EPJ Web of Conferences **319**, 12013 (2025) https://doi.org/10.1051/epjconf/202531912013
RICAP-24

APEIRON: a framework for the development of smart TDAQ systems

Roberto Ammendola[2], *Andrea* Biagioni[1], *Carlotta* Chiarini[13], *Andrea* Ciardiello[4], *Paolo* Cretaro[1], *Ottorino* Frezza[1], *Francesca* Lo Cicero[1], *Alessandro* Lonardo[1], *Michele* Martinelli[1], *Pier Stanislao* Paolucci[1], *Pierpaolo* Perticaroli[1], *Luca* Pontisso[1], *Francesco* Simula[1], *Cristian* Rossi[13,*], and *Piero* Vicini[1]

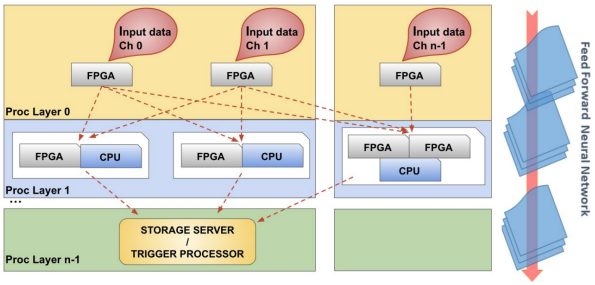
[1]Istituto Nazionale di Fisica Nucleare (INFN), Sezione di Roma, Rome, Italy
[2]Istituto Nazionale di Fisica Nucleare (INFN), Sezione di Roma Tor Vergata, Rome, Italy
[3]Università di Roma "La Sapienza", Rome, Italy
[4]Istituto Superiore di Sanità

Abstract. APEIRON is a framework encompassing the general architecture of a distributed heterogeneous processing platform and the corresponding software stack, from the low level device drivers up to the high level programming model. Developers can define scalable applications that can be deployed on a multi-FPGA system coding at high level: the APEIRON communication IPs allow low-latency communication between processing tasks deployed on FPGAs, even if hosted on different computing nodes. Thanks to the use of High Level Synthesis tools, tasks are described in high level language (C/C++) while communication is expressed through a lightweight API. The aim of the APEIRON project was to develop a flexible framework that could be adopted in the design and implementation of both "traditional" low level trigger systems and of data reduction stages in trigger-less or streaming readout experimental setups.

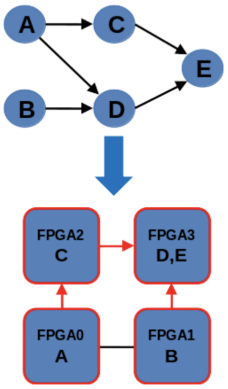
1 Introduction

The APEIRON framework [1] aims at offering hardware and software support to run distributed real-time dataflow applications on a network of interconnected FPGAs.
Its architecture has been developed taking as reference a custom distributed processing platform composed by m input data streams recombined through n processing layers using a low-latency, modular and scalable network infrastructure. This mimics the dataflow of a typical Trigger and Data AcQuisition (TDAQ) system, where data keep streaming from the readout to a trigger processor or a storage system through several layers of processing stages (Fig. 1a).
Developers can use a dataflow programming model inspired by Kahn processing networks [2] for developing and deploying scalable application on a multi-FPGAs system. An embedded configuration tool allows the straightforward mapping of the application computational dataflow graph onto the underlying network of FPGAs (Fig. 1b), linking the processing tasks and the interconnection logic to generate the FPGA bistreams.

*e-mail: cristian.rossi@roma1.infn.it



(a) APEIRON architecture model: input data streams from several different channels (data sources/sub-detectors) processed through several NN computations, implemented as subsequent network-interconneced computing nodes (typically CPU orchestrated via APEIRON Software Stack). Classification produced by the NN may be used as input for trigger processor/storage online data reduction stage for triggerless systems.



(b) Dataflow graph (composed by different computation steps, indicated by different letters) mapped on 4 interconneted FPGAs system

Processing tasks can be implemented in C/C++ via High Level Synthesis tools (e.g. Xilinx Vitis) and deployed on the different FPGA interconnected boards. Communication between tasks is implemented by the APEIRON communication IP and is expressed through a lightweight API (called HAPECOM) based on non-blocking *send()* and blocking *receive()* primitives.

2 APEIRON Building blocks

2.1 INFN Communication IP

Based on the HPC direct network designs previously developed by our group, like APEnet [3] and ExaNet [4], the Communication IP represents the main enabling component of the APEIRON framework. This IP has been developed to implement a direct network enabling low-latency data transfer between processing tasks deployed on the same FPGA (intra-node communication) or on different FPGAs (inter-node communication).
Figure 2 shows the Communication IP hardware block structure, which contains:

- **Host Interface IP** interfacing the FPGA logic with the host through the system bus
- **Routing IP** defining the switching technique and routing algorithm. It consists of the Switch component, the Configuration/Status Registers and the InterNode/IntraNode interfaces. In this scheme, the Router configures the proper path across the switch while the Arbiter solves contentions between packets requiring the same port.
- **Network IP** composed by Application-dependent I/O and Network channels using APElink, the INFN proprietary high-throughput, low-latency data transmission protocol for direct network interconnect based on word-stuffing technique, with a bandwidth up to 40 Gbps.

The transmission is packet-based: the Communication IP sends, receives and routes packets with a header, a variable size payload and a footer.

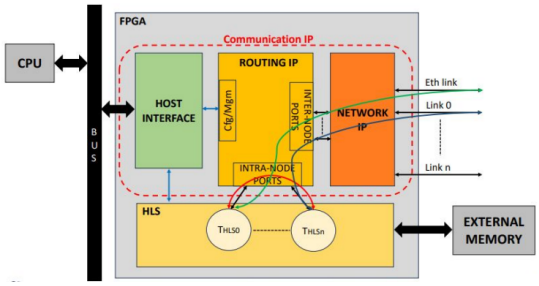


Figure 2: Communication IP hardware block structure with HLS kernels performing intra-node (red line) and inter-node (green line – Ethernet, blue line– APElink) communications.

2.2 Runtime Software Stack

Within the APEIRON framework, we designed a runtime software stack, shown in Fig. 3, based on the Xilinx Runtime (XRT) architecture and implemented as a combination of user-space and kernel driver components [5]. *Apeirond*, a persistent daemon used to manage multiple access requests from user host applications to the board. It uses functions exposed by the APEIRON library to operate on the device. *Apeirond* module accepts client connections over a network socket (*apeirons* module) and oversees creating the socket with the client and handling the incoming commands.

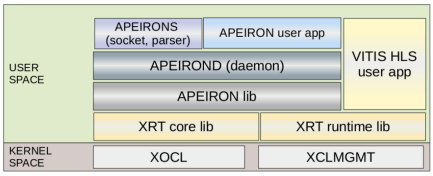


Figure 3: APEIRON Software Stack scheme

2.2.1 Workflow for FPGA bitstream generation

Via a YAML configuration file, users can describe the attributes of each HLS kernel. Starting from this, the APEIRON framework links the Communication IP and the HLS kernels that are connected to it and generates the bitstream for the overall design.

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

2.2.2 HAPECOM Communication API

The communication between kernels is expressed through HAPECOM: a lightweight C++ API based on non-blocking *send()* and blocking *receive()* operations. This API allows

the HLS developer to perform communication without knowing the details of the underlying packet protocol. The HAPECOM Communication API is represented by the following pseudo-code:

```
size_t send (msg, size, dest_node, task_id, ch_id);
size_t receive (ch_id);
```

where: `msg` is a user-defined data buffer to be streamed through the network; `size` is the bitsize of the buffer; `dest_node` is the n-Dim coordinate of the destination node (FPGA) in an n-Dim torus network; `task_id` is the local-to-node receiving task (kernel) identifier; `ch_id` is the local-to-task receiving FIFO (channel) identifier.
Two APEIRON IPs manage the adaptation toward/from IntraNode ports of the Routing IP: they are *Aggregator* and *Dispatcher* HLS kernels, shown in Fig. 4.

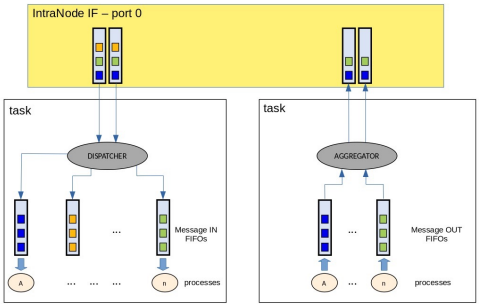


Figure 4: *Aggregator* and *Dispatcher* kernels mediating interface between Intranode Port 0 and the corresponding HLS Task.

3 Acknowledgments

This work has received funding from the European High-Performance Computing Joint Undertaking under grant agreement No. 956831 (TEXTAROSSA). Pierpaolo Perticaroli is a PhD student enrolled in the National PhD program in Artificial Intelligence, XXXIX cycle, course on Health and life sciences, organized by Università Campus Bio-Medico di Roma.

References

[1] R. Ammendola, *et al.*, Apeiron: A framework for high level programming of dataflow applications on multi-fpga systems, EPJ Web of Conf. **295**, 11002 (2024). 10.1051/epjconf/202429511002

[2] K. Gilles, The semantics of a simple language for parallel programming, Information processing **74**, 471 (1974).

[3] R. Ammendola, *et al.*, Apenet+ 34 gbps data transmission system and custom transmission logic, Journal of Instrumentation **8**, C12022 (2013). 10.1088/1748-0221/8/12/C12022

[4] R. Ammendola, *et al.*, The Next Generation of Exascale-Class Systems: The ExaNeSt Project, in *Proceedings - 20th Euromicro Conference on Digital System Design, DSD 2017* (IEEE, United States, 2017), pp. 510–515

[5] https://xilinx.github.io/xrt/master/html/index.html