EVEREST

DESIGN ENVIRONMENT
FOR EXTREME-SCALE BIG DATA ANALYTICS
ON HETEROGENEOUS PLATFORMS

European Big Data Value Forum 2022 – Nov 23rd, 2022

EVEREST SDK:
Environment for High-Performance, Distributed, Reconfigurable and Heterogeneous Platforms

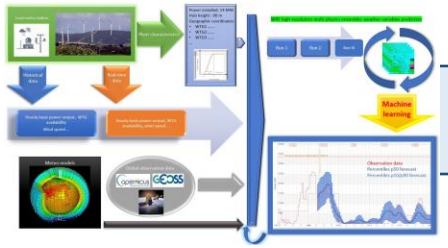
CHRISTIAN PILATO

EVEREST Scientific Coordinator

christian.pilato@polimi.it

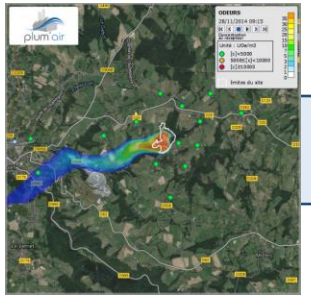
http://www.everest-h2020.eu

EVEREST Use Cases



Renewable energy production prediction

★ Improve **quality of the predictions**

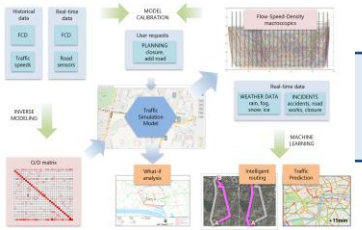


Air-quality monitoring of industrial sites

★ Improve the **response time of predictions**

Weather prediction modelling (WRF)

★ Accelerate kernels to execute more tests



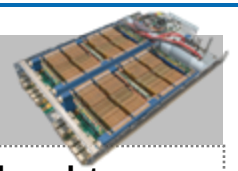
Traffic modeling for intelligent transportation

★ Improve the **overall performance of traffic simulation**

Accelerated computationally-intensive kernels + **Machine-learning** kernels

EVEREST Target System

cloudFPGA



- **Disaggregated FPGAs** directly attached to the network (64 FPGA instances)
- **Low latency** and **high bandwidth** system
- **cFDK framework** for system generation
- Separation between **Shell** and **Role** modules

FPGA-Accelerated HPC Cluster

- Cluster of **PCIe-attached FPGAs** (Alveo) with HBM architecture (up to 460 GB/s per board)
- **Xilinx Vitis framework** for HLS and system integration
- Support for the integration of **custom HDL**

CPU Reference System

- CPU-based infrastructure to **execute end-to-end workflows**, **manage storage**, and **data transfers**
- Extended to support the **offloading of tasks** to **FPGA servers**

LEXIS

Exploit **spatial parallelism**

High **memory bandwidth**

Different nodes to better **match applications**

Data-intensive (memory-bound) applications

Seamless support for multiple nodes

Limited **FPGA resources** (esp. memories)

The EVEREST Project

Big data applications with heterogeneous data sources

FPGA-based architectures to accelerate selected kernels

- App designers are not FPGA experts
- Hardware accelerators require many optimizations
- Target nodes can have different characteristics

How to optimize big data applications on FPGA-based architectures?

Compilation

Unified hardware generation flow (high-level synthesis)

MLIR

Generation of **variants**

Increase designers' productivity

Increase quality of accelerators

Improve applications' results

Runtime

Dynamic adaptation to variants

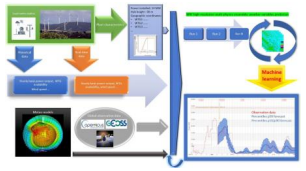
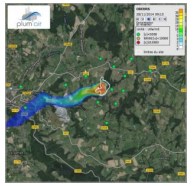
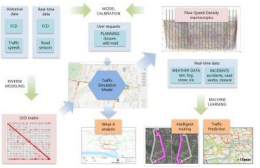
Virtualization of resources

Multi-node support

EVEREST Approach

Big data applications with heterogeneous data sources

Three use cases

What are the relevant requirements for data, languages and applications?

How to design data-driven policies for computation, communication, and storage?

How to create FPGA accelerators and associated binaries?

How to manage the system at runtime?

How to evaluate the results?

How to disseminate and exploit the results?

EVEREST SDK

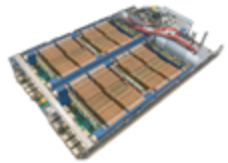
Open-source framework to support the optimization of selected workflow tasks

FPGA-based architectures to accelerate selected kernels

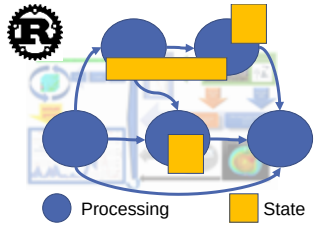
CPU-based infrastructure

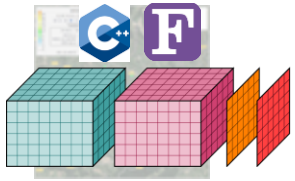
+

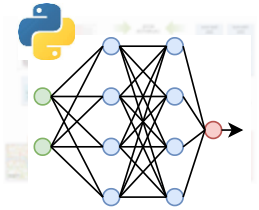
Two FPGA-based clusters


EVEREST System Development Kit (SDK)







Different **input flows**
with different **input languages**

Support for **multiple target boards**

Collection of interoperable and open-source tools to accelerate applications by matching the target system, the application workflow, and the data characteristics

Compilation

- **Convergence** of the input flows into a unified hardware generation flow thanks to **MLIR**
- Use of **high-level synthesis** and **custom memory architectures** to integrate **data management policies**
- Automatic generation of **hardware/software variants** based on the possible targets

Runtime

- Automatic **allocation** of target nodes and virtual machines
- **Virtualization extensions** to expose hardware resources
- **Autotuning framework** to dynamically select the variant that best matches with the application and the underlying hardware

tvm
MLIR
DandA
XILINX VITIS
cFDK
HEAppE
HyperLoom
(and more...)

Main EVEREST Challenges

Challenge 1: Input languages and frameworks – Gap between application designers and hardware/system designers

- Application designers are usually not FPGA experts and may use high-level framework that are not supported by current HLS tools – how to talk with them?

Challenge 2: Optimization of accelerators – Gap between compiler experts and hardware designers

- Traditional compiler optimizations can create inefficient hardware solutions – can we create hardware-oriented and data-driven compilation flows?

Challenge 3: System-level optimization – Gap between hardware designers and system designers

- Accelerator design must consider also the implications for data transfers and FPGA synthesis – how to co-optimize the kernel and the system?

The Case of Computational Fluid Dynamics

Numerical simulations are becoming more and more popular for many applications

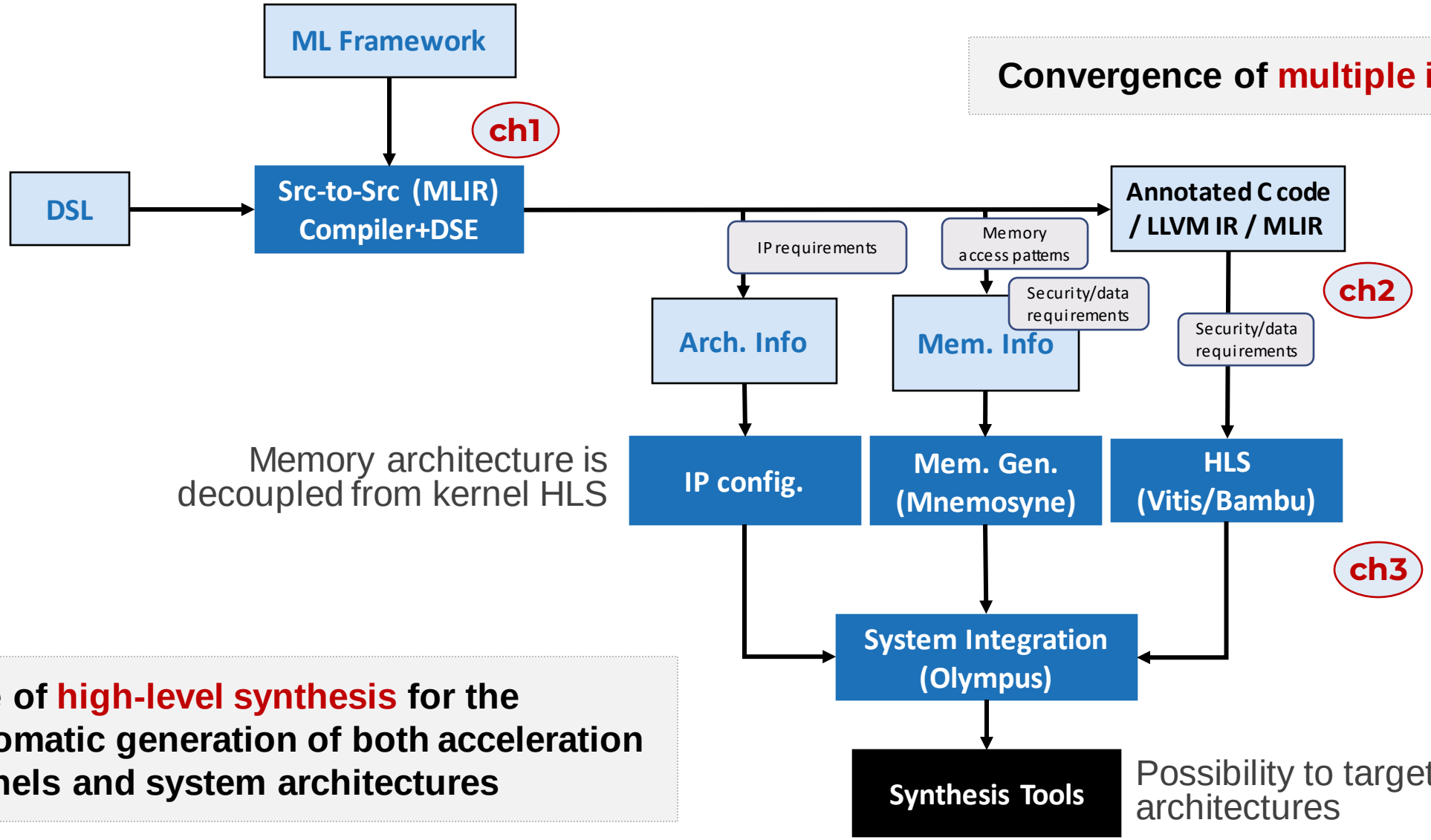
- **Computational Fluid Dynamics** requires to solve partial differential equations
- **Inverse Helmholtz operator** ("Helmholtz" for the friends) is parametric with respect to polynomial degree *p*

```
var input S   : [11 11]
var input D   : [11 11 11]
var input u   : [11 11 11]
var output v  : [11 11 11]
var t         : [11 11 11]
var r         : [11 11 11]
t = S # S # S # u . [[1 6] [3 7] [5 8]]
r = D * t
v = S # S # S # t . [[0 6] [2 7] [4 8]]
```

Final result is obtained by **"small" contributions** on **independent data**

- CFD kernel is composed of **three high-level tensor operators** (two contractions and one Hadamard product) repeated millions of times – good for spatial parallelism
- Each operator requires $p^2 + 2 \cdot p^3$ *(double) elements* as input and produces p^3 *(double) elements* – 21.74 KB + 10.40 KB per element when p = 11
- *Six tensors (p^3 elements)* to store intermediate results – additional 62.39 KB

MLIR-based Compilation Flow



Convergence of **multiple input flows**

Memory architecture is decoupled from kernel HLS

Possibility to target different architectures

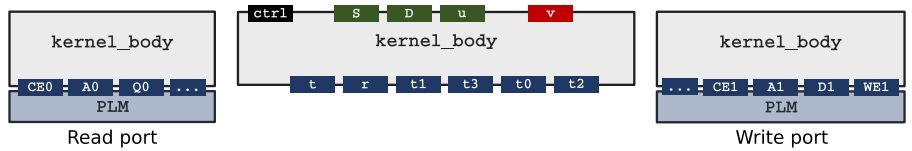
Use of high-level synthesis for the automatic generation of both acceleration kernels and system architectures

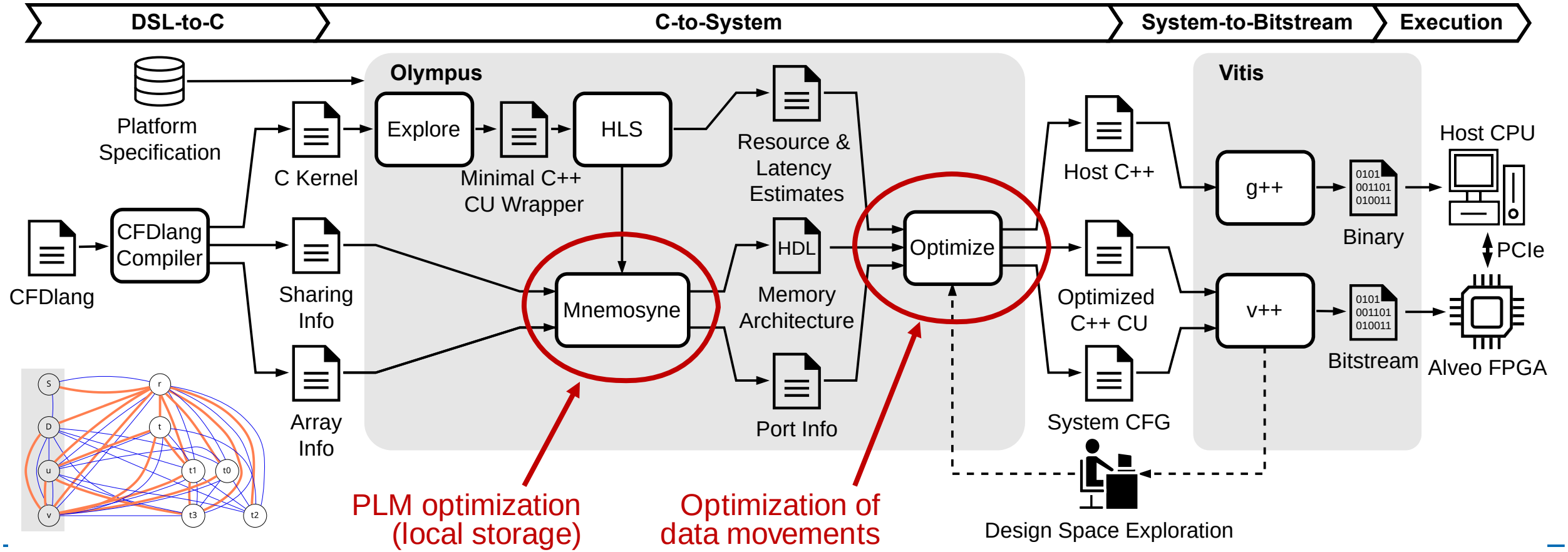
EVEREST

From DSL to Bitstream – Focus on Memory

```
var input S    : [11 11]
var input D    : [11 11 11]
var input u    : [11 11 11]
var output v   : [11 11 11]
var t          : [11 11 11]
var r          : [11 11 11]
t = S # S # S # u . [[1 6] [3 7] [5 8]]
r = D * t
v = S # S # S # t . [[0 6] [2 7] [4 8]]
```

```
void kernel_body(double S[11][11], double D[11][11][11], double u[11][11][11],
                 double v[11][11][11],
                 double t[11][11][11], double r[11][11][11], double t1[11][11][11],
                 double t3[11][11][11], double t0[11][11][11], double t2[11][11][11])
```



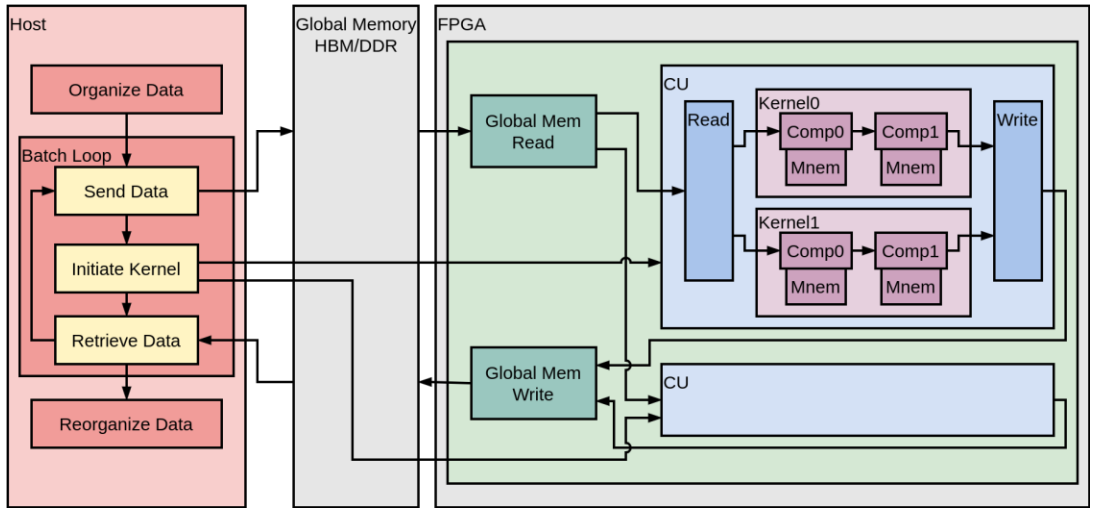


EVEREST

Automated System Generation Flow

EVEREST SDK (statically) determines and (dynamically) manages the **system-level architectures** based on:

- **Algorithm parallelism**
- Characteristics of the **target platform(s)**
- Interfaces of the modules (**HLS tools**)



Automation for the generation of:

- **Synthesizable C++ code** that includes:
  - **Accelerators** and **PLM** generated with HLS
  - **Communication modules** to match interfaces (for *either cloudFPGA SHELL or HBM channels*)
- **System configuration file** to create the overall architecture
  - Support for multiple computing units executing in parallel
- **Host code** for hardware/software integration and interaction
- Support for **virtualization**, **multi-node runtime**, and **autotuning**

Conclusions – Bridging the gaps is possible!

Data management optimizations are becoming the key for the creation of **efficient FPGA architectures** (… more than pure kernel optimizations)

- Additional support for **data protection** and **anomaly detection**

HLS is now used not only to create accelerator kernels but also to generate the **system-level architecture**

- **Portable solutions** across multiple target platforms

Novel **HBM architectures** offer high bandwidth (that's why they are called *high-bandwidth memory* architectures… ☺) but their design is complex:

- Necessary to match **application requirements** and **technology characteristics**

The **EVEREST flow** aims to hide the "low-level" details and bridge the gaps among all designers

Thanks!

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957269