APEIRON: a framework for the development of smart TDAQ systems

(INFN Sezione di Roma - APE Lab)

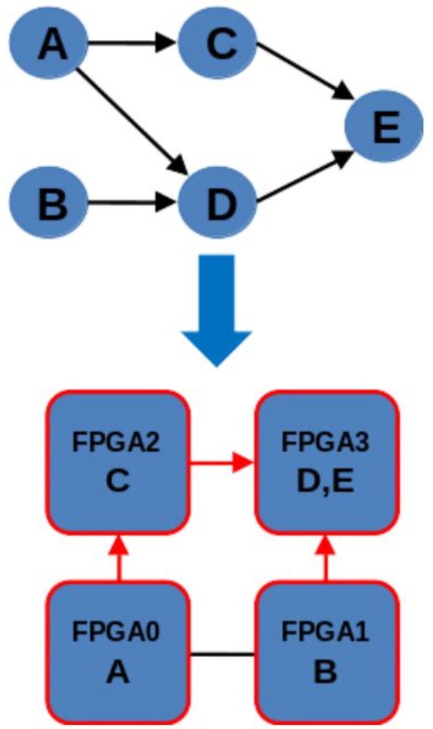
Speaker: Cristian Rossi
(cristian.rossi@roma1.infn.it)

APEIRON: overview

APEIRON is a framework developed to offer hardware and software support for the execution of real-time dataflow applications on a system composed by interconnected FPGAs

- Enabling the mapping the dataflow graph of the application on the distributed FPGA system and offering runtime support for the execution.
- Allowing users, with no (or little) 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 (Xilinx® Vitis).
  - Lightweight C++ communication API (HAPECOM)
    - Non-blocking *send()*
    - Blocking *receive()*

APEIRON enables the scaling of Xilinx® Vitis High Level Synthesis applications on multiple FPGA interconnected by the INFN communication IP.

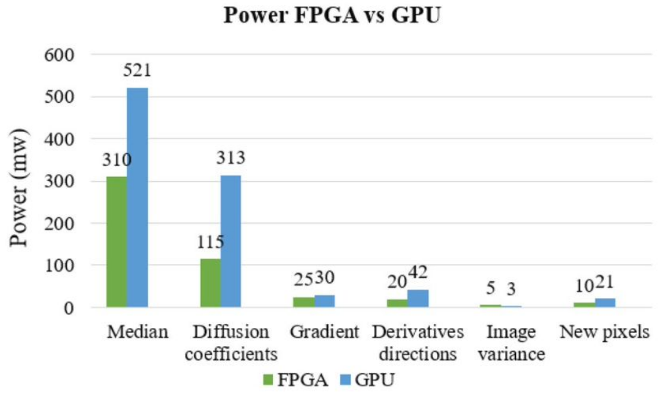


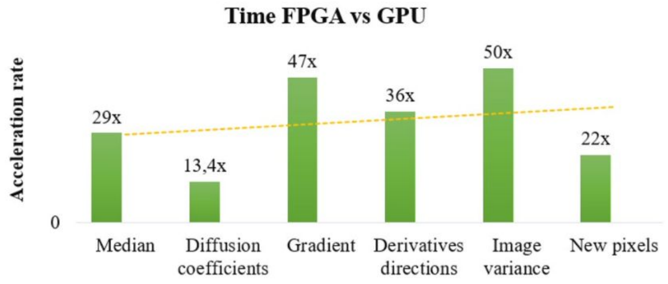
Why using FPGA?

➢ Energy efficiency

- **Energy Efficiency** has become an important metric of performance in recent years. The application and computing scale is increasing exponentially every year leading to an enormous amount of data to be processed
  ⇒ high power and energy consumption.

- FPGA architectures, together with a high programmability, offer a good balance in terms of **performance** and **energy efficiency** without sacrificing the throughput of the application.





JRTIP (2021) 18:2429–2440

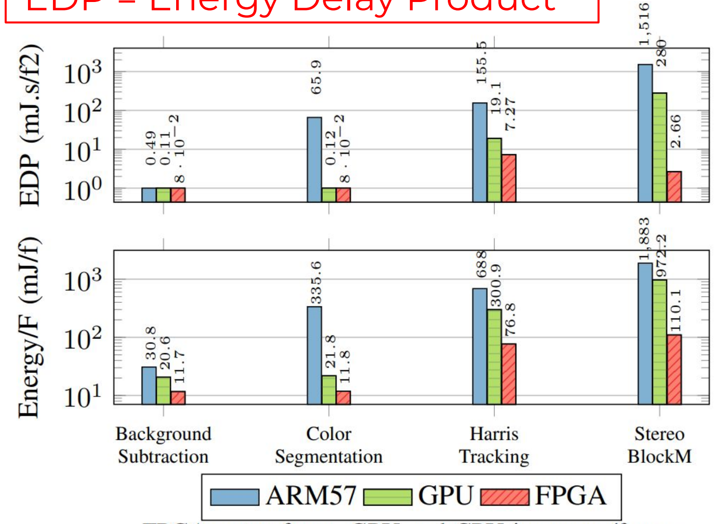
Why using FPGA?

➢ Energy efficiency

- **Energy Efficiency** has become an important metric of performance in recent years. The application and computing scale is increasing exponentially every year leading to an enormous amount of data to be processed
  ⇒ high power and energy consumption.

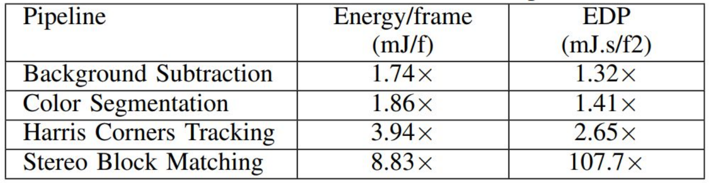
- FPGA architectures, together with a high programmability, offer a good balance in terms of **performance** and **energy efficiency** without sacrificing the throughput of the application.

EDP = Energy Delay Product



FPGA outperforms GPU and CPU in energy/frame consumption and EDP

FPGA's Reduction Ratios with repsect to GPU



Pipeline	Energy/frame (mJ/f)	EDP (mJ.s/f2)
Background Subtraction	1.74×	1.32×
Color Segmentation	1.86×	1.41×
Harris Corners Tracking	3.94×	2.65×
Stereo Block Matching	8.83×	107.7×

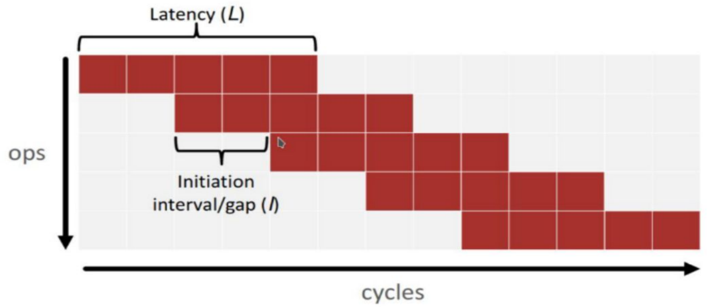
ICESS (2019) doi:10.1109/ICESS.2019.878252

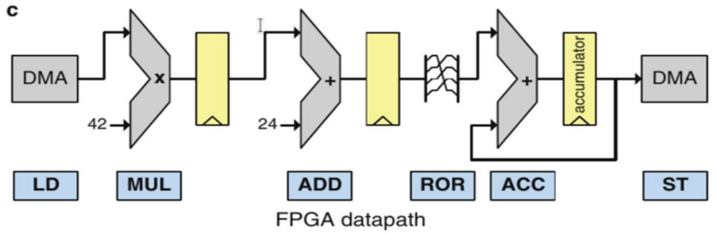
Why using FPGA?

➢ Real-time inference

- Customizable I/O and deterministic latency make them well suited for TDAQ systems.
- In such systems, design challenge is the **processing throughput**:
  - Pipelined designs can potentially produce a new output at each clock cycle.
  - **Initiation interval (II)**: Number of clock cycles before the function can accept new input data. **The lower the II, the higher the throughput**

- High level synthesis tools allows to describe datapaths in FPGA using high level software languages (C/C++, OpenCL, SYCL,...), leveraging *#pragma HLS* directives in order to increase overall performances

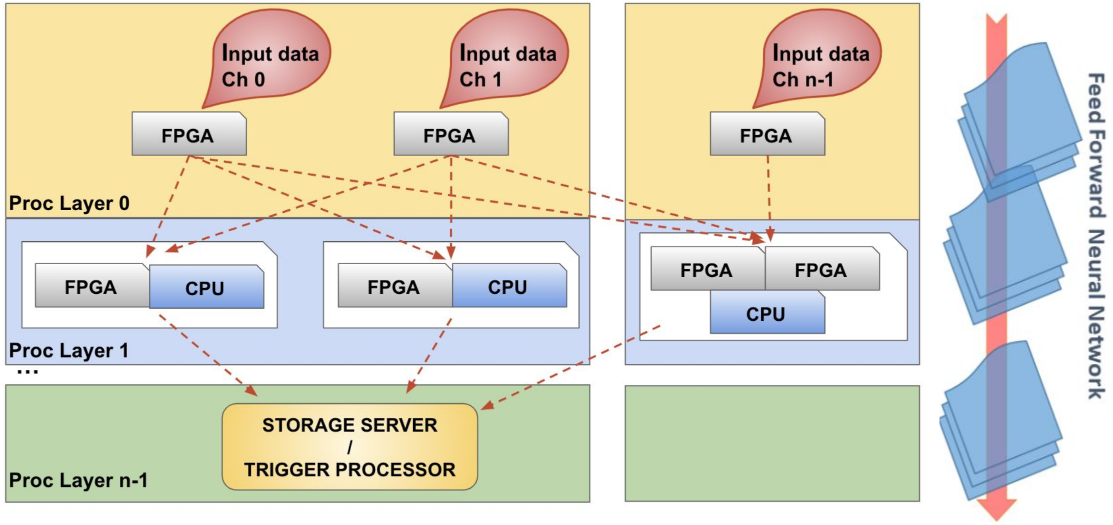




APEIRON for smart TDAQ Systems

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

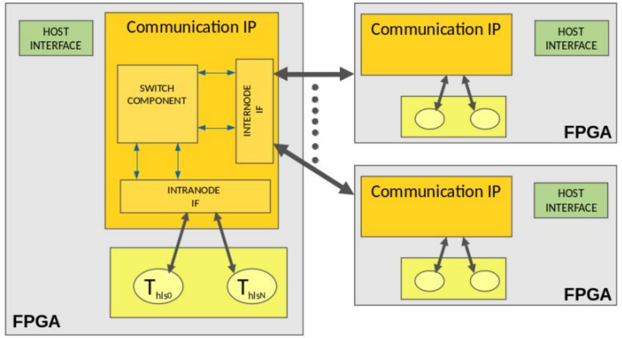
- Input **data streams** from several different channels (data sources, detectors/sub-detectors) recombined through the processing layers using a **low-latency, modular and scalable network infrastructure**

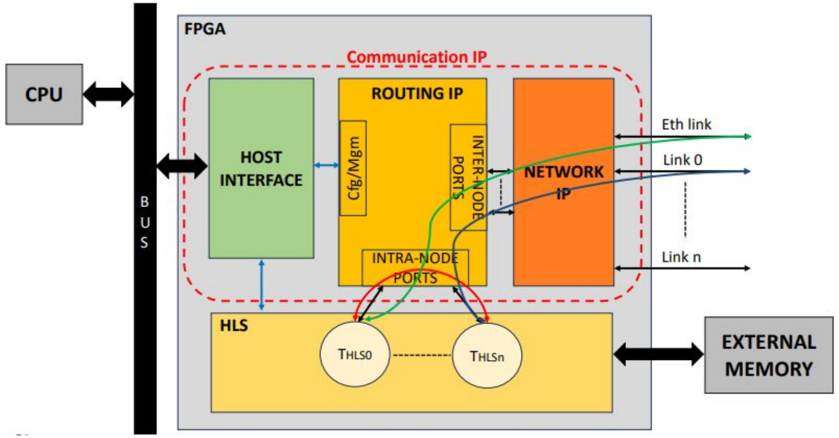


- More resource-demanding NN layers can be implemented in subsequent processing layers.
- Classification produced by the NN in last processing layer (e.g. pid) will be input for the **trigger processor/storage online data reduction stage for triggerless systems**.

APEIRON building blocks:

- INFN Communication IP



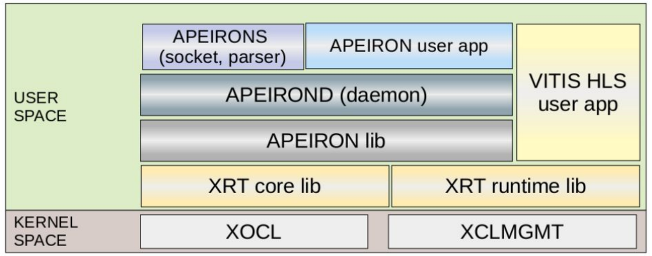


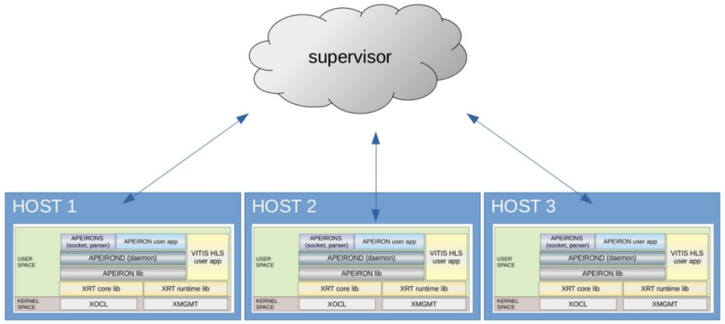
INFN is developing the IPs implementing a direct network that allows **low-latency data transfer** between processing tasks deployed on the same FPGA (**intra-node communication**) and on different FPGAs (inter-node communication)

- **Host Interface IP:** Interface the FPGA logic with the host through the system bus.
- **Routing IP:** Routing of intra-node and inter-node messages between processing tasks on FPGA. •
- **Network IP:** Network channels and Application-dependent I/O
  - **APElink** 20 Gbps → 40 Gbps
  - UDP/IP over 1/10 GbE → 25/40/100 GbE
  - **ETH port** → Xilinx® 10G/25G High Speed Ethernet Subsystem

APEIRON building blocks:

• Software Stack





The APEIRON runtime software stack is built on top of the Xilinx® XRT one adding three layers to:

- add the functionalities required to manage multiple FPGA execution platforms (e.g., **program** the devices, **configure** the IPs, start/stop **execution**, **monitor** the status of IPs, …);
- reduce the impact of changes in XRT API introduced with any new version of Vitis on the APEIRON host-side applications;
- decouple the APEIRON software stack from the specific platform, easing the future porting of the framework to different platforms/vendors.

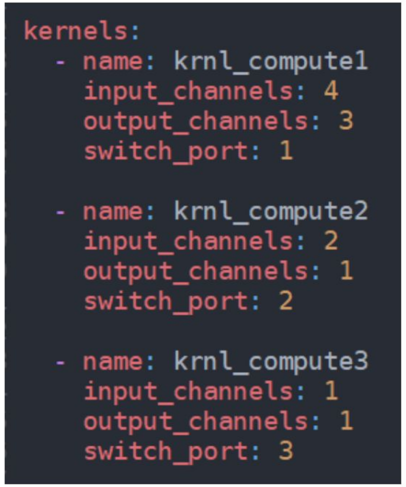
Apeirond is a persistent daemon used to manage multiple access request from user apps to the board.
Using the network socket exposed by apeirond modules, the **supervisor** can write commands and read status of the different instances of the APEIRON framework running in each node, allowing the user to have a complete overview of the multiple FPGA execution platform

APEIRON: FPGA bitstream generation

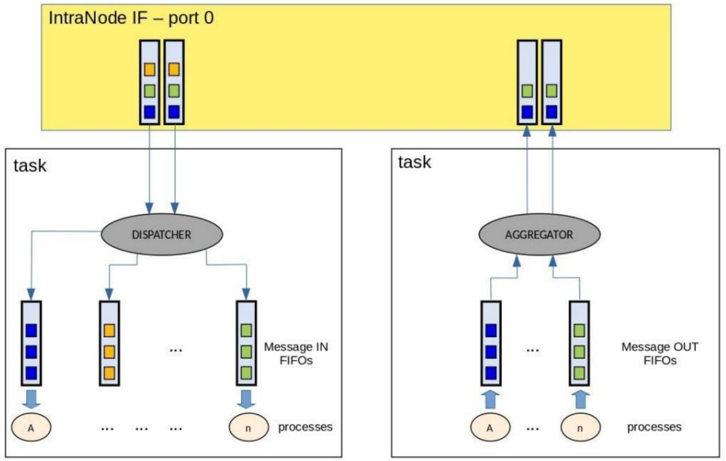
- The **HLS task** must have a generic interface, implementation is free
- A **YAML configuration file** is used to describe the kernels interconnection topology, specifying how many input/output channels they have

```
void example_task(
[list of optional kernel specific
parameters], message_stream_t
message_data_in[N_INPUT_CHANNELS],
message_stream_t
message_data_out[N_OUTPUT_CHANNELS])
```

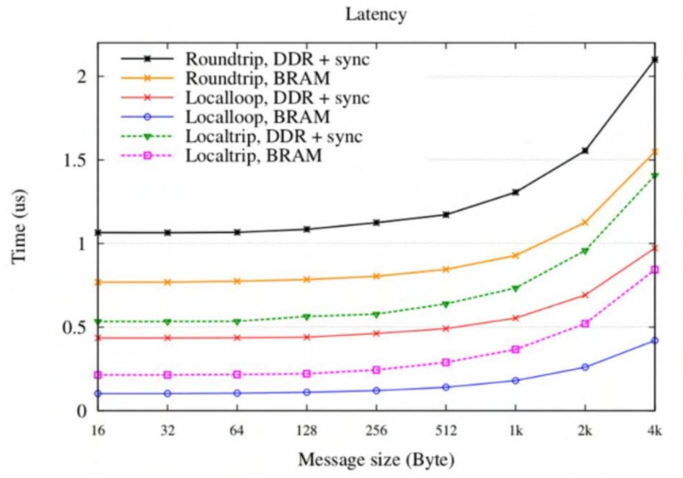
Adaptation toward/from IntraNode ports of the Routing IP is done by the automatically generated **Aggregator** and **Dispatcher** kernel templates.

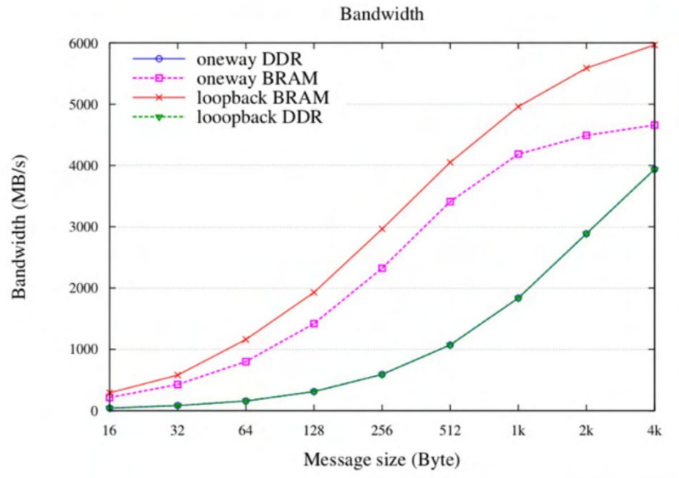


```
kernels:
  - name: krnl_compute1
    input_channels: 4
    output_channels: 3
    switch_port: 1

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

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



APEIRON performance
(Communication IP: 256 bit datapath @200MHz)





Latency	DDR+sync(ns)	BRAM(ns)
Intra-node (localtrip)	533	213
Inter-node (roundtrip)	1065	768

Bandwidth	DDR+sync(MB/s)	BRAM(MB/s)
Intra-node (loopback)	3938	5967
Inter-node (oneway)	3938	4658

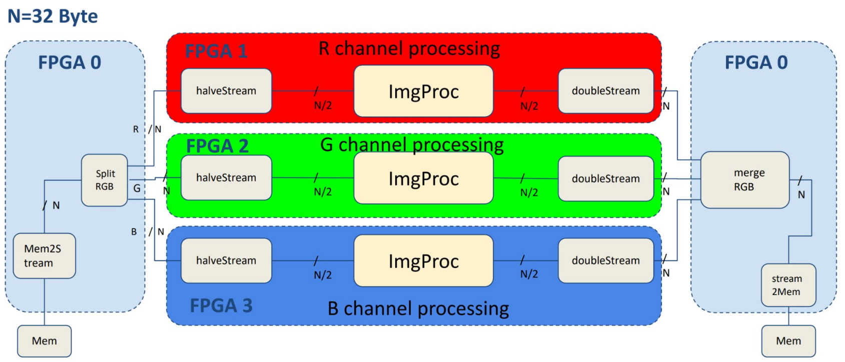
APEIRON applications:
● FIPLib-multiFPGA

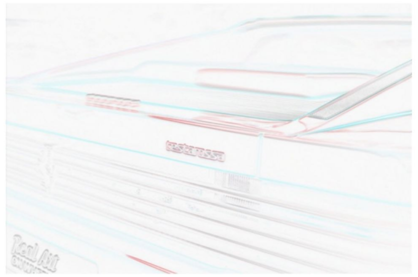
FPGA **I**mage **P**rocessing **Lib**rary ⇒ multi-FPGA implementation via APEIRON

- Developed by ENEA in C++, it employs the **Vitis HLS flow** to construct the library's kernels for the execution of image processing algorithms.

- FIPLib encompasses nearly 70 functionalities, conceived with a **streaming behavior**

- On a multi-FPGA setup, we were able to split the overall image processing by implementing a single RGB kernel on each node
 ⇒ **increased internal datapath to 32B**, avoiding FPGA resource limitation



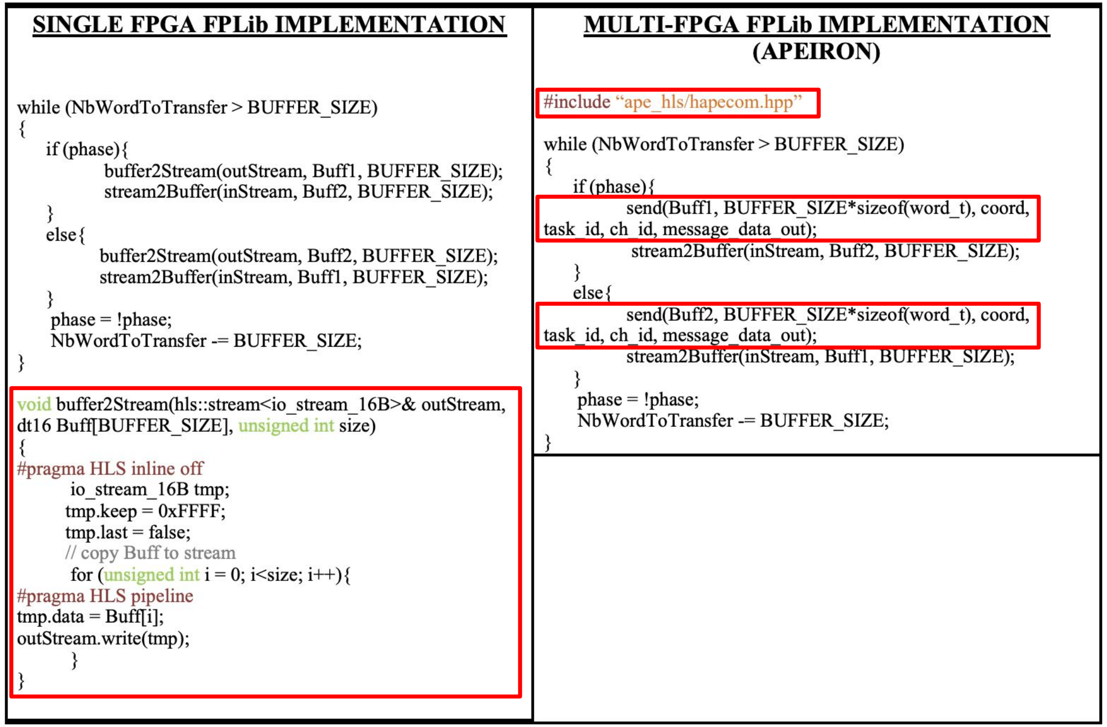


APEIRON applications:
● FIPLib-multiFPGA

textarossa

FPGA **I**mage **P**rocessing **Lib**rary ⇒ multi-FPGA implementation via APEIRON

- Implementing **FIPLib HLS kernels as APEIRON tasks** means changing the interface of each of them to cope with the standard required by the framework to compile the entire project and to generate the bitstream
  ⇒ use of HAPECOM C++ communication API



SINGLE FPGA FPLib IMPLEMENTATION

```
while (NbWordToTransfer > BUFFER_SIZE)
{
    if (phase){
            buffer2Stream(outStream, Buff1, BUFFER_SIZE);
            stream2Buffer(inStream, Buff2, BUFFER_SIZE);
    }
    else{
            buffer2Stream(outStream, Buff2, BUFFER_SIZE);
            stream2Buffer(inStream, Buff1, BUFFER_SIZE);
    }
    phase = !phase;
    NbWordToTransfer -= BUFFER_SIZE;
}
```

```
void buffer2Stream(hls::stream<io_stream_16B>& outStream,
dt16 Buff[BUFFER_SIZE], unsigned int size)
{
#pragma HLS inline off
        io_stream_16B tmp;
        tmp.keep = 0xFFFF;
        tmp.last = false;
        // copy Buff to stream
        for (unsigned int i = 0; i<size; i++){
#pragma HLS pipeline
tmp.data = Buff[i];
outStream.write(tmp);
        }
}
```

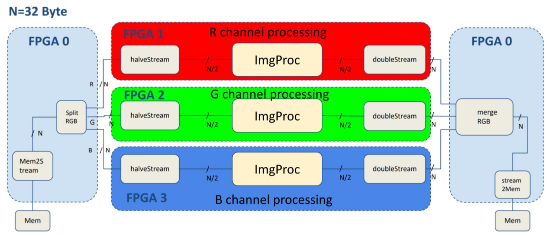
MULTI-FPGA FPLib IMPLEMENTATION (APEIRON)

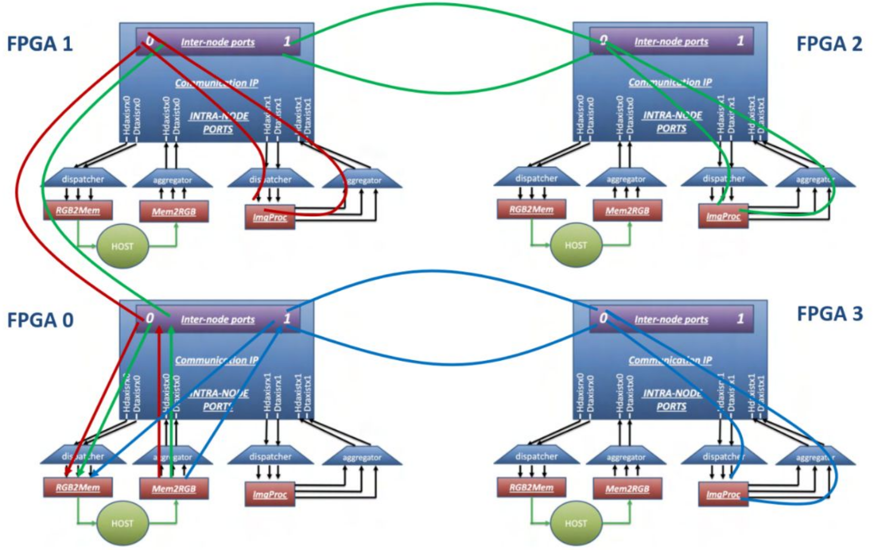
```
#include "ape_hls/hapecom.hpp"

while (NbWordToTransfer > BUFFER_SIZE)
{
    if (phase){
            send(Buff1, BUFFER_SIZE*sizeof(word_t), coord,
task_id, ch_id, message_data_out);
            stream2Buffer(inStream, Buff2, BUFFER_SIZE);
    }
    else{
            send(Buff2, BUFFER_SIZE*sizeof(word_t), coord,
task_id, ch_id, message_data_out);
            stream2Buffer(inStream, Buff1, BUFFER_SIZE);
    }
    phase = !phase;
    NbWordToTransfer -= BUFFER_SIZE;
}
```

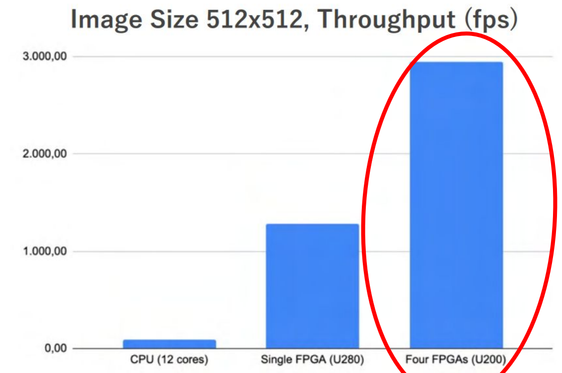
APEIRON applications:

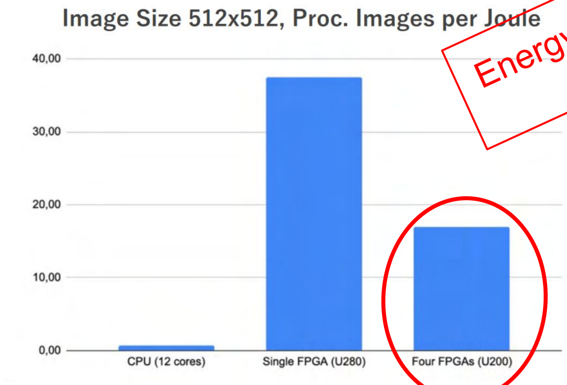
- **FIPLib-multiFPGA**

texta**rossa**









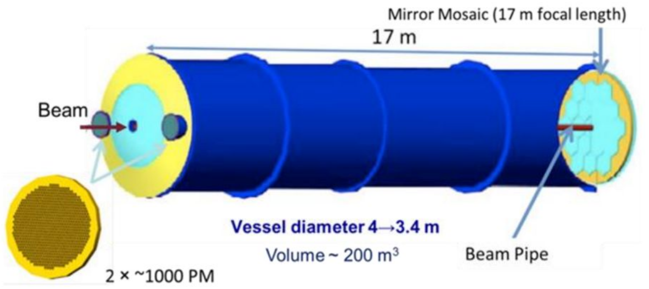
Energy Efficiency - Throughput trade-off

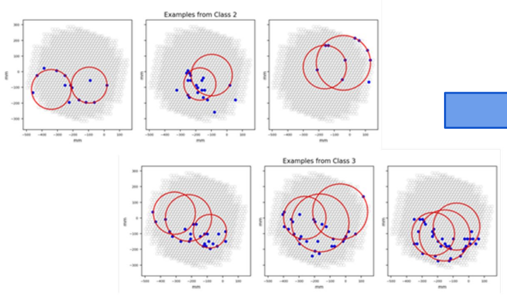
APEIRON applications:
● RAIDER

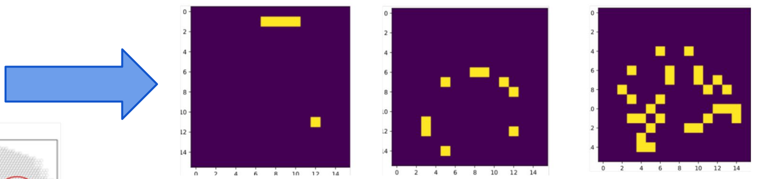
Real-time **AI**-based **D**ata analytics on h**E**te**R**ogeneous distributed systems

- **High throughput online streaming processing** on multi-FPGA ⇒ **number of Cherenkov rings prediction** on the stream of events generated by the RICH detector in the CERN NA62 experiment at a rate of about 10 MHz, using multiple *CNN_kernel* replica.

- **Lightweight CNN model** deployed on Xilinx Alveo U280 FPGA (limited resource usage)
⇒ receives as input compressed representation of the original event in form of B&W 16x16 image
(via *imagifier* kernel)







APEIRON applications:

• RAIDER

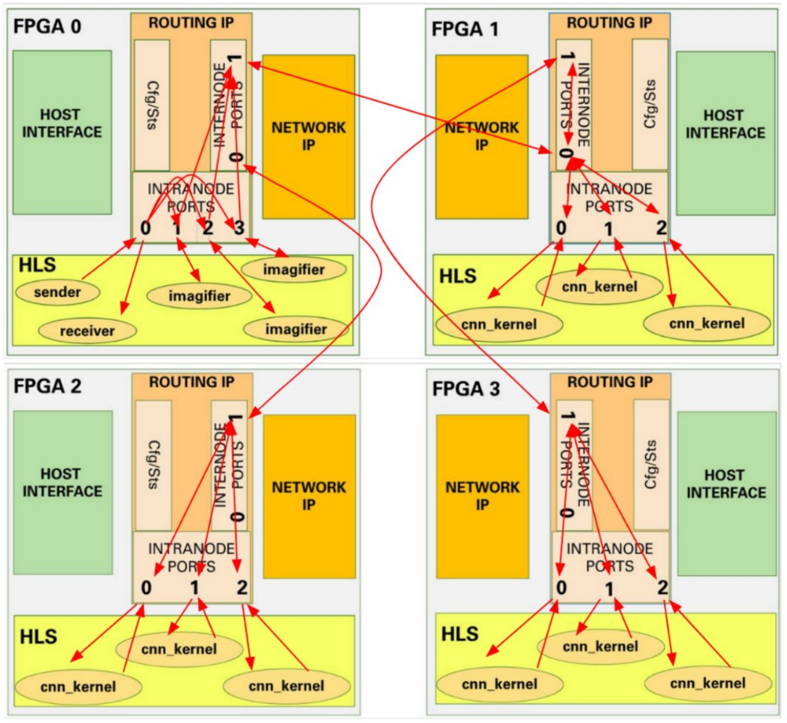
textarossa

KPI	CNN CPU tensorflow	CNN CPU+GPU tensorflow
purity/efficiency (per class)	efficiency: - 0: 93% - 1: 83% - 2: 75% - 3+: 83% purity: - 0: 88% - 1: 90% - 2: 71% - 3+: 78%	efficiency: - 0: 93% - 1: 83% - 2: 75% - 3+: 83% purity: - 0: 88% - 1: 90% - 2: 71% - 3+: 78%
time to solution [s]	158.521	125.963
throughput [events/s]	**189250**	**238165**
energy to solution [J]	11091.919	17497.783 (8724.648 GPU)
energy efficiency [events/J]	**270.467**	**154.305**

KPI	RAIDER @200 MHZ [4 FPGA, 9CNNs]
time to solution [s]	0.554
throughput [events/s]	**4873646.209**
energy to solution [J]	165.277 (101.055 FPGA)
energy efficiency [events/J]	**16336.183 (26718.126 FPGA)**

x20

x100



Conclusions

- The APEIRON framework enables the **development and deployment of Vitis HLS dataflow applications distributed on multiple-FPGA systems,** leading to increased performance in terms of **throughput** and **energy efficiency**

- The co-design of its software stack and of the Communication IP allowed to reach **very low and deterministic latency and a high fraction of the channel's raw bandwidth** for communications between FPGAs, addressing fundamental bottlenecks for real-time distributed dataflow applications.

- We control the workflow for the implementation of **real-time/high throughput classifiers on FPGA** using limited resources,
  This hints for applying the methodology also to:
  - less capable (i.e. front-end) FPGAs
  - complex design making use of a large fraction of FPGA resource
    ⇒ multi-node setup/user-defined topology

Conclusions

- Eager to find new applications for APEIRON framework, feel free to contact us!

Thanks for your attention!

Contacts:

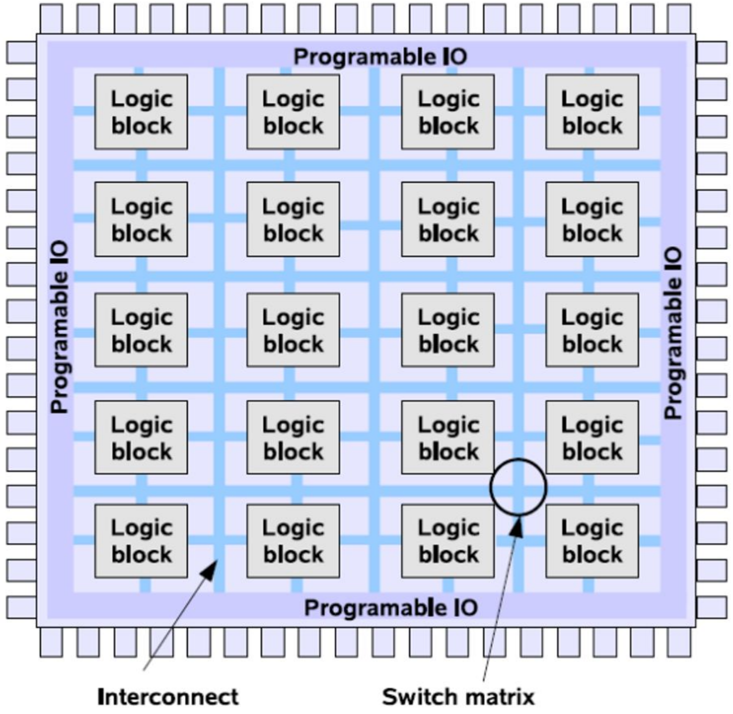
- cristian.rossi@roma1.infn.it
- alessandro.lonardo@roma1.infn.it

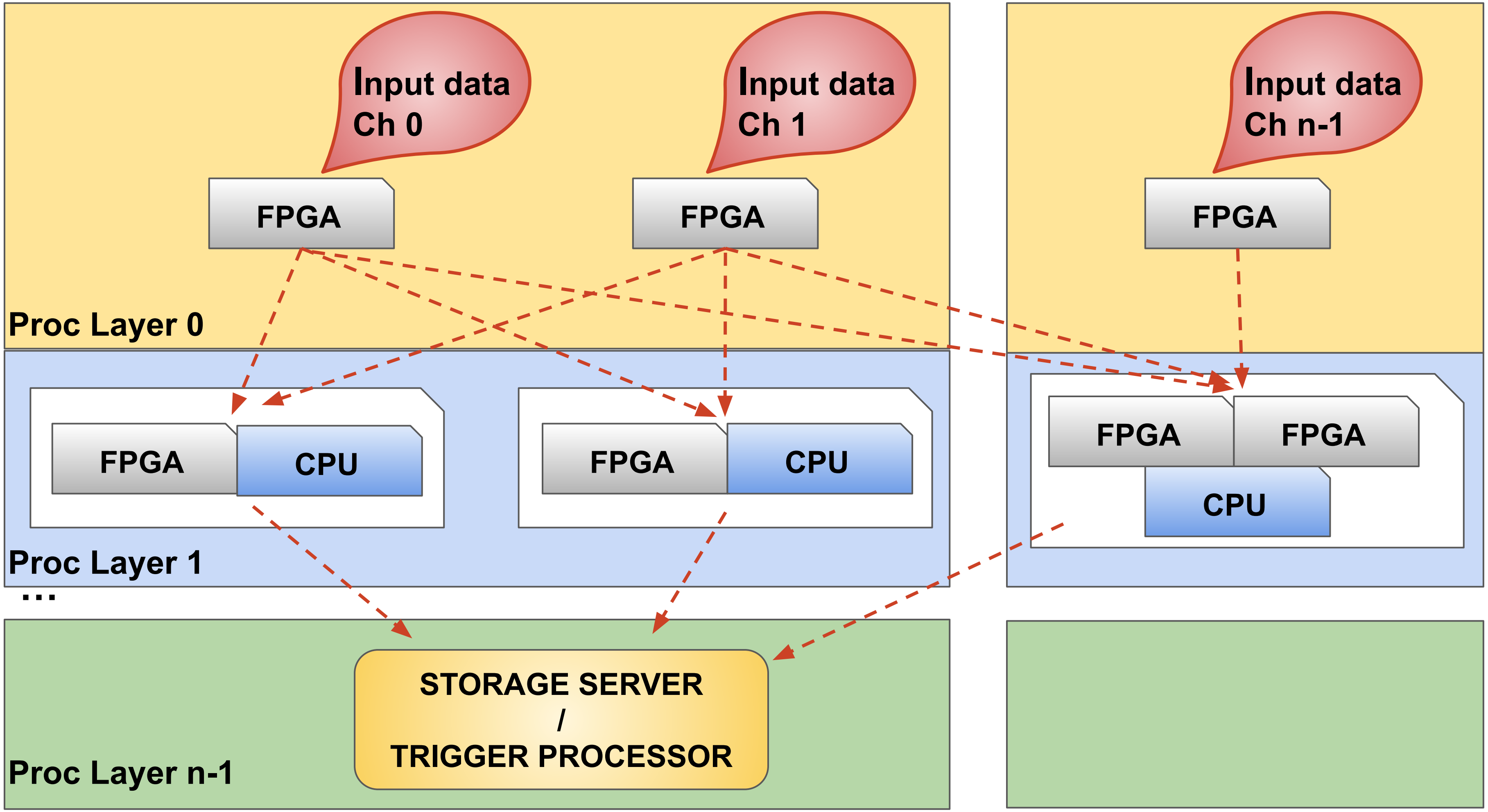
BACKUP SLIDES

FPGA overview

The basic structure of an FPGA is composed of the following elements:

- **Look-up table (LUT):** This element performs logic operations
- **Flip-Flop (FF):** This register element stores the result of the LUT
- **Wires:** These elements connect elements to one another, both logic and clock
- **Input/Output (I/O) pads:** These physically available ports get signals in and out of the FPGA




Input data Ch 0
Input data Ch 1
Input data Ch n-1
FPGA
FPGA
FPGA
Proc Layer 0
FPGA
CPU
FPGA
CPU
FPGA
FPGA
CPU
Proc Layer 1
...
STORAGE SERVER / TRIGGER PROCESSOR
Proc Layer n-1