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

EVEREST+DAPHNE Workshop @ HiPEAC 2024 – January 19, 2024

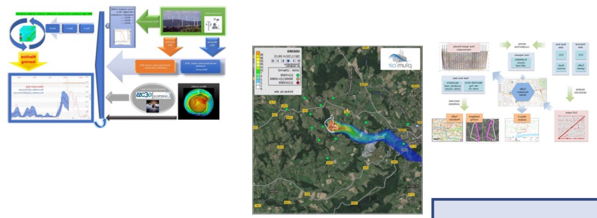
A System Development Kit for Big Data Applications on FPGA-based Clusters: The EVEREST Approach

CHRISTIAN PILATO

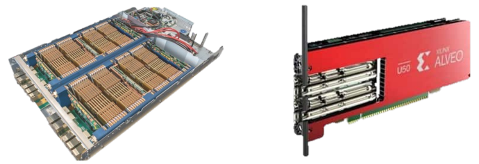
Associate Professor (Politecnico di Milano) & EVEREST Scientific Coordinator

christian.pilato@polimi.it

The EVEREST Project



Big data applicati
heterogeneous d



- App designers are not FPGA experts
- Hardware accelerators require many optimizations

an have different

H2020 Project – 10 partners, 6 countries

Project Coordinator: Christoph Hagleitner, IBM Research Europe

Scientific Coordinator: Christian Pilato, Politecnico di Milano

Budget: ~5M€

Start date: October 1, 2020 (36+6 months)

Compilation

Unified hardware gene
(high-level synth

MLIR

Generation of **variants**

Increase quality of accelerators

Improve applications' results

Runtime

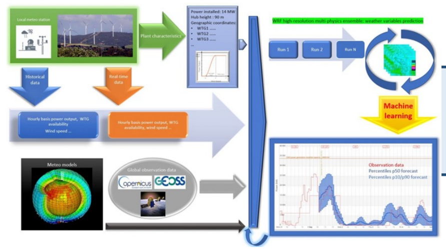
Adaptation to variants

Virtualization of resources

Multi-node support

EVEREST

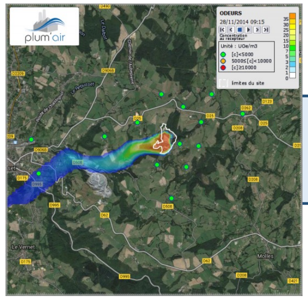
EVEREST Use Cases



Renewable energy production prediction

★ Improve **quality of the predictions**

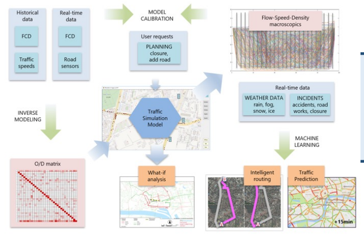
Weather prediction modelling (WRF)



Air-quality monitoring of industrial sites

★ Improve the **response time of predictions**

★ Accelerate kernels to execute more tests



Traffic modeling for intelligent transportation

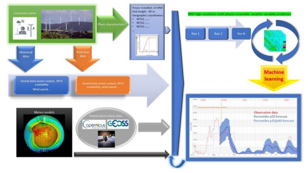
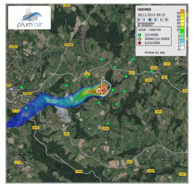
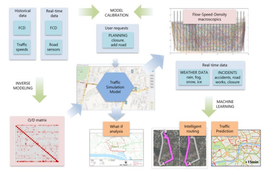
★ Improve the **overall performance of traffic simulation**

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

EVEREST 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

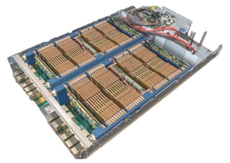
Open-source framework to support the optimization of selected workflow tasks

FPGA-based architectures to accelerate selected kernels

CPU-based infrastructure

+

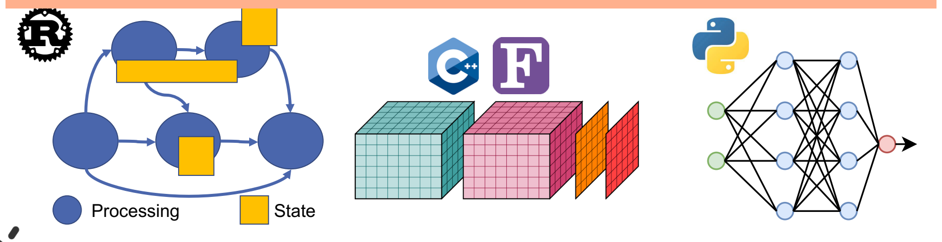
Two FPGA-based clusters

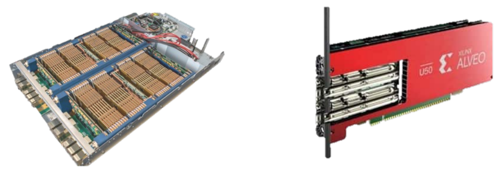


EVEREST System Development Kit (SDK)

Different **input flows** starting from different **input languages**

Support for **multiple target boards** (and possibly other targets)





System Development Kit (SDK): Collection of languages, tools, and methods to simplify the deployment and optimize the execution of large use cases in heterogeneous systems

- Simplified **workflow** definition and automatic deployment, and management
- Compilation framework based on **MLIR** to unify the input languages
  - **High-level synthesis** and **hardware generation flow** to automatically create optimized architectures
- Hardware and software **variants** to match architecture and application features
  - **Virtualized environment and autotuning** for runtime adaptation
- Built on top of state-of-the-art frameworks and commercial toolchains for FPGA synthesis

tvm

MLIR

PandA

XILINX VITIS

cFDK

HEAppE

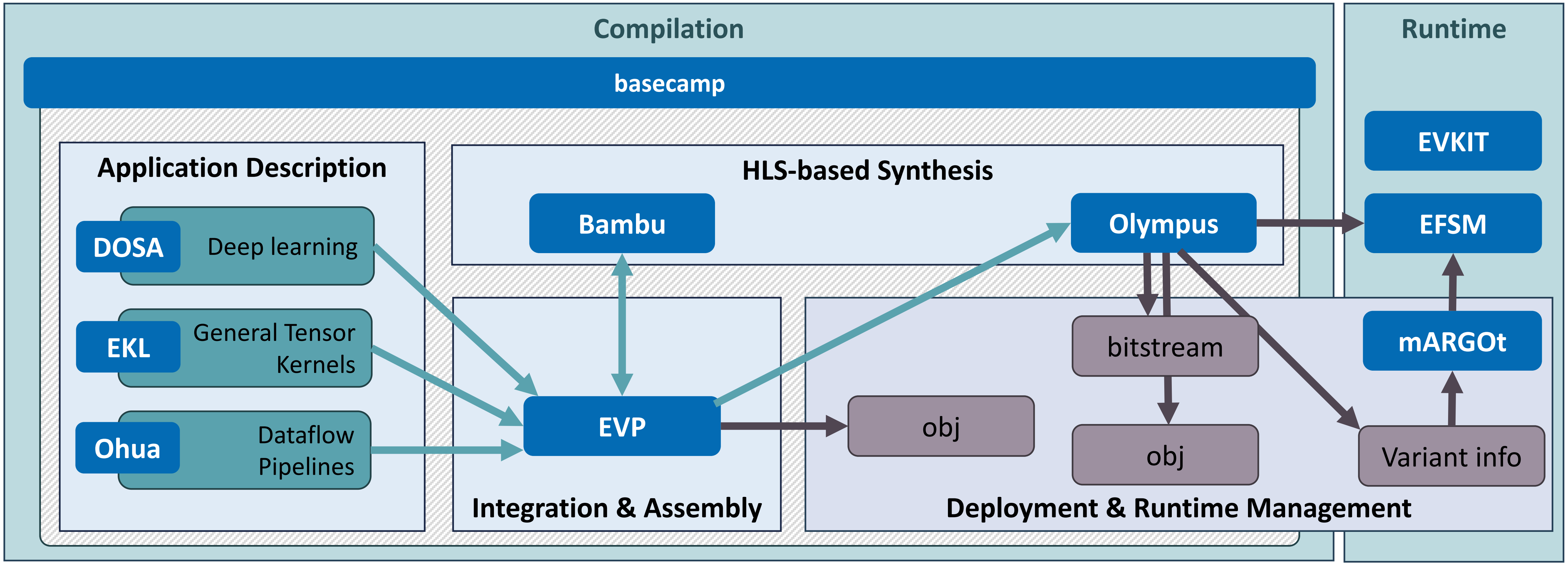
HyperLoom

(and more...)

EVEREST

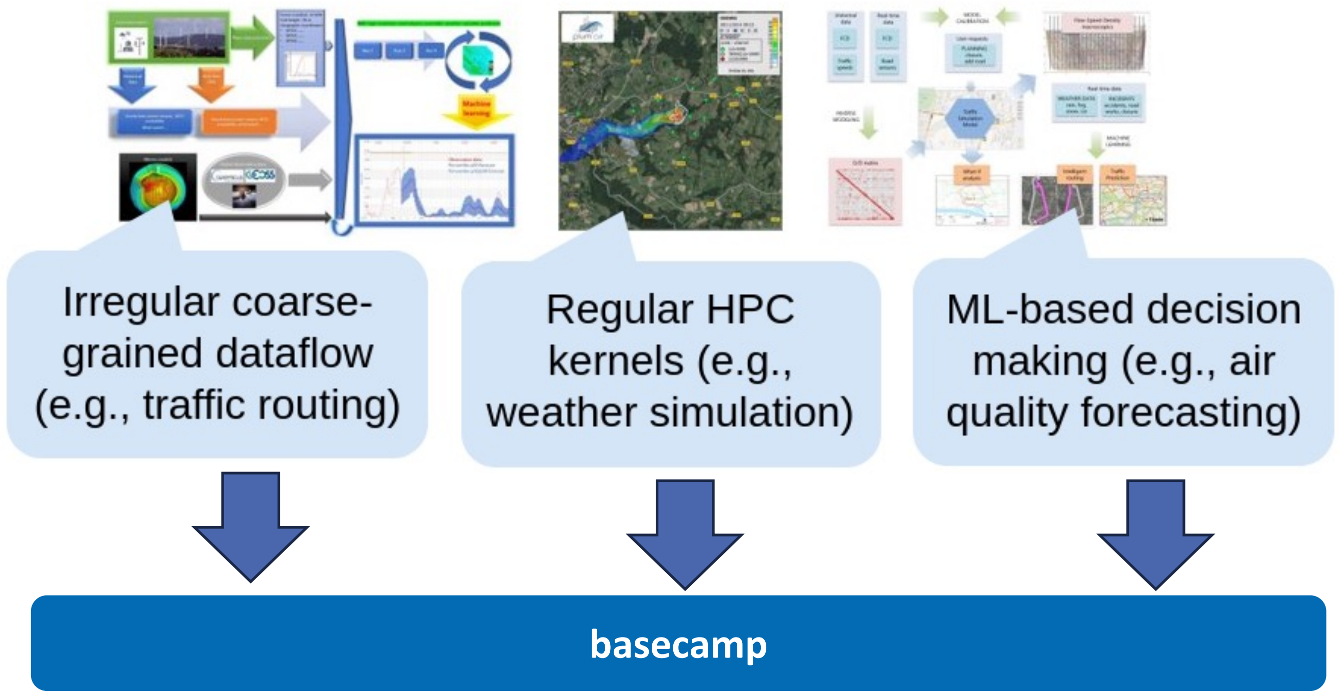
EVEREST SDK Overview

Collection of tools with common interface to interact with each other, exchange information, and produce output files



basecamp – The start of all EVEREST endeavors

- Single Entry for the EVEREST SDK → **basecamp**
  - Wraps components together
  - Modular dependencies for simplified installation
- Single Exit → EVEREST Runtime


Irregular coarse-grained dataflow (e.g., traffic routing)
Regular HPC kernels (e.g., weather simulation)
ML-based decision making (e.g., air quality forecasting)
basecamp

EVEREST Compilation Framework

The **EVEREST Compilation Framework** is based on MLIR and leverages HLS to generate FPGA accelerators:

- Supports different input flows with domain-specific languages for tensor expressions (**EKL**), dataflow descriptions (**Ohua**), and ML-based applications (**DOSA**) thanks to **MLIR abstractions**
- **Unified MLIR-based compiler (EVP)** coordinates kernel- and system-level optimizations
- Uses academic (**Bambu**) and commercial (**Vitis HLS**) tools for **high-level synthesis** to obtain hardware descriptions
- **System-level generation** (**Olympus**) aims at optimizing the FPGA architecture with an efficient implementation of multiple parallel units

Multi-Level Intermediate Representation (MLIR)

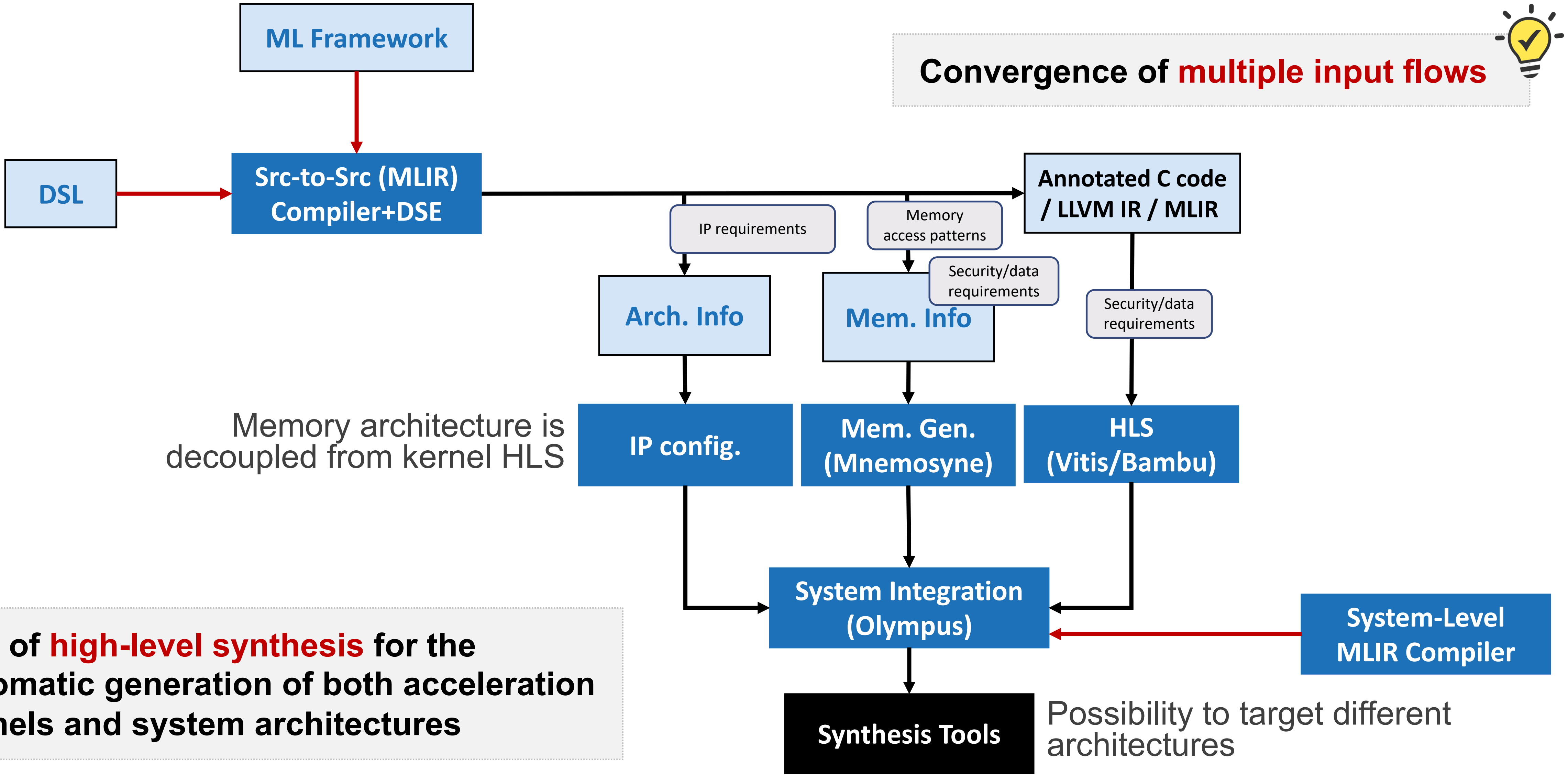
Novel compiler infrastructure centered on reuse and extensibility

- Becoming popular as a framework for domain-specific language (DSL) compilers for heterogeneous systems

MLIR is a **collection of dialects**, each representing different layers of abstraction through various operators, types, and attributes

- Custom dialects can easily be added for domain-specific problems while reusing existing infrastructure
- Dialects can be integrated into larger language stacks via lowering, transforming a more abstract dialect into a more concrete one

MLIR-based Compilation Flow


ML Framework
DSL
Src-to-Src (MLIR) Compiler+DSE
Annotated C code / LLVM IR / MLIR
IP requirements
Memory access patterns
Security/data requirements
Arch. Info
Mem. Info
Security/data requirements
IP config.
Mem. Gen. (Mnemosyne)
HLS (Vitis/Bambu)
System Integration (Olympus)
System-Level MLIR Compiler
Synthesis Tools

Convergence of multiple input flows

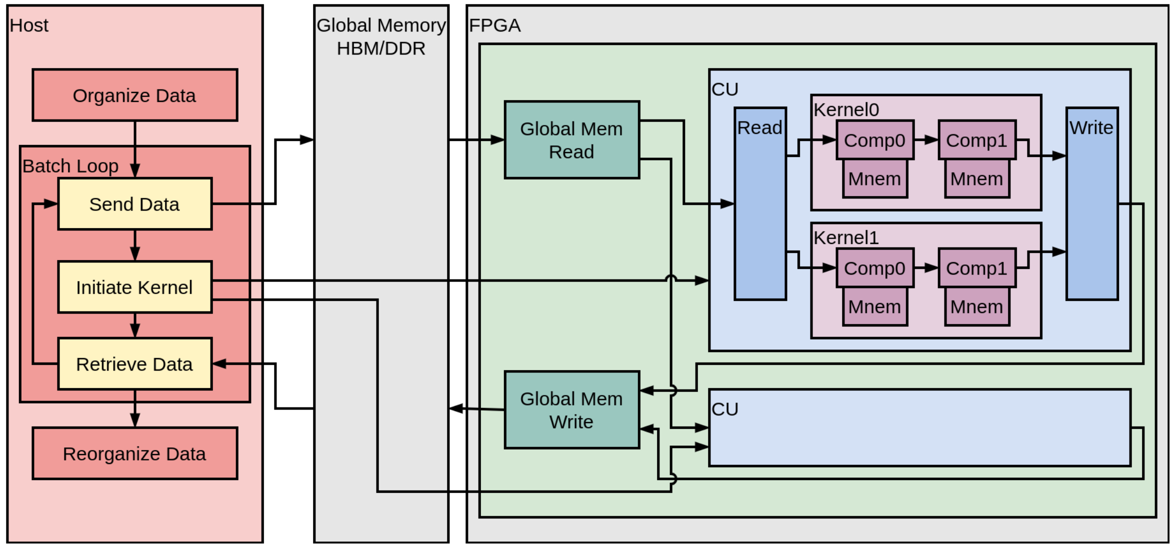
Memory architecture is decoupled from kernel HLS

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

Possibility to target different architectures

Olympus – System Generation Flow

Automatic system generation for FPGA accelerators



Parallel computing units

Inputs

- ★ DFG description (MLIR)
- ★ Characteristics of the target node(s)
- ★ Kernel descriptions

Outputs

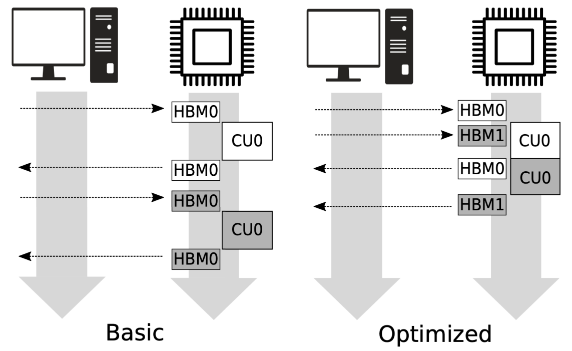
- ★ Synthesizable C++ code
- ★ Host library implementation
- ★ System configuration file

"Intelligent" policies to coordinate and/or protect data transfers

Olympus Optimizations

Double buffering

★ To hide latency of host-FPGA data transfers

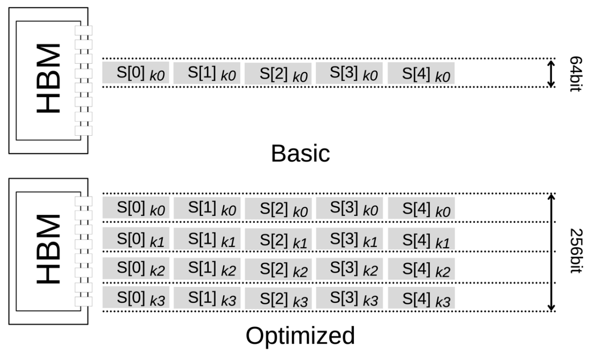


automatic batch sizing

Bus optimization and data interleaving

★ To maximize bandwidth (e.g., 256-bit AXI channels)

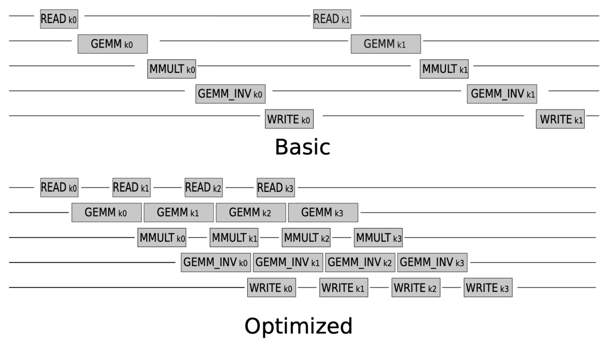
S. Soldavini, D. Sciuto, C. Pilato: **Iris: Automatic Generation of Efficient Data Layouts for High Bandwidth Utilization**. ASP-DAC (2023)


HBM
S[0] k0 S[1] k0 S[2] k0 S[3] k0 S[4] k0
64bit
Basic
HBM
S[0] k0 S[1] k0 S[2] k0 S[3] k0 S[4] k0
S[0] k1 S[1] k1 S[2] k1 S[3] k1 S[4] k1
S[0] k2 S[1] k2 S[2] k2 S[3] k2 S[4] k2
S[0] k3 S[1] k3 S[2] k3 S[3] k3 S[4] k3
256bit
Optimized

algorithms for efficient data layout on the bus

Dataflow execution model

★ To enable kernel pipelining


READ k0
READ k1
GEMM k0
GEMM k1
MMULT k0
MMULT k1
GEMM_INV k0
GEMM_INV k1
WRITE k0
WRITE k1
Basic
READ k0 READ k1 READ k2 READ k3
GEMM k0 GEMM k1 GEMM k2 GEMM k3
MMULT k0 MMULT k1 MMULT k2 MMULT k3
GEMM_INV k0 GEMM_INV k1 GEMM_INV k2 GEMM_INV k3
WRITE k0 WRITE k1 WRITE k2 WRITE k3
Optimized

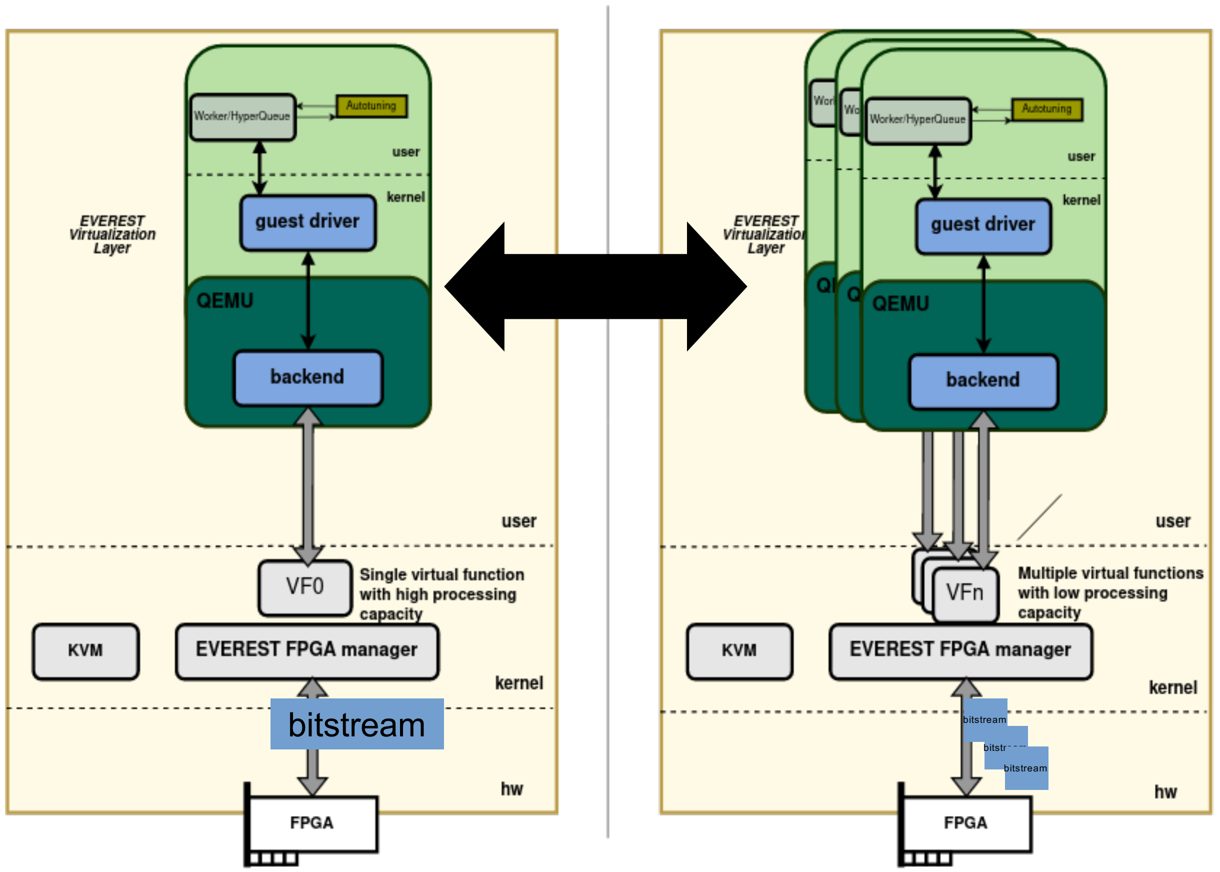
automatic (pre-HLS) code transformations

EVEREST

EVEREST Runtime Environment

The **EVEREST Runtime Environment** supports the selection of “variants” at runtime and in a virtualized manner:

- **SDK Compilation Framework** provides the accelerator variants – bitstream with metadata (e.g., number of clients-VMs, type of acceleration, etc)
- **Virtualization support (EFSM)** attaches them to newly-created VM at runtime and manage dynamic reallocation
- **Dynamic adaptation and autotuning (mARGOt)** take decisions depending on the execution status

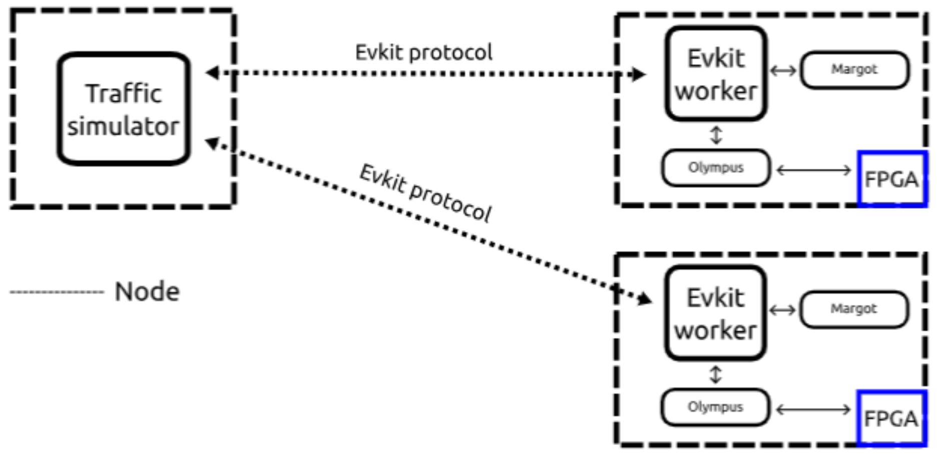


EVEREST

EVKIT

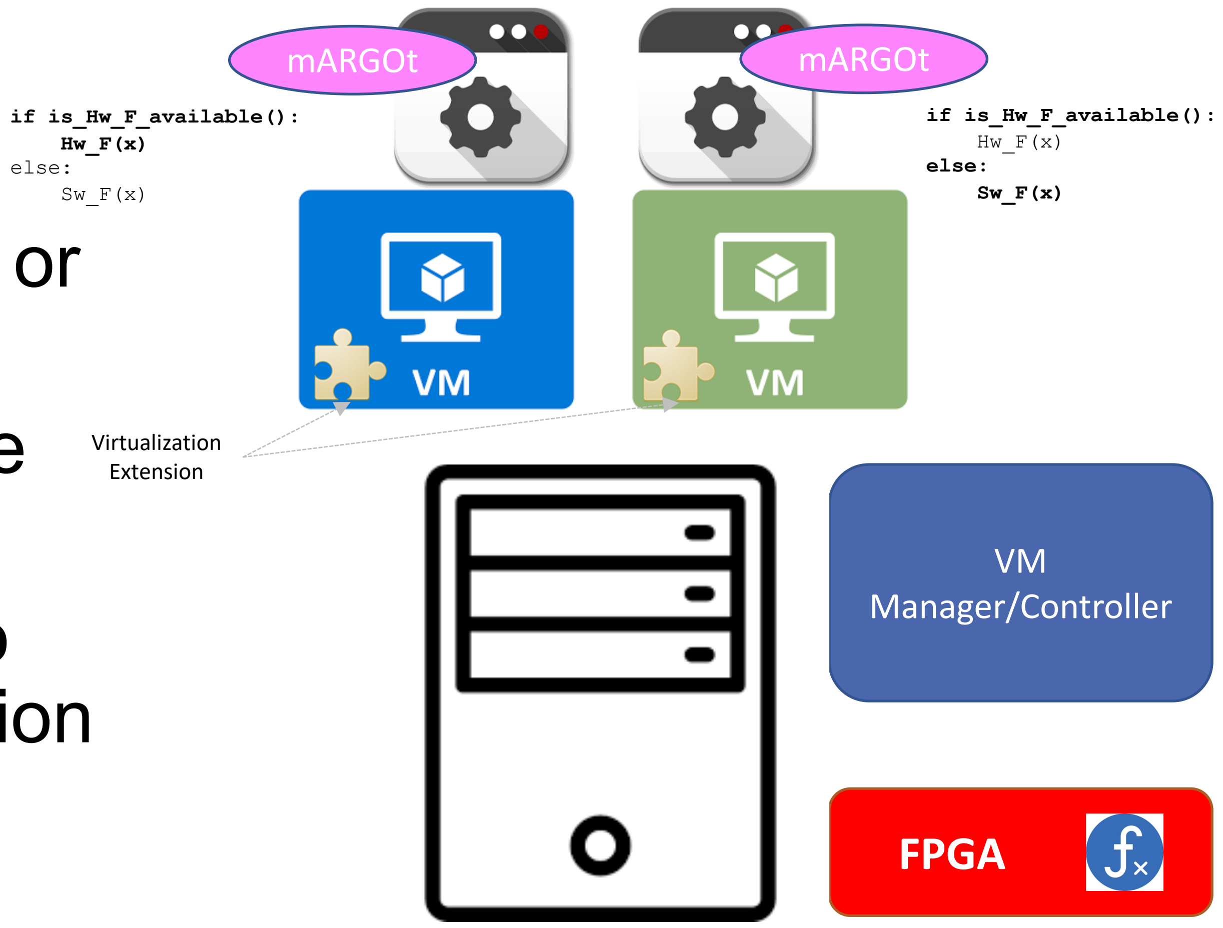
Distributed runtime library developed in EVEREST

- Simple frontend API available in Python and Rust
- **Distributes** and **load-balances** embarrassingly-parallel computations to a set of cluster nodes
- Supports **CPU and FPGA** execution of kernels
- Can cooperate with **mARGOt** to choose CPU/FPGA kernel variants or modify their parametrizations



EVEREST Runtime Features

- Possibility of **reconfiguring hardware function assignment** according to application feedback
- **Unique application instance** with or without hardware function
- **Dynamic switching** needed for the hardware functions
- **Feedback from the application** to permit the dynamic hardware function allocation



```
if is_Hw_F_available():
    Hw_F(x)
else:
    Sw_F(x)
```

```
if is_Hw_F_available():
    Hw_F(x)
else:
    Sw_F(x)
```

Conclusions

The **EVEREST SDK** is a collection of tools to **simplify the deployment** and **optimize the execution** of selected computational kernels on FPGA

★ Combination of compilation and runtime methods to match the execution of the applications and the characteristics of the underlying hardware

The **EVEREST Compilation Framework** is based on HLS to optimize the generation of hardware descriptions while reasoning at higher levels of abstractions

★ Highly based on MLIR infrastructure to unify the input flows and the optimizations
★ Possibility to combine different HLS tools in the same hardware architecture

The **EVEREST Runtime Environment** uses hardware/software variants in a virtualized environment to adapt the computation

★ Support for CPU/FPGA execution of kernels
★ Load-balancing methods among cluster nodes

Thanks!

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