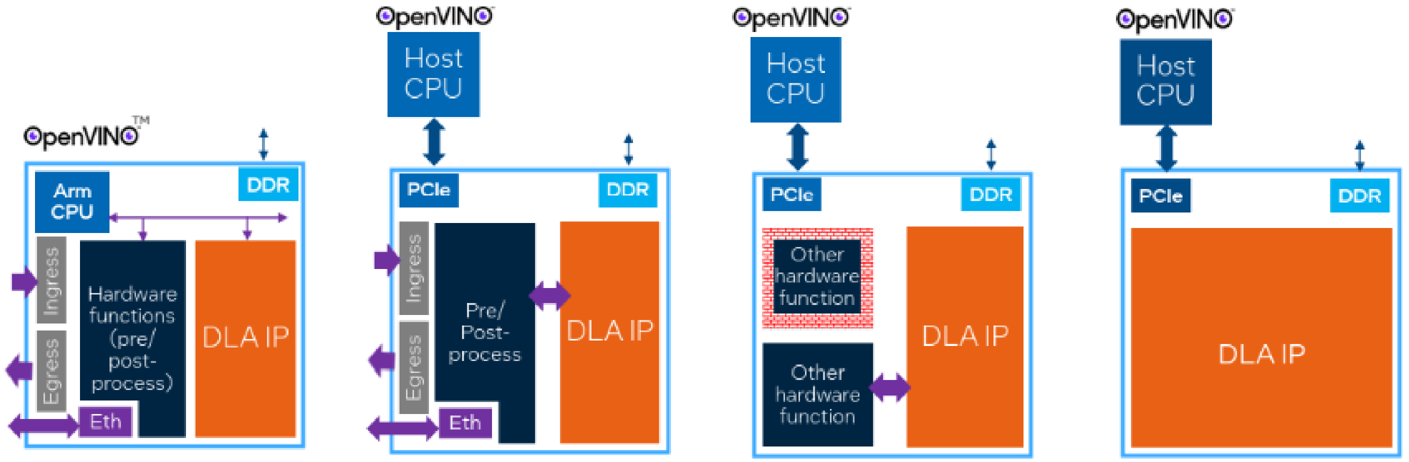
- No loading weights from external sources (e.g. DDR, PCIe).
- Much **faster access times** (on-chip weights).

- Hls4ml was originally developed to process extremely high data rates at the (HL-)LHC
- Therefore, **support** for the **Xilinx** boards, commonly used in the ATLAS and CMS experiments, is much **more advanced** at the moment.
- Hls4ml support for Altera® devices is being implemented by Fermilab.

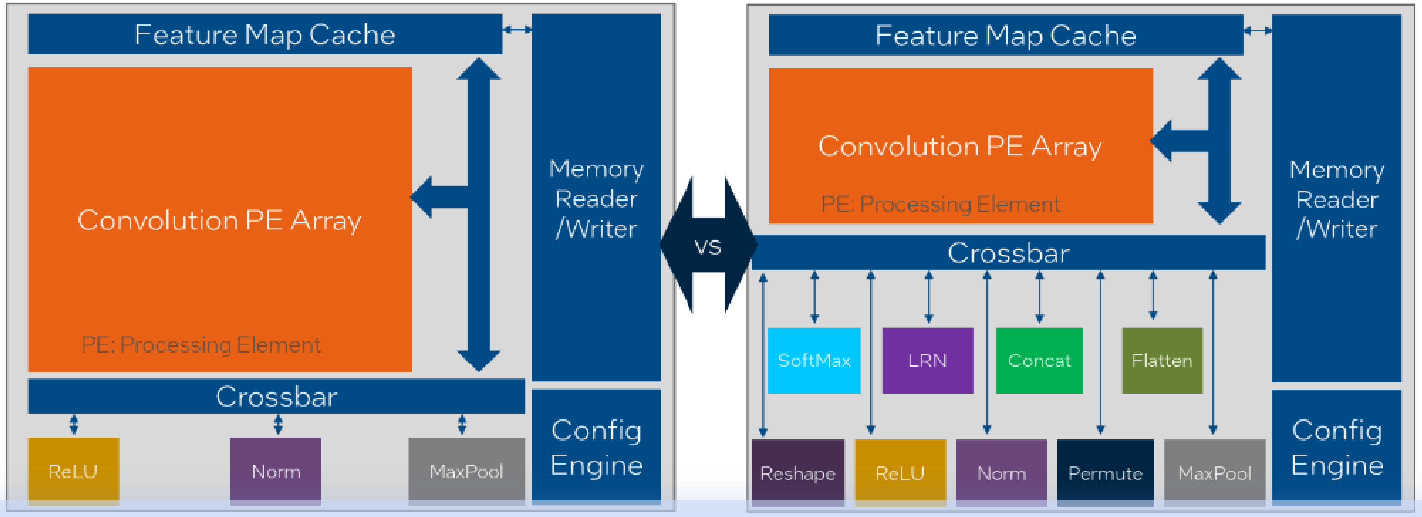
The network must be optimised for efficient use:

- **compression**: reducing the number of synapses or neurons
- **quantization**: reducing the precision of the calculations (inputs, weights, biases)
- **parallelization**: tuning the degree of parallelization to make inference faster/slower versus FPGA resources

Intel® FPGA AI Suite



- Architecture is adaptable to support new or evolving networks



- Model optimizer for creating network files (.xml) and files with weights and biases (.bin) for intermediate representation.
- DLA compiler to provide estimated area or performance metrics for a given architecture file or to create an optimized architecture file and compile the network.
- The compiled file is imported at runtime (Inference Engine API; FPGA AI)
- Allows mixed heterogeneous execution
- Enables different use cases of FPGA resources
- The architecture optimizer can be used to optimise the implementation for the specific network and achieve the best performance.
- Model optimizers configure the network for the best performance on Altera® hardware.