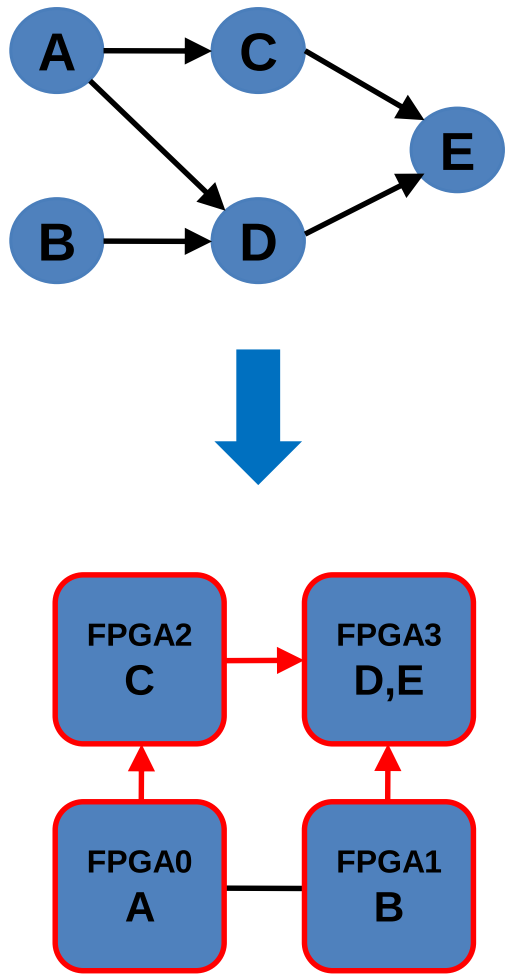
APEIRON
a heterogeneous computing platform for real-time inference

Paolo Cretaro

for the APEIRON team

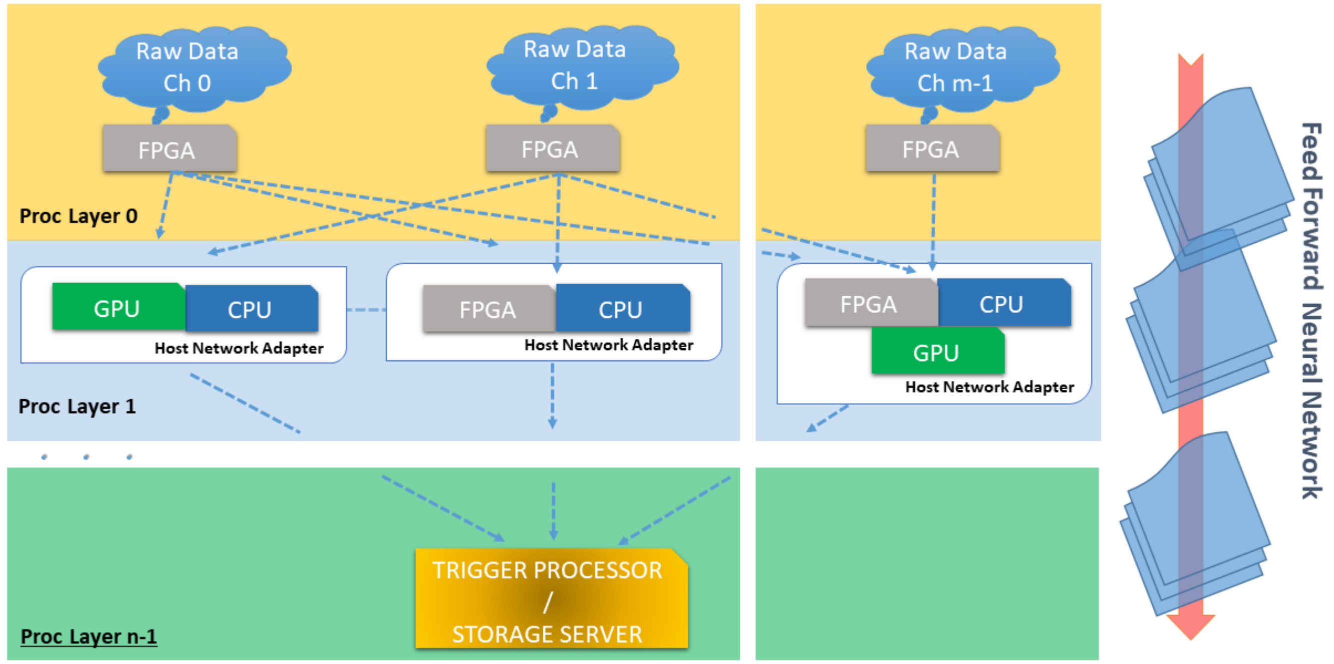
The INFN CSN-5 APEIRON Project Goals

- Offer hardware and software support for the execution on a system of **multiple interconnected FPGAs of applications developed according to a dataflow programming model**
- Map the directed graph of tasks on the distributed FPGA system and offer runtime support for the execution.
- Allow **users with no experience in hardware design tools** to develop their applications on such distributed FPGA-based platforms
  - Tasks are implemented in **C++ using High Level Synthesis tools**.
  - Simple **Send/Receive** C++ communication API.



APEIRON rationale

Abstract **P**rocessing **E**nvironment for **I**ntelligent **R**ead-**O**ut systems based on **N**eural networks

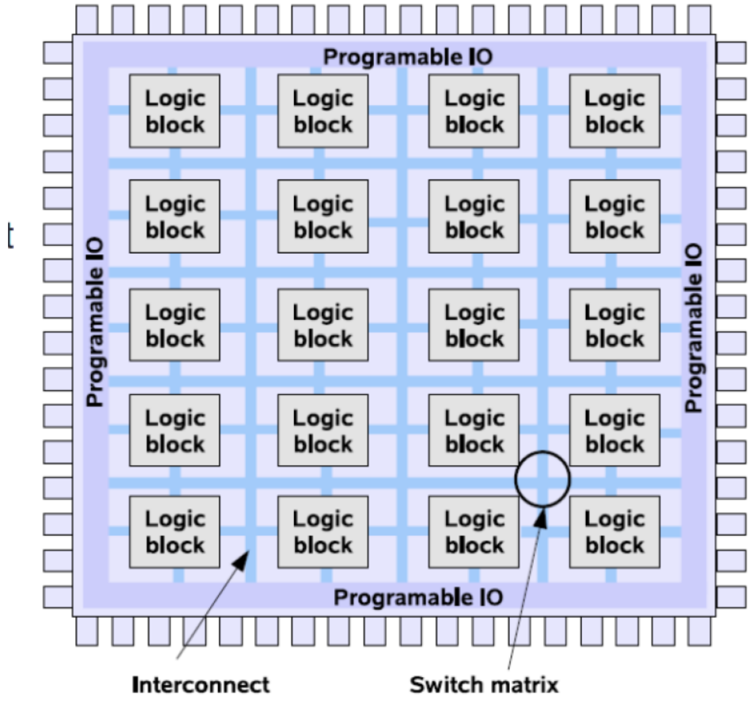


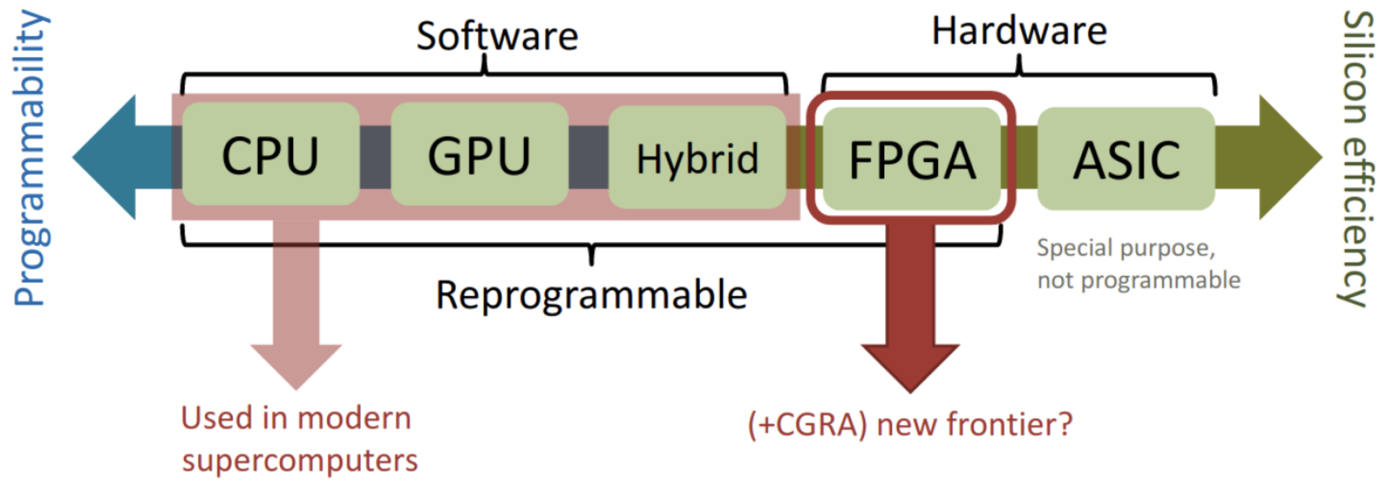
- Input data from several different channels (data sources, detectors/sub-detectors).
- **Data streams** from different channels recombined through the processing layers using **a low-latency, modular and scalable network infrastructure**
- Distributed online processing on heterogeneous computing devices in *n* subsequent layers.
- Features extraction will occur in the first NN layers on FPGAs
- More resource-demanding NN layers can be implemented in subsequent processing layers.
- Classification produced by the NN in last processing layer (e.g. pid) will be input for the trigger processor/storage **online data reduction stage for triggerless systems**

textarossa

Why FPGA are good to implement NN for real-time inference

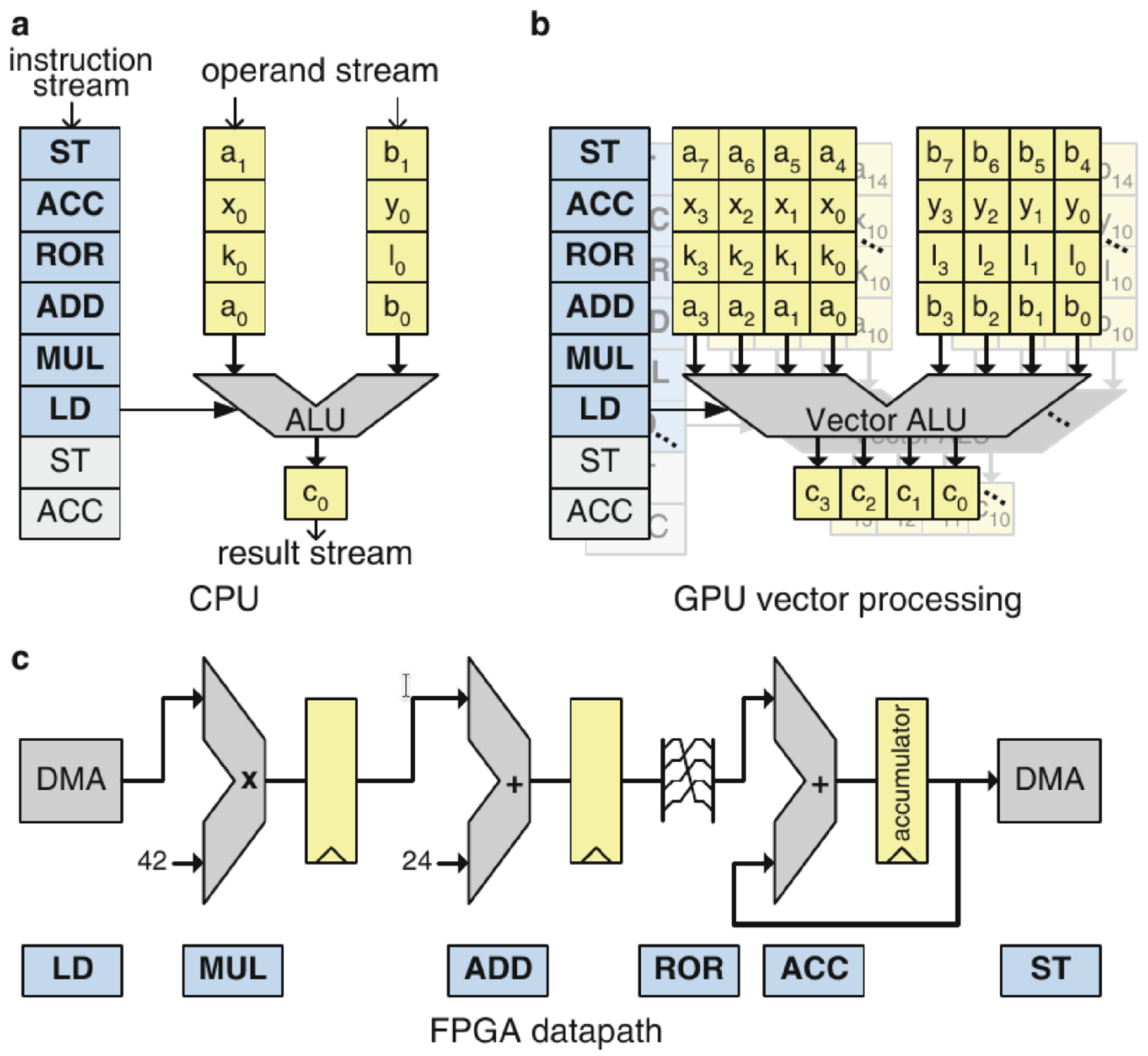
- FPGAs are chips made with a finite number of predefined resources with programmable interconnect to implement a reconfigurable digital circuit and I/O blocks
- FPGA adoption is driven by their flexibility, hardware-time speed, reliability and parallelism





- Customizable I/O and deterministic latency make them well suited expecially for TDAQ systems
- Improvements to silicon manufacturing process made them very interesting for heavy computation as well

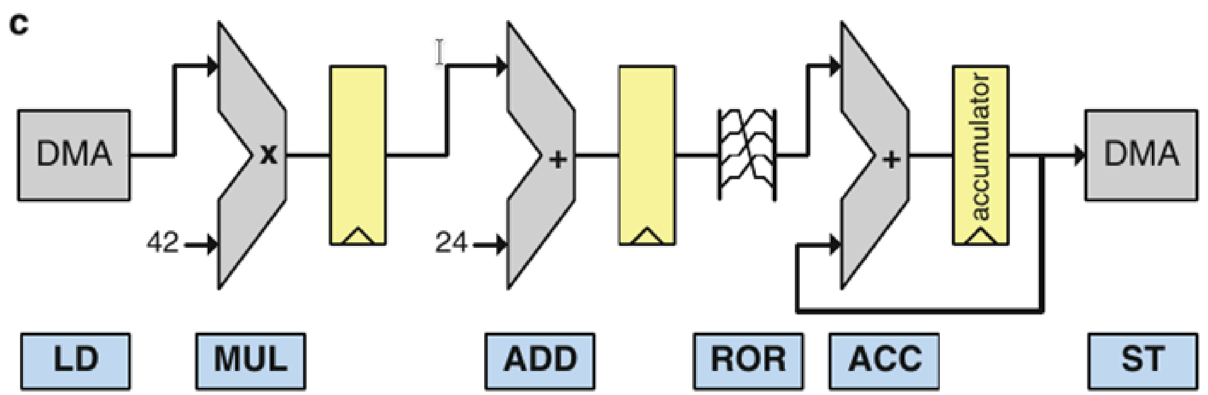
Dataflow Programming Model is Natural on FPGA



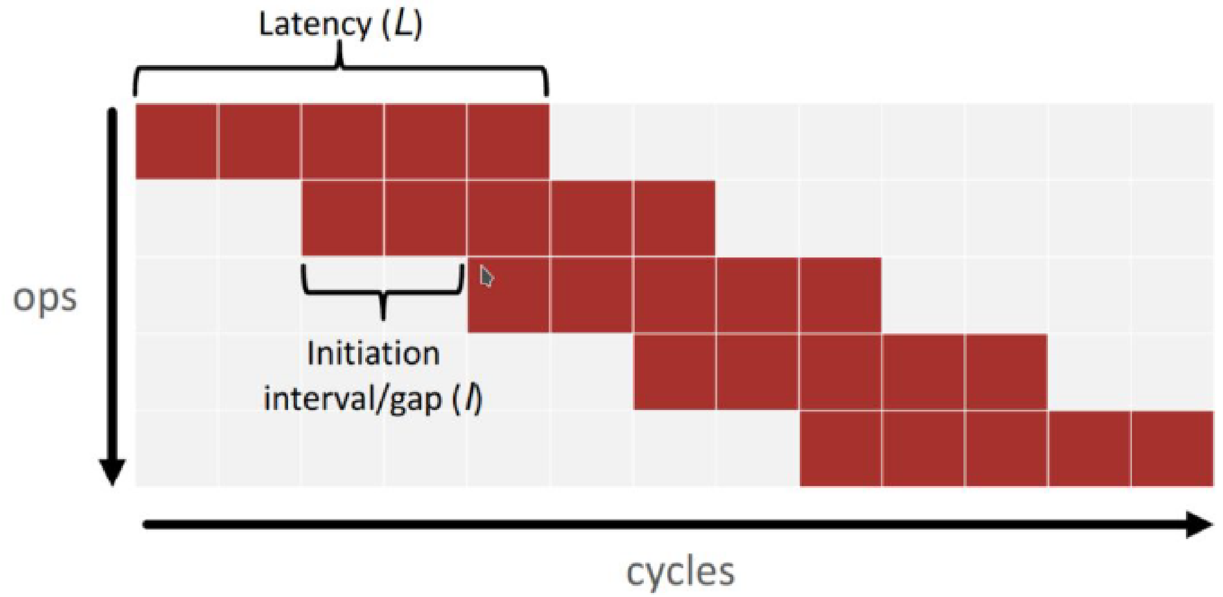
- Traditional Control Flow (CPU)
- Vector Processing (GPU...so to say)
- Dataflow paradigm (straightforward implementation on FPGA)
- FPGA: specialized hardware description languages (Verilog/VHDL) and design tools but recently...
- High level synthesis tools allows to describe datapaths in FPGA using high level software languages (C/C++).

Dataflow Programming Model is Natural on FPGA

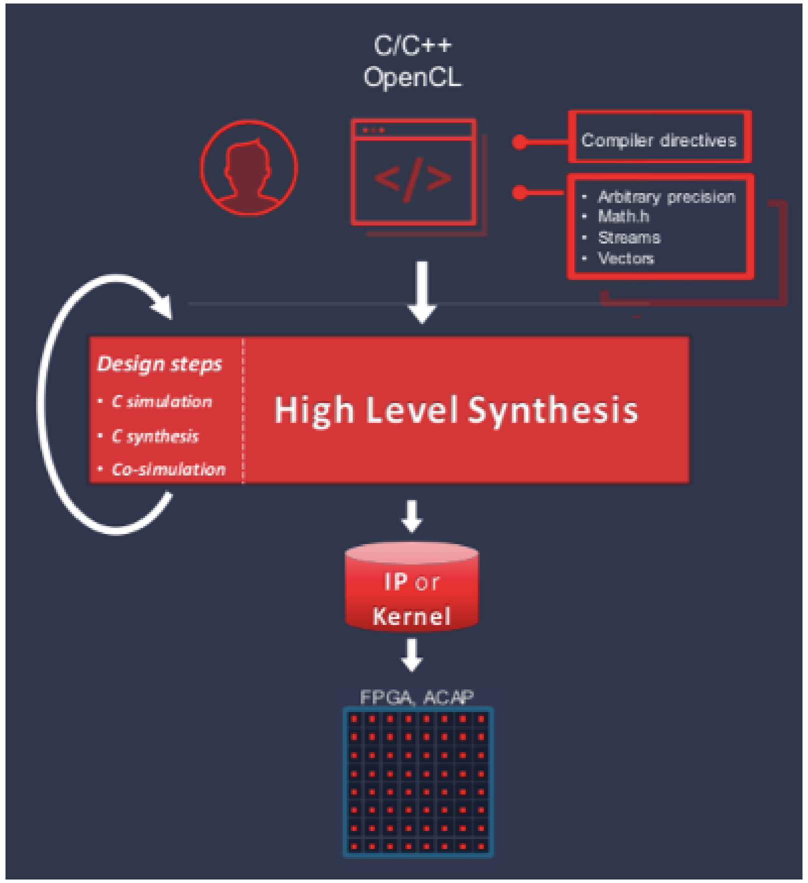
- Pipelined design can potentially produce a new output each clock cycle
- The greater the number of pipeline stages, the greater the latency
- ***Initiation interval (II)**:* Number of clock cycles before the function can accept new input data. The lower, the higher the throughput
- Once latency is overcome a pipelined design yields one output per II clock cycles, irrespective of the number of pipeline stages (parallelism between stages)
- Increasing throughput through parallelism (replicate the number of pipelines)



FPGA datapath



High Level Synthesis: write software, think hardware



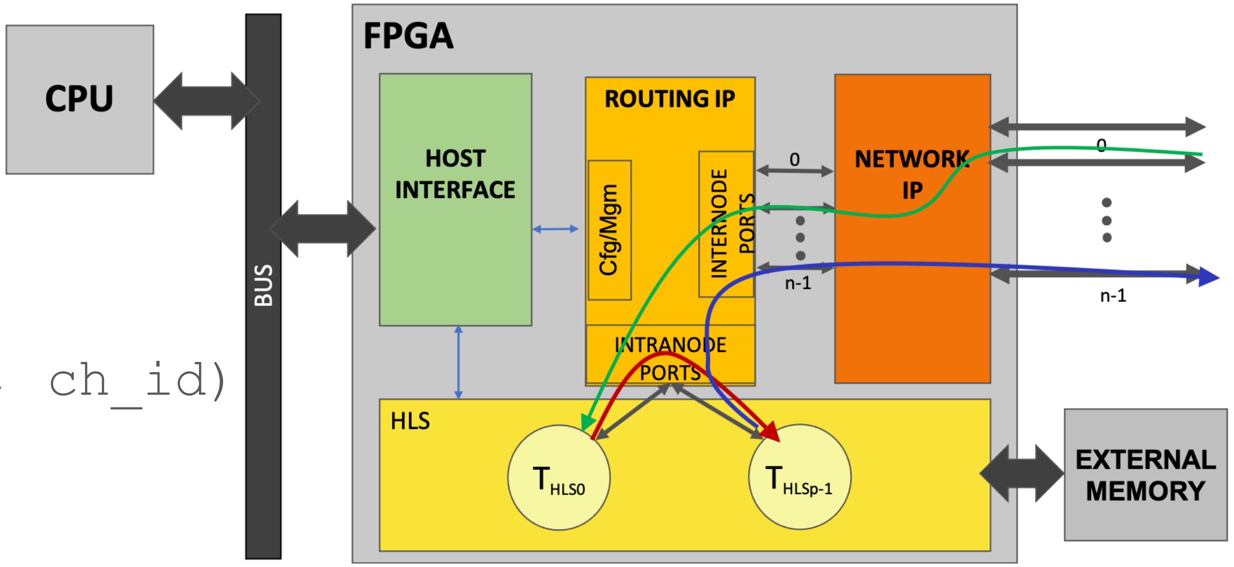
HLS flow representation by Xilinx

HLS: C based design entry

High level instructions transposed to hardware components
e..g. an array becomes an on-chip RAM
Code annotations (pragmas) can "guide" the compiler
e.g. array partition to access stored data in parallel

- Build tailored/custom processors
- Fast verification of the algorithm
- Detailed report about generated digital circuit
- Pragmas determines circuit topology

APEIRON HLS C++ Communication Primitives



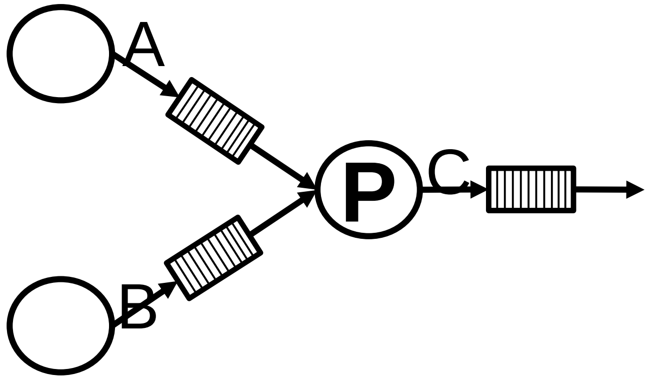
- APEIRON C++ HLS API
  - **send**(msg, size, dest_node, task_id, ch_id)
  - **receive**(ch_id)

  Where :

  `dest_node` are the n-Dim coordinates of the destination node (FPGA) in a n-Dim torus network.

  `task_id` is the local-to-node receiving task (kernel) identifier (0-3, 0-7).

  `ch_id` is the local-to-task receiving fifo (channel) identifier (0-127).

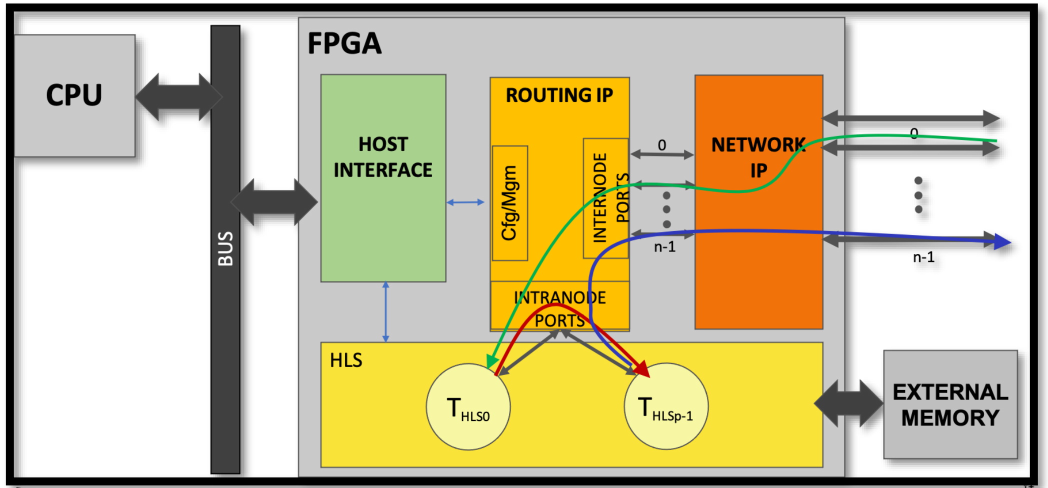


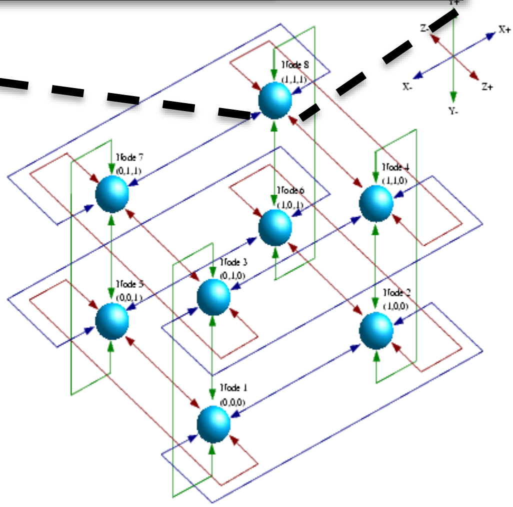
APEIRON HW IPs for low-latency FPGA communication

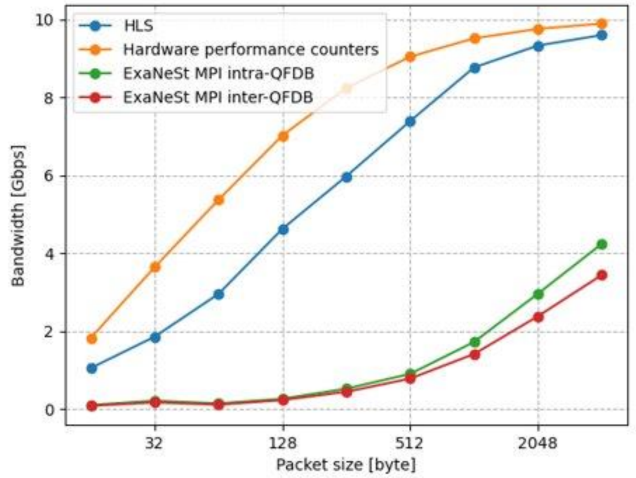
IPs implementing a direct network enabling low-latency communication between processing tasks deployed:

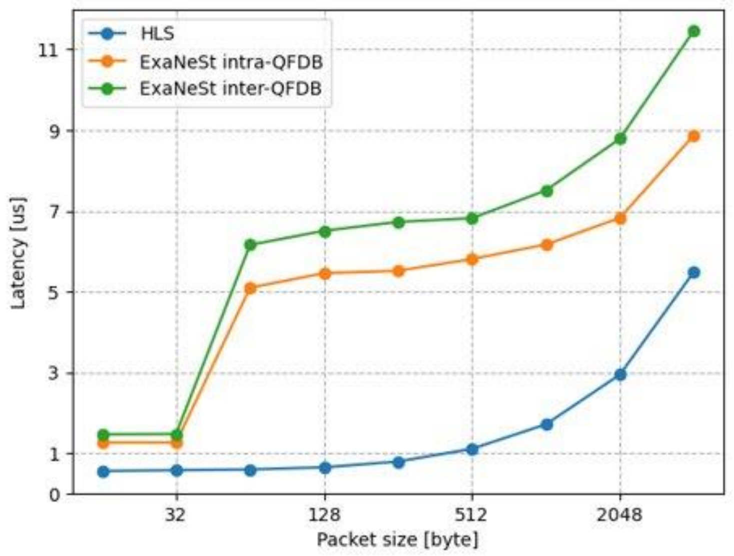
- on the same FPGA (**intra-node comm.**)
- on different FPGAs (**inter-node comm.**)

- **Host Interface IP:** Interface the FPGA logic with the host through the system bus.
  - PCI Express Gen3 → Gen4
- **Network IP:** Network channels and Application-dependent I/O
  - APElink 32 Gbps → 64/100 Gbps
  - UDP/IP over 1/10 GbE → 40/100 GbE
- **Routing IP:** Routing of intra-node and inter-node messages between processing tasks on FPGA.





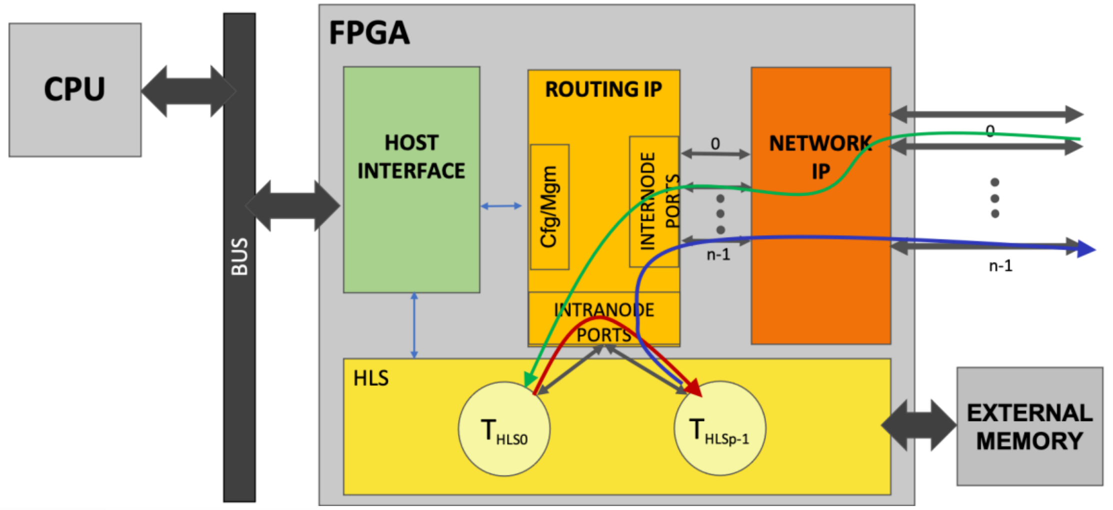




APEIRON – How to use it

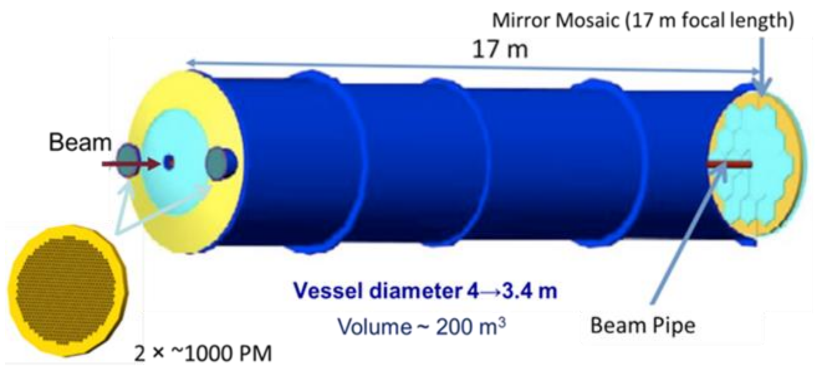
- For Xilinx FPGAs, APEIRON leverages the Vitis flow to generate the bitstream
- HLS kernels are written in C++, there are no particular restrictions, apart from the top-level interfaces (channels)
- A YAML configuration file is used to describe the kernels interconnection topology, specifying how many input/output channels they have

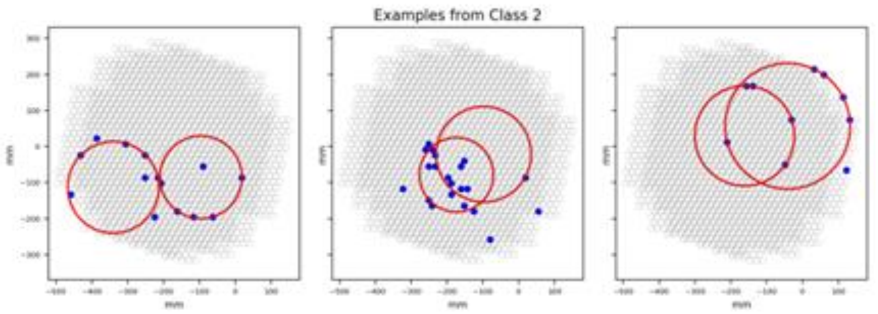
```yaml
kernels:
  - name: krnl_compute1
    input_channels: 4
    output_channels: 3
    switch_port: 1

  - name: krnl_compute2
    input_channels: 2
    output_channels: 1
    switch_port: 2

  - name: krnl_compute3
    input_channels: 1
    output_channels: 1
    switch_port: 3

  - name: krnl_compute4
    input_channels: 1
    output_channels: 1
    switch_port: 4
```

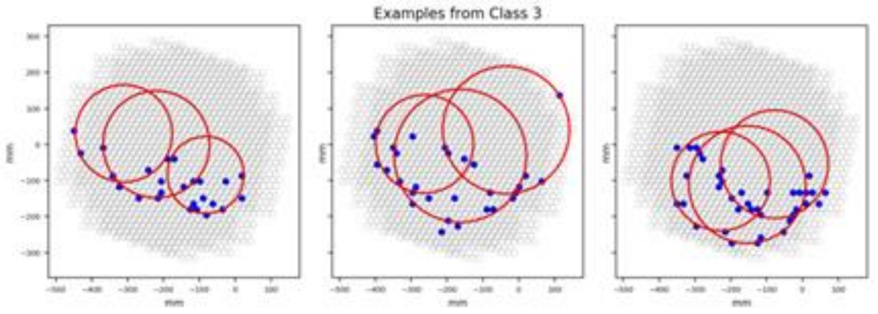


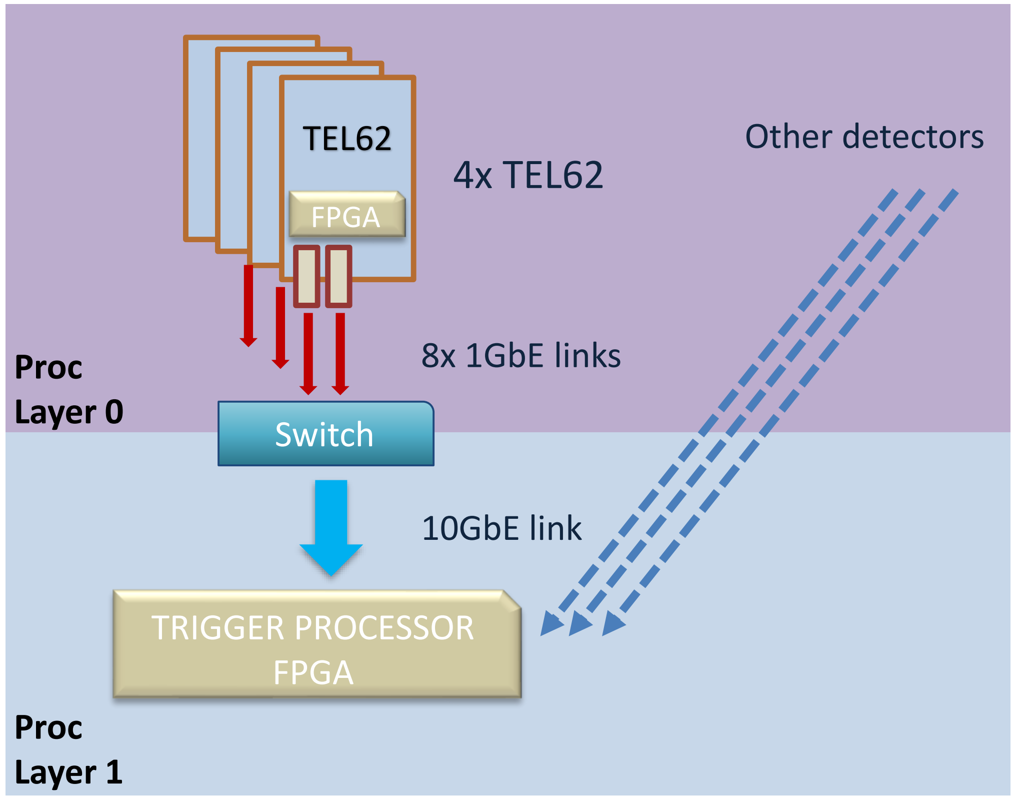
APEIRON use case: Partial Particle ID with the NA62 RICH

- Located at the CERN SPS
- Measure rare kaon decay:
  - $k \rightarrow \pi \nu \nu$ with $BR(k \rightarrow \pi \nu \nu) = (8.4 \pm 1.0) \times 10^{-11}$
- Nominal event rate at L0: 10MHz
- Number of Cherenkov rings is a good candidate to improve L0 decisions: can we achieve required throughput with a good accuracy?





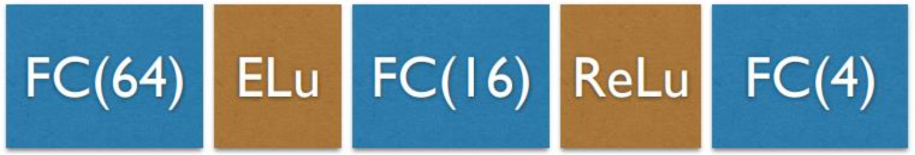


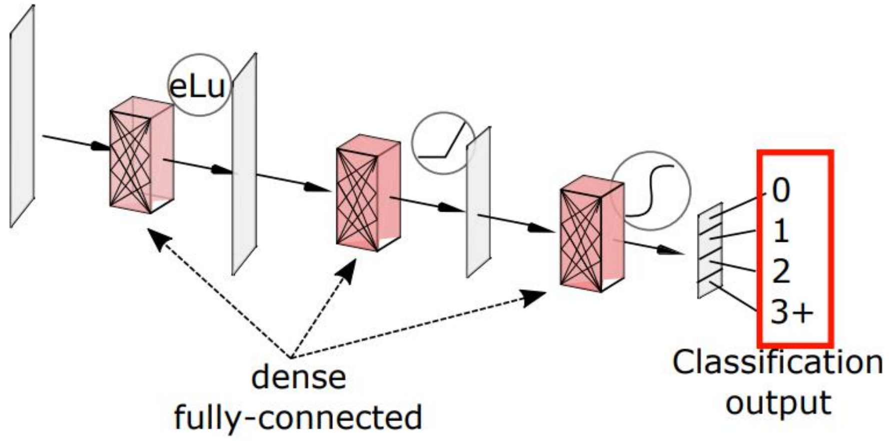


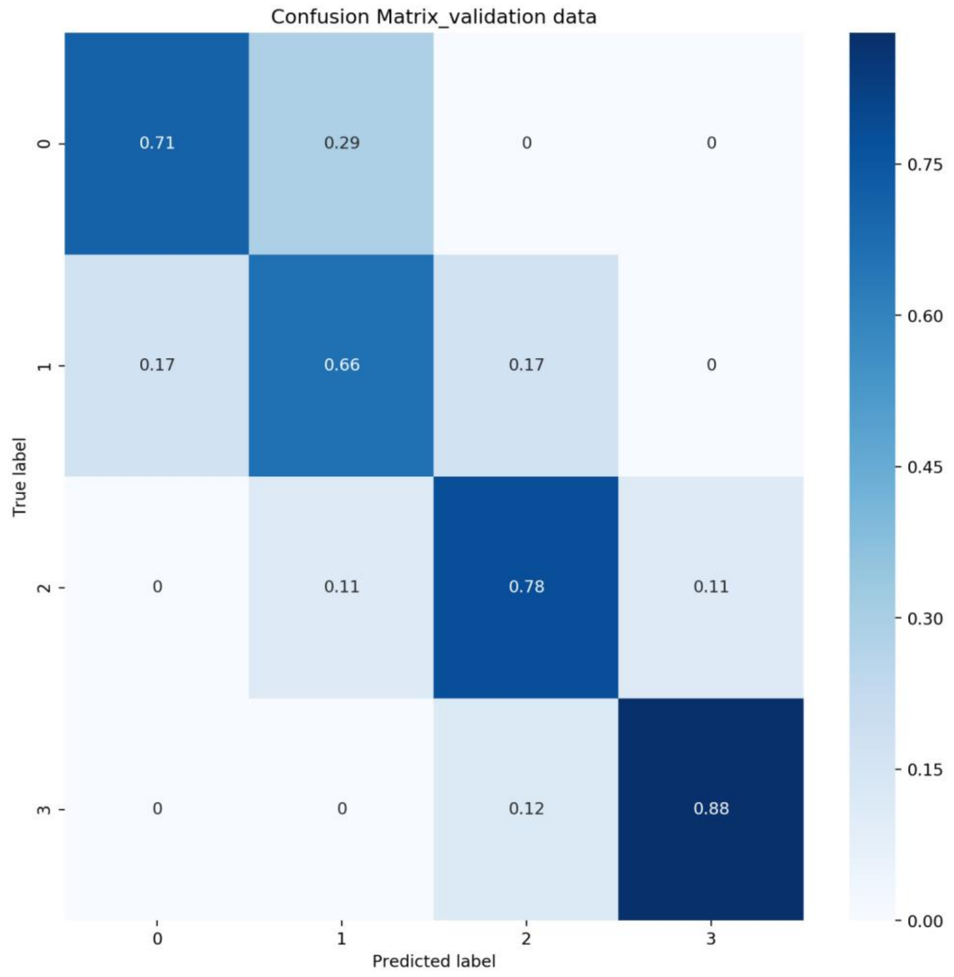
Rings detection - Dense model

Fully Connected

- Input: 64 hits per event
- Architecture: 3 fully connected layers
- Output: 4 classes (0, 1, 2, 3+ rings per event)
- Qkeras, quantization aware training:
  - ~75% average accuracy with low resource usage: LUT 14%, DSP 2%, BRAM 0% (VCU118)
- Latency: 22 cycles @ 150MHz
- Initiation Interval (II): 8 cycles







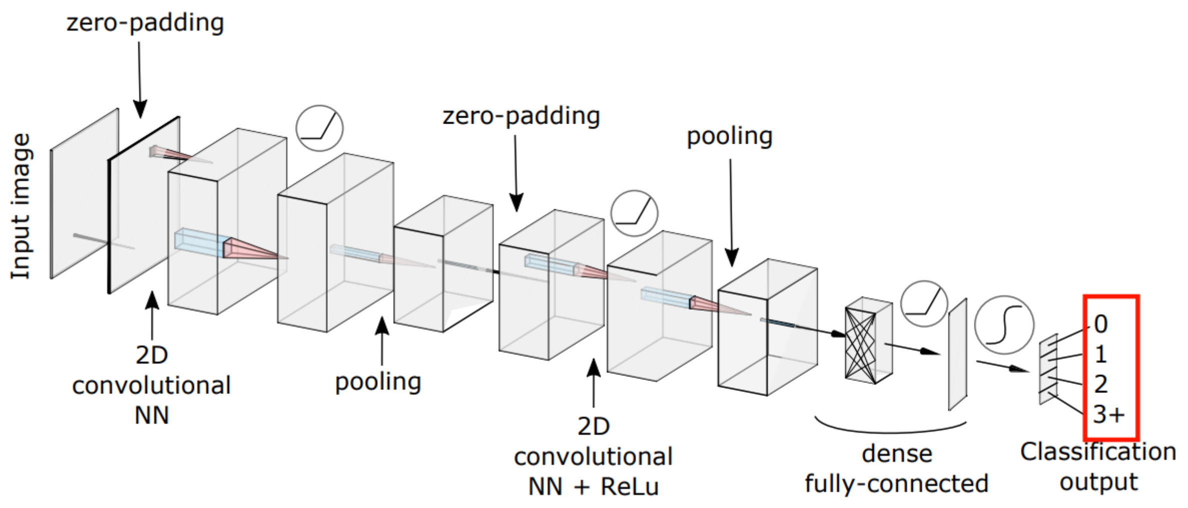
True label \ Predicted label	0	1	2	3
0	0.71	0.29	0	0
1	0.17	0.66	0.17	0
2	0	0.11	0.78	0.11
3	0	0	0.12	0.88

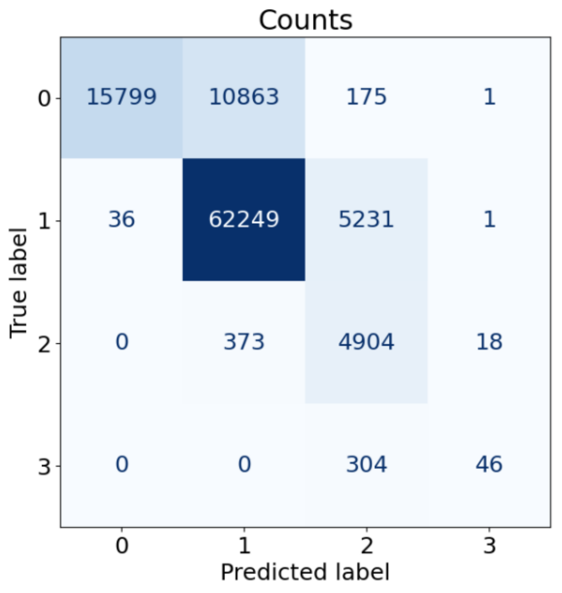
TensorFlow ➡ hls 4 ml ➡ VIVADO HLS ➡ IP integration in L0TP+ infrastructure

https://fastmachinelearning.org/hls4ml/

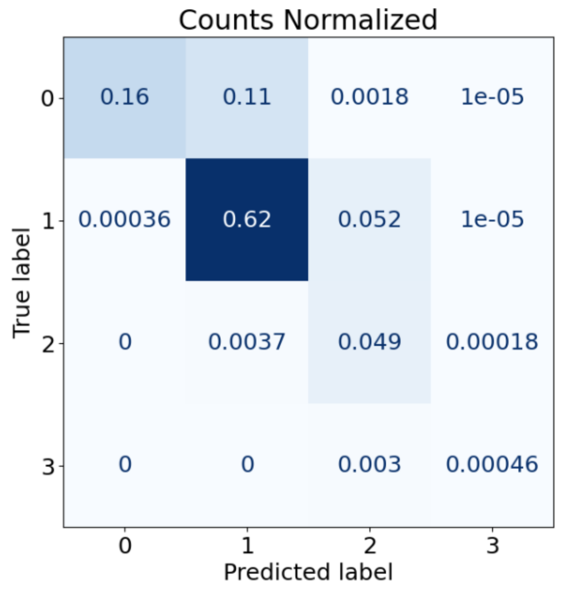
Rings detection - Convolutional model

- Input: PMT channels into image 16x16 pixels
- Architecture: 2 conv layers and 2 dense
- Output: 4 classes (0, 1, 2, 3+ rings per event)
- Qkeras, quantization aware training:
  - ~83% average accuracy with low footprint: LUT 5.2%, FF 1.5%, DSP 4.8%, BRAM 0.05% (Alveo U200)
- Latency: 388 cycles @ 220MHz
- Initiation Interval (II): 369 cycles

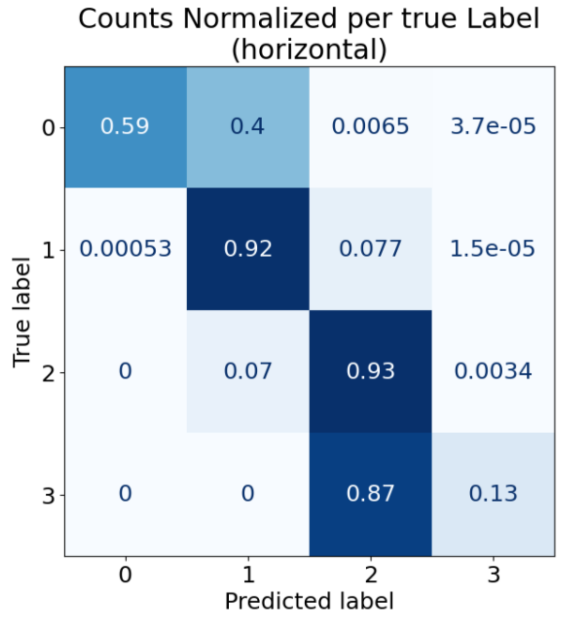




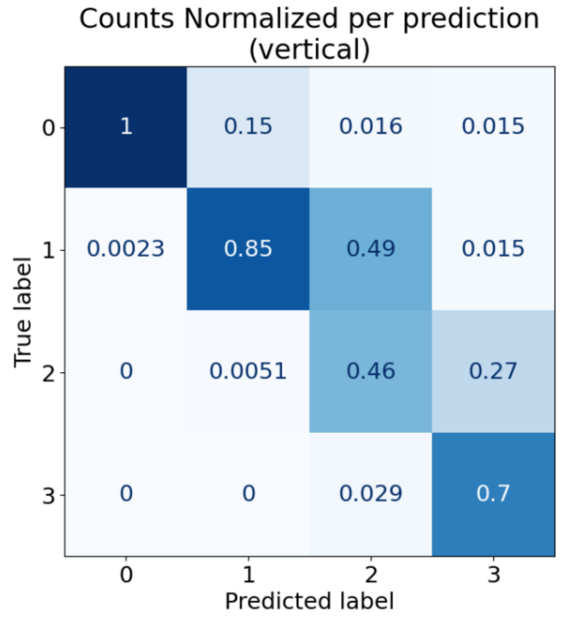
True \ Predicted	0	1	2	3
0	15799	10863	175	1
1	36	62249	5231	1
2	0	373	4904	18
3	0	0	304	46



True \ Predicted	0	1	2	3
0	0.16	0.11	0.0018	1e-05
1	0.00036	0.62	0.052	1e-05
2	0	0.0037	0.049	0.00018
3	0	0	0.003	0.00046



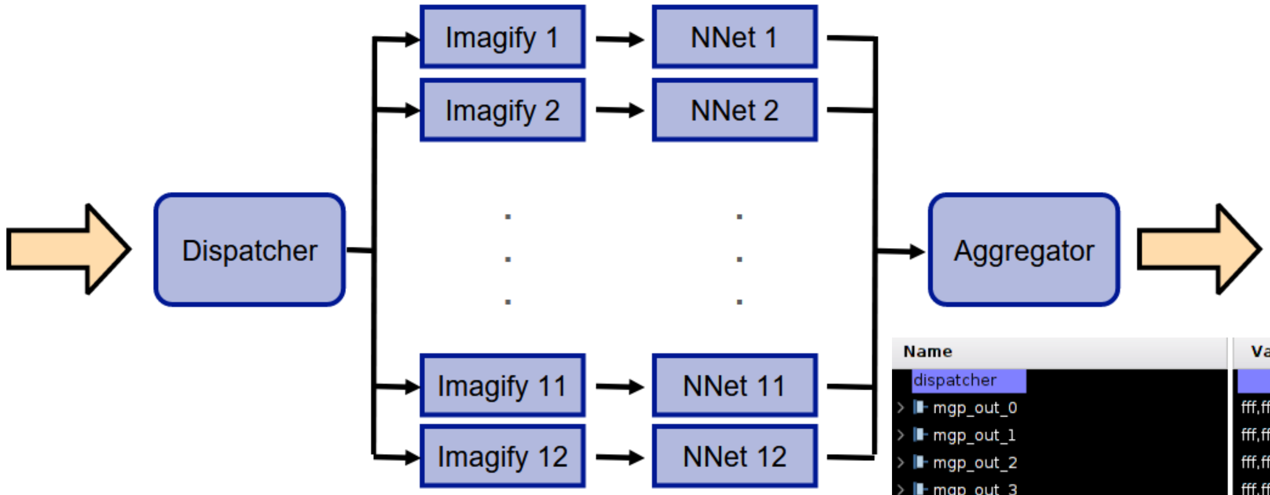
True \ Predicted	0	1	2	3
0	0.59	0.4	0.0065	3.7e-05
1	0.00053	0.92	0.077	1.5e-05
2	0	0.07	0.93	0.0034
3	0	0	0.87	0.13

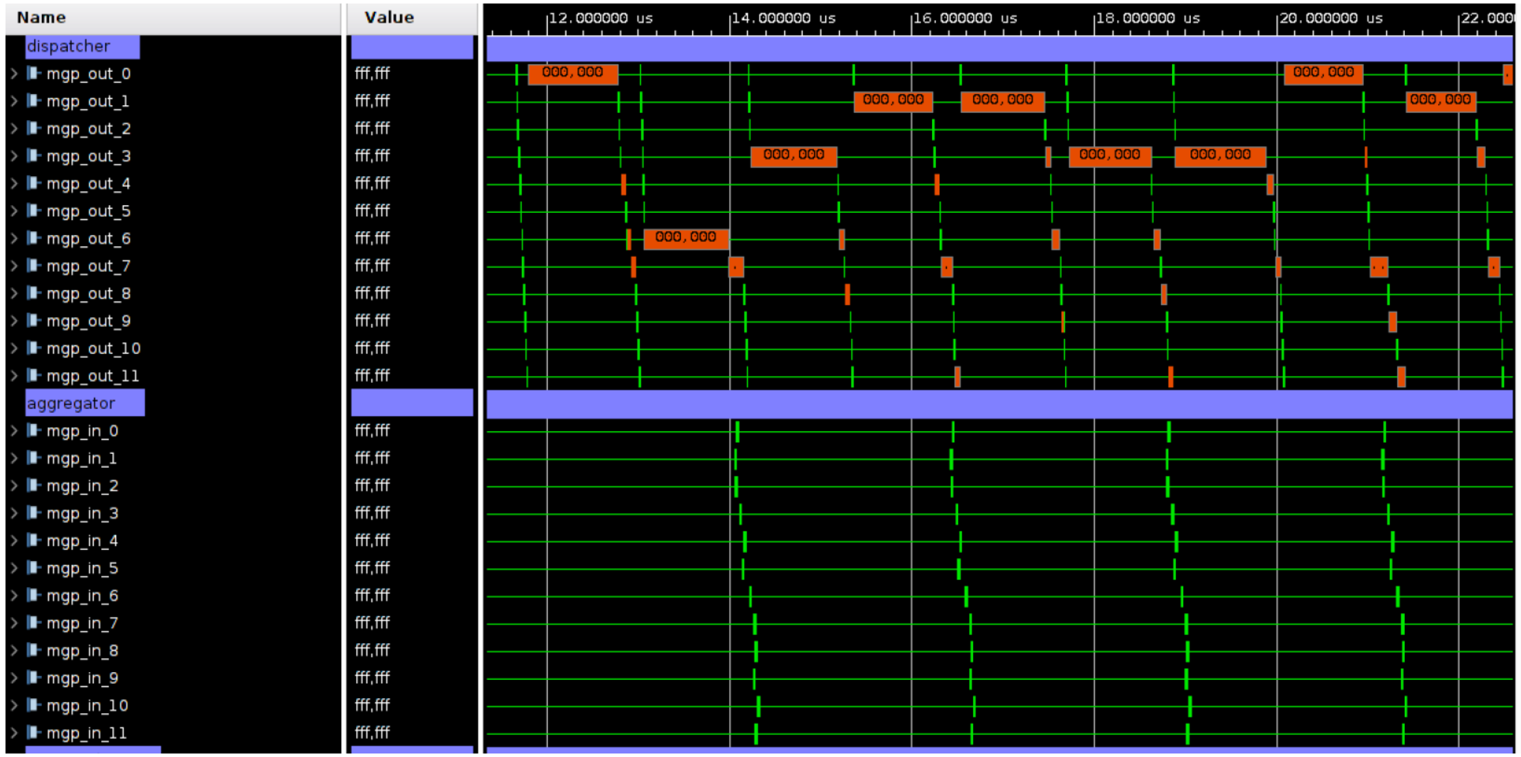


True \ Predicted	0	1	2	3
0	1	0.15	0.016	0.015
1	0.0023	0.85	0.49	0.015
2	0	0.0051	0.46	0.27
3	0	0	0.029	0.7

Convolutional model – Kernel replication

Throughput is not enough to sustain L0 rate, but we can replicate the network multiple times, also on multiple devices if necessary.





```
Name            Value
dispatcher
mgp_out_0       fff,fff
mgp_out_1       fff,fff
mgp_out_2       fff,fff
mgp_out_3       fff,fff
mgp_out_4       fff,fff
mgp_out_5       fff,fff
mgp_out_6       fff,fff
mgp_out_7       fff,fff
mgp_out_8       fff,fff
mgp_out_9       fff,fff
mgp_out_10      fff,fff
mgp_out_11      fff,fff
aggregator
mgp_in_0        fff,fff
mgp_in_1        fff,fff
mgp_in_2        fff,fff
mgp_in_3        fff,fff
mgp_in_4        fff,fff
mgp_in_5        fff,fff
mgp_in_6        fff,fff
mgp_in_7        fff,fff
mgp_in_8        fff,fff
mgp_in_9        fff,fff
mgp_in_10       fff,fff
mgp_in_11       fff,fff
```

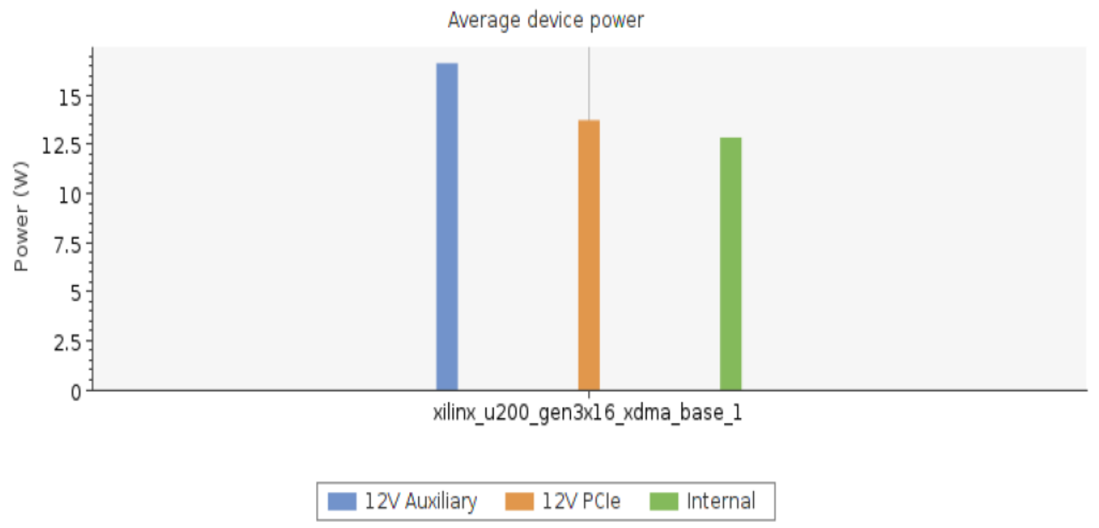
Resources usage for 12 replicas:

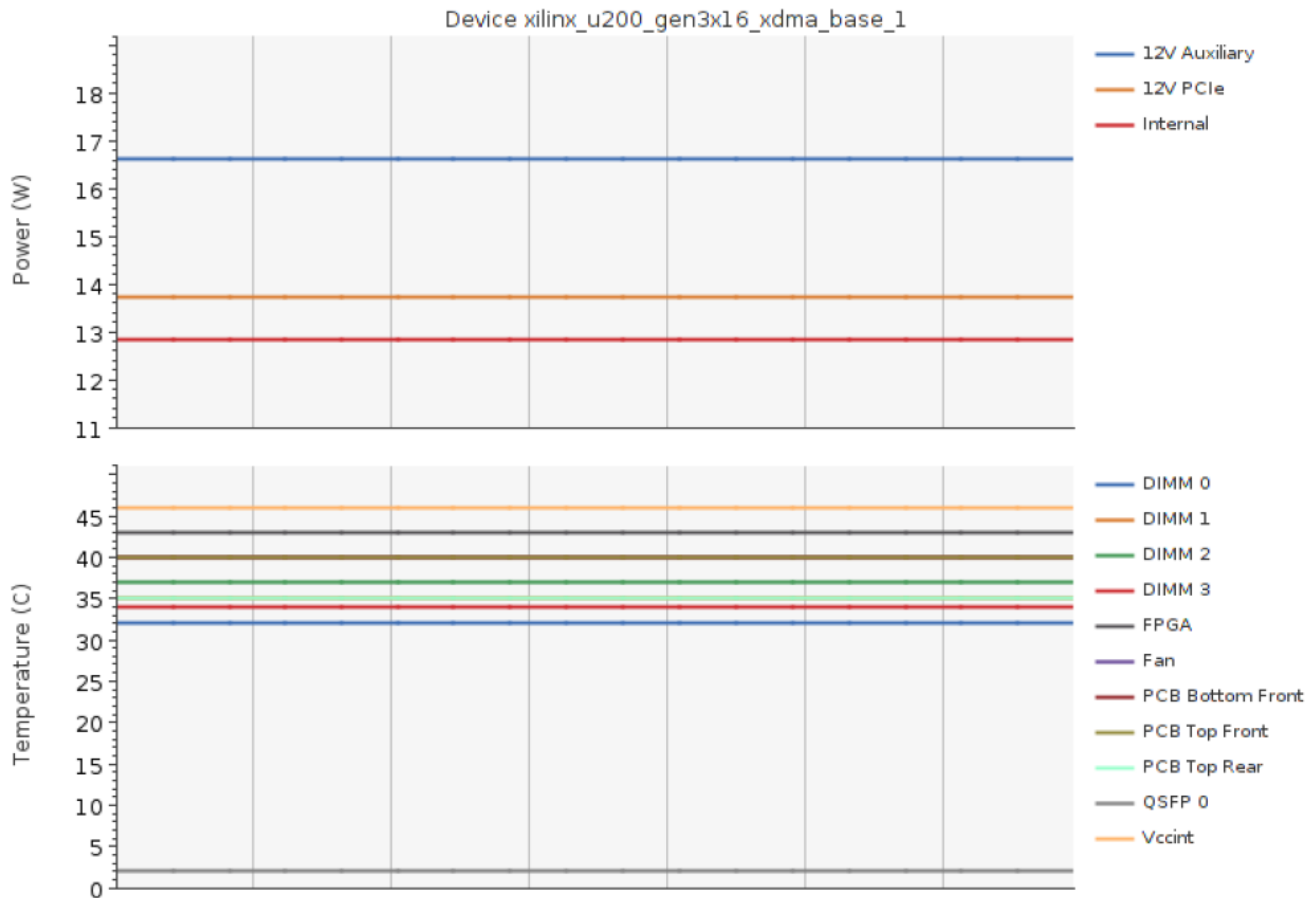
- LUT 74%
- FF 17%
- DPS 61%
- BRAM 1.4%

Processing time @220MHz: 137 ns per event

Energy efficiency

Low energy consumption compared to other devices (e.g. TDP for GPUs is in the order of hundreds of Watts)



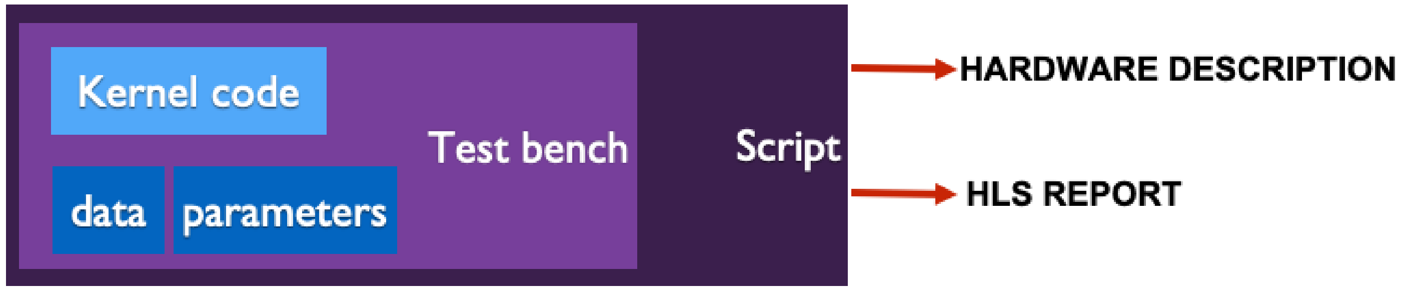


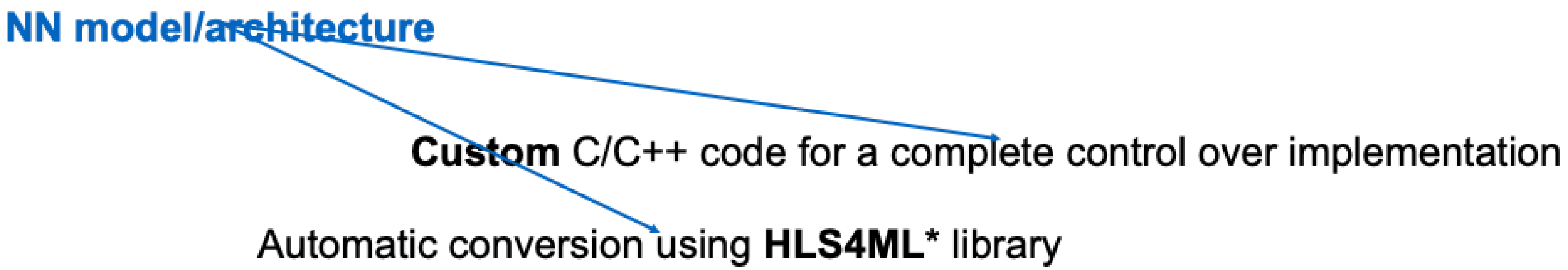
▪ Backup Slides

Fare clic per inserire testo

APEIRON Workflow for Neural Network Tasks

1.Bottom-up	Fully connected, convolutional, activations, precision
2.Design Space Exploration	Size, random data and weights, easy/corner cases
3.Automation	Unassisted operations (e.g. scans)





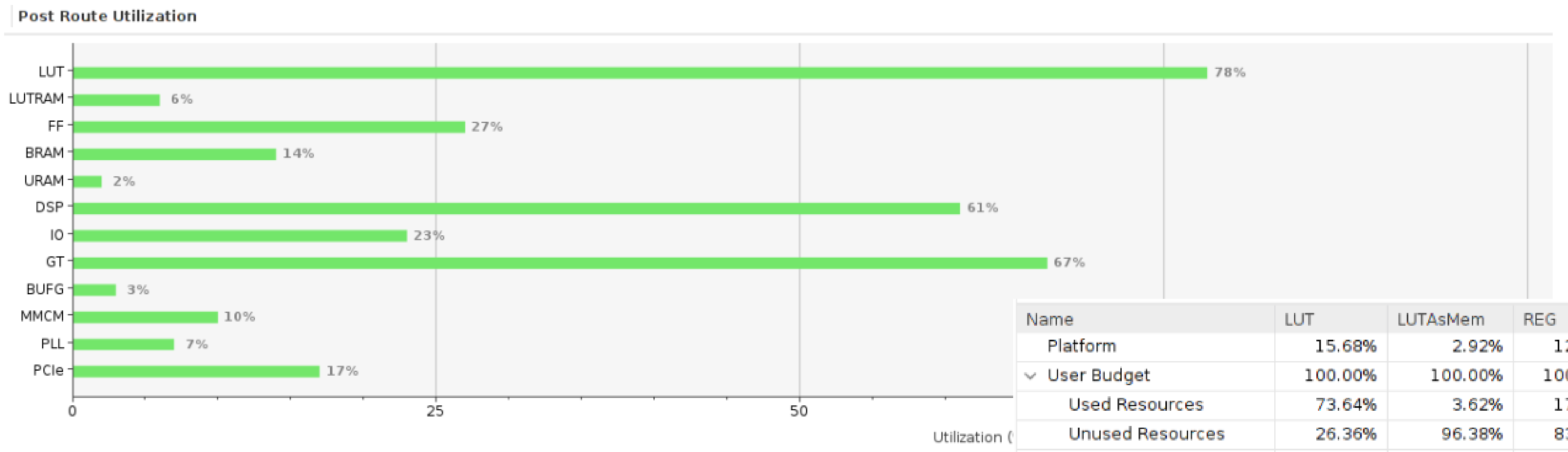
Custom C/C++ code for a complete control over implementation

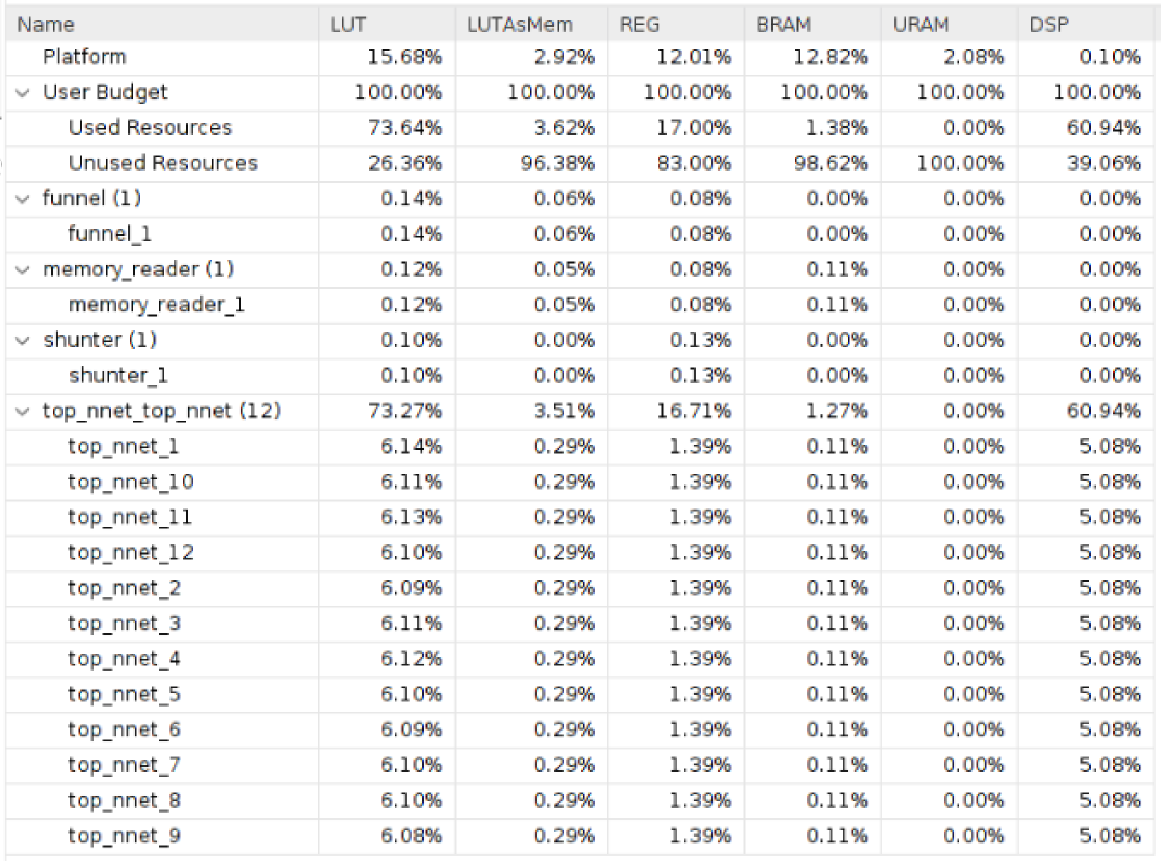
Automatic conversion using **HLS4ML*** library

* https://github.com/fastmachinelearning/hls4ml

textarossa

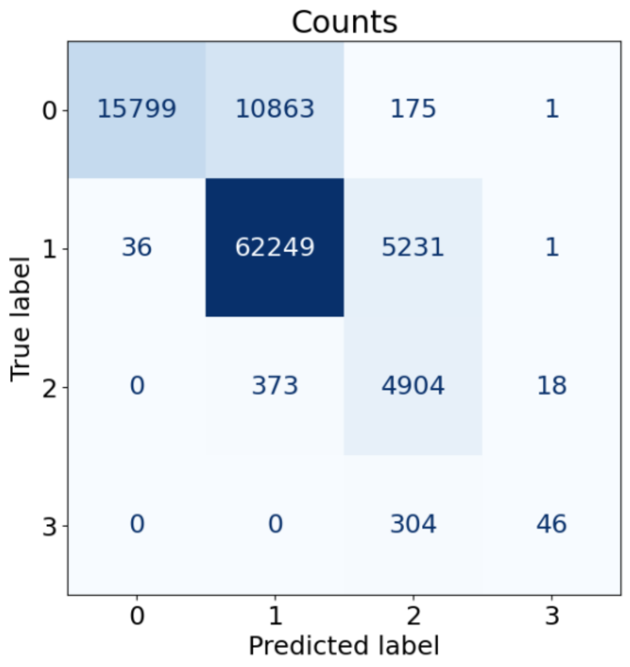
Resources usage



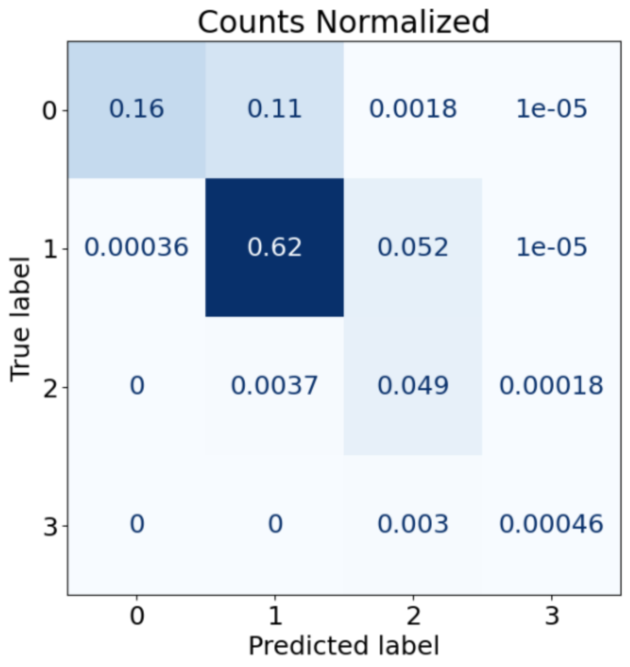


Name	LUT	LUTAsMem	REG	BRAM	URAM	DSP
Platform	15.68%	2.92%	12.01%	12.82%	2.08%	0.10%
User Budget	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Used Resources	73.64%	3.62%	17.00%	1.38%	0.00%	60.94%
Unused Resources	26.36%	96.38%	83.00%	98.62%	100.00%	39.06%
funnel (1)	0.14%	0.06%	0.08%	0.00%	0.00%	0.00%
funnel_1	0.14%	0.06%	0.08%	0.00%	0.00%	0.00%
memory_reader (1)	0.12%	0.05%	0.08%	0.11%	0.00%	0.00%
memory_reader_1	0.12%	0.05%	0.08%	0.11%	0.00%	0.00%
shunter (1)	0.10%	0.00%	0.13%	0.00%	0.00%	0.00%
shunter_1	0.10%	0.00%	0.13%	0.00%	0.00%	0.00%
top_nnet_top_nnet (12)	73.27%	3.51%	16.71%	1.27%	0.00%	60.94%
top_nnet_1	6.14%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_10	6.11%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_11	6.13%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_12	6.10%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_2	6.09%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_3	6.11%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_4	6.12%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_5	6.10%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_6	6.09%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_7	6.10%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_8	6.10%	0.29%	1.39%	0.11%	0.00%	5.08%
top_nnet_9	6.08%	0.29%	1.39%	0.11%	0.00%	5.08%

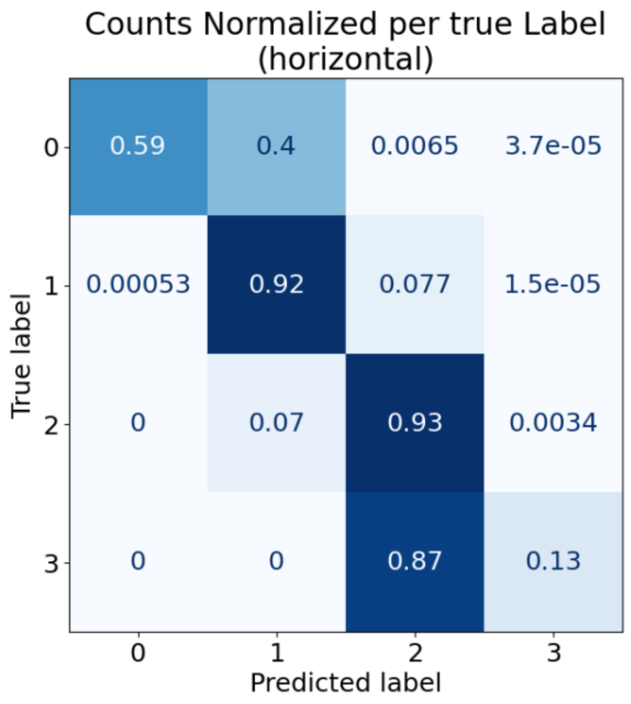
Solo i primi 100k



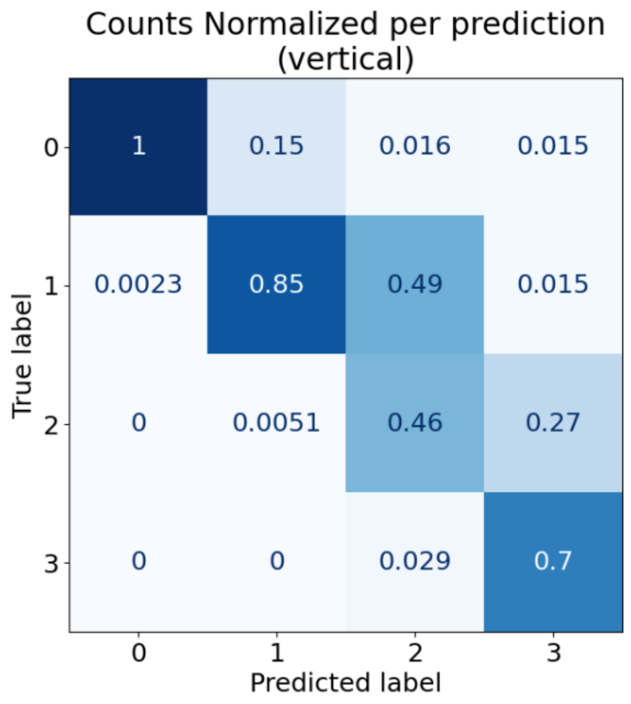
True label \ Predicted label	0	1	2	3
0	15799	10863	175	1
1	36	62249	5231	1
2	0	373	4904	18
3	0	0	304	46



True label \ Predicted label	0	1	2	3
0	0.16	0.11	0.0018	1e-05
1	0.00036	0.62	0.052	1e-05
2	0	0.0037	0.049	0.00018
3	0	0	0.003	0.00046

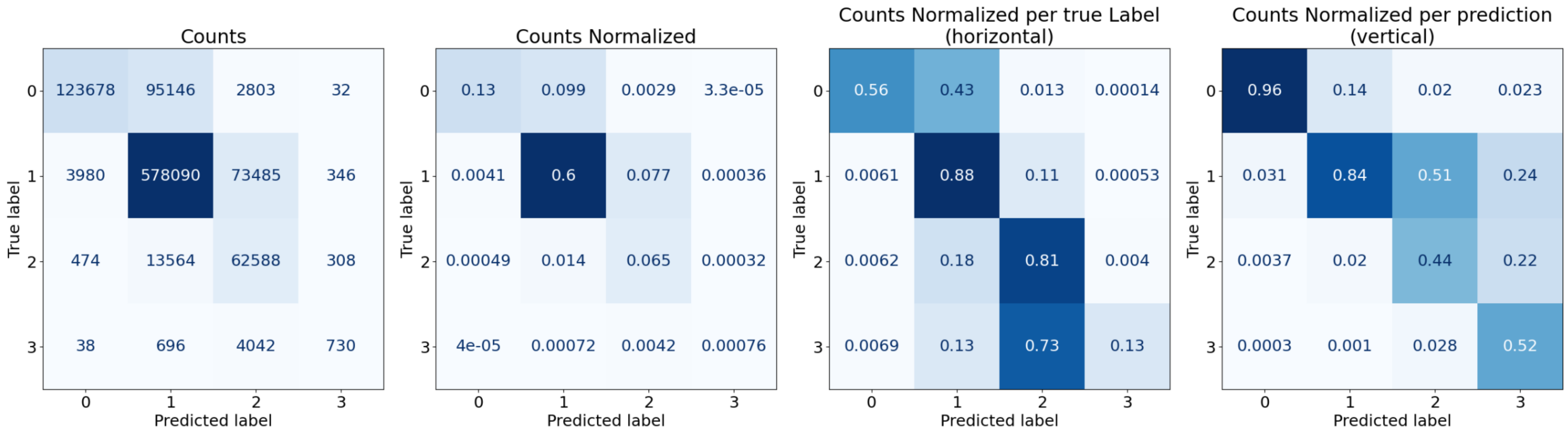


True label \ Predicted label	0	1	2	3
0	0.59	0.4	0.0065	3.7e-05
1	0.00053	0.92	0.077	1.5e-05
2	0	0.07	0.93	0.0034
3	0	0	0.87	0.13



True label \ Predicted label	0	1	2	3
0	1	0.15	0.016	0.015
1	0.0023	0.85	0.49	0.015
2	0	0.0051	0.46	0.27
3	0	0	0.029	0.7

Tutti gli eventi



Counts

True label \ Predicted label	0	1	2	3
0	123678	95146	2803	32
1	3980	578090	73485	346
2	474	13564	62588	308
3	38	696	4042	730

Counts Normalized

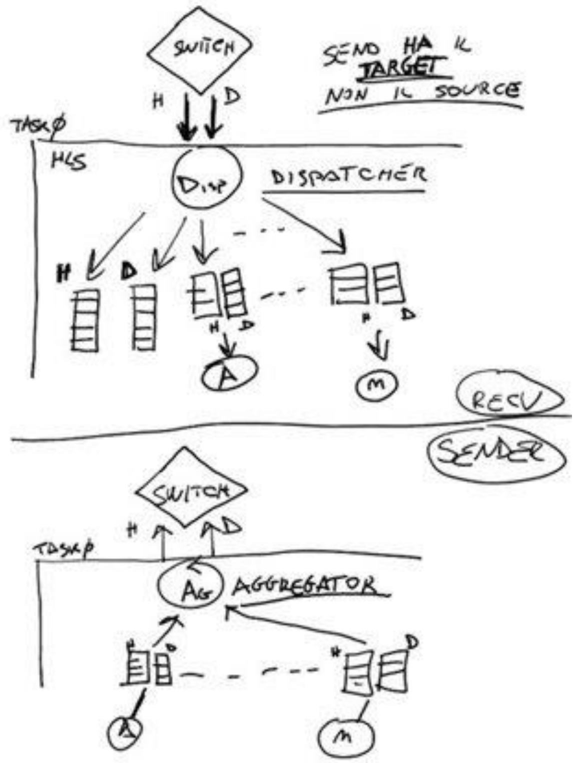
True label \ Predicted label	0	1	2	3
0	0.13	0.099	0.0029	3.3e-05
1	0.0041	0.6	0.077	0.00036
2	0.00049	0.014	0.065	0.00032
3	4e-05	0.00072	0.0042	0.00076

Counts Normalized per true Label (horizontal)

True label \ Predicted label	0	1	2	3
0	0.56	0.43	0.013	0.00014
1	0.0061	0.88	0.11	0.00053
2	0.0062	0.18	0.81	0.004
3	0.0069	0.13	0.73	0.13

Counts Normalized per prediction (vertical)

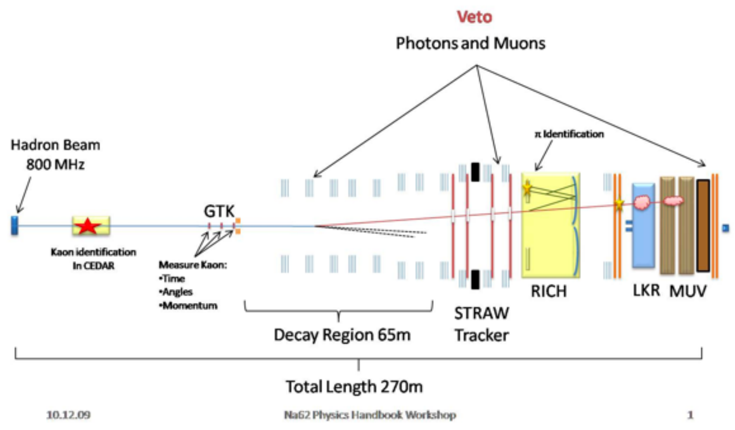
True label \ Predicted label	0	1	2	3
0	0.96	0.14	0.02	0.023
1	0.031	0.84	0.51	0.24
2	0.0037	0.02	0.44	0.22
3	0.0003	0.001	0.028	0.52


SWITCH
SEND HA IL TARGET NON IL SOURCE
H
D
TASKφ
HLS
Disp
DISPATCHER
H
D
H
D
A
H
D
M
RECV
SENDER
SWITCH
H
D
TASKφ
AG
AGGREGATOR
H
D
H
D
A
M

NA62

NA62 Experiment

- Located at the CERN SPS
- Measure rare kaon decay:
  - $k \rightarrow \pi\nu\nu$ with BR($k \rightarrow \pi\nu\nu$) = $(8.4 \pm 1.0) \times 10^{-11}$
- ~60% of nominal intensity in 2017-18 data taking, 100% in 2021

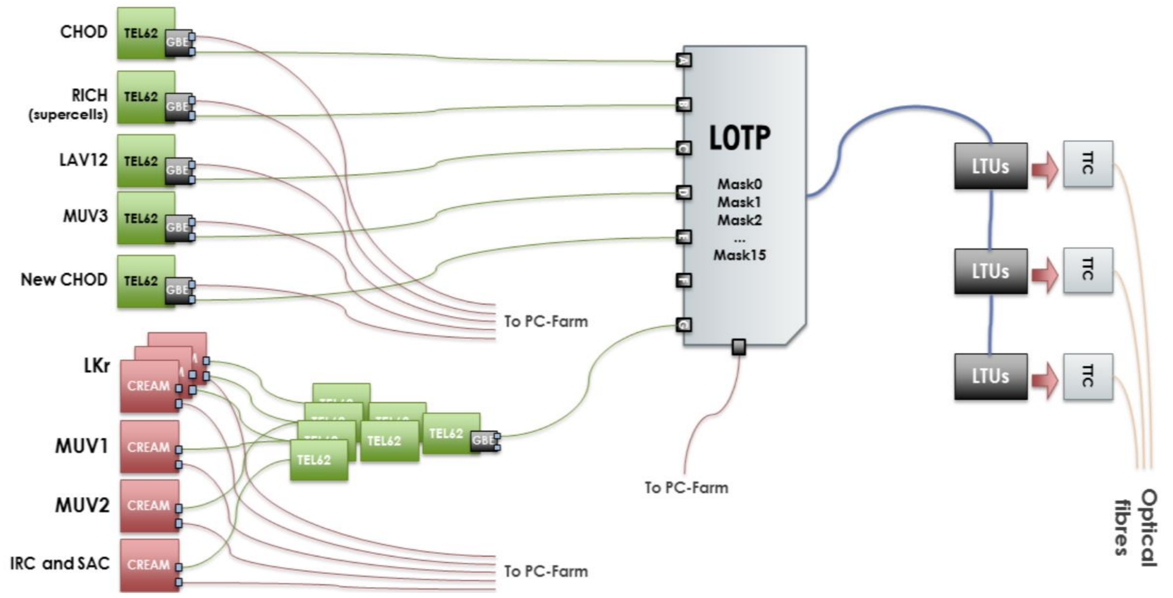


Multi-level trigger

- Level-0 (L0TP) HW trigger on FPGA (10MHz -> 1MHz)
- Level-1 software trigger running in DAQ farm (1MHz -> 100kHz)
- Level-2

DAQ

- Data bursts are ~6s long
- Some detectors primitives, generated from TEL62 readout boards, are sent to L0TP
- L0TP generates trigger with max latency of 1ms
- L2 trigger run over the complete event information



CHOD
RICH
(supercells)
LAV12
MUV3
New CHOD
LKr
MUV1
MUV2
IRC and SAC
TEL62
GBE
CREAM
L0TP
Mask0
Mask1
Mask2
...
Mask15
LTUs
TTC
To PC-Farm
To PC-Farm
To PC-Farm
Optical fibres