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

FPL 2023 – September 6, 2023

Automatic system-level design for reconfigurable HPC applications: The EVEREST approach

CHRISTIAN PILATO

EVEREST Scientific Coordinator

christian.pilato@polimi.it

http://www.everest-h2020.eu

EVEREST: Big Data Analytics on FPGA

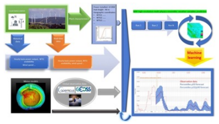
Project Coordinator: C. Hagleitner, IBM
Scientific Coordinator: C. Pilato, POLIMI
Web: everest-h2020.eu
LinkedIn: EVEREST Project
Facebook: @EVERESTH2020

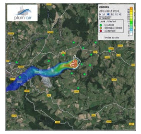
EVEREST | DESIGN ENVIRONMENT FOR EXTREME-SCALE BIG DATA ANALYTICS ON HETEROGENEOUS PLATFORMS

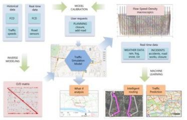
IBM POLITECNICO MILANO 1863 usi cima RESEARCH FOUNDATION VSB TECHNICAL UNIVERSITY OF OSTRAVA IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER NUMTECH

Big data applications with heterogeneous data sources

Three use cases







EVEREST SDK

Collection of interoperable and open-source tools to match target system, application workflow, and data characteristics

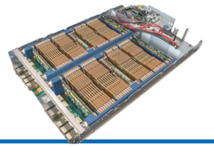
Compilation	Runtime
◉ Unified hardware generation flow based on **MLIR** to support multiple input flows ◉ Combination of **high-level synthesis** and **specialized memory architectures**	◉ Automatic **allocation** of target nodes and virtual machines ◉ **Autotuning** to match the application and the underlying hardware

FPGA-based architectures to accelerate selected kernels

CPU-based infrastructure

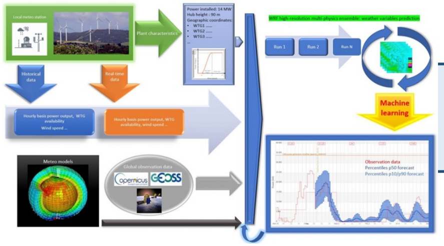
+

Two FPGA-based clusters



EVEREST Use Cases

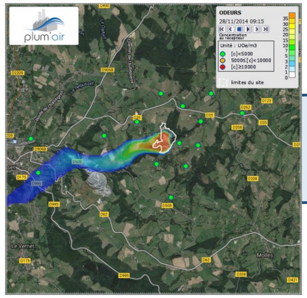
Accelerated computationally-intensive kernels + **Heterogeneous** data sources



Renewable energy production prediction

★ Improve **quality of the predictions** by combining weather models and plant info

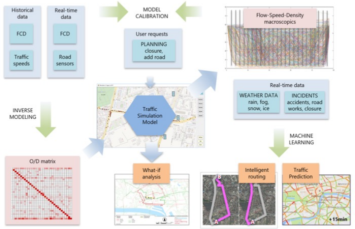
Weather prediction modelling (WRF)



Air-quality monitoring of industrial sites

★ Improve the **response time of predictions** by combining weather models and site actions

★ **Accelerate kernels** to execute more tests and create more (or more precise) models

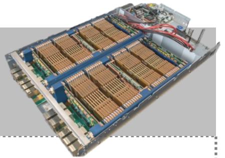


Traffic modeling for intelligent transportation

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

EVEREST Target System

cloudFPGA

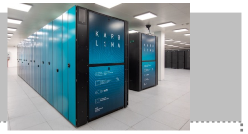


- **Disaggregated FPGAs** directly attached to the network (64 FPGA instances)
- **Low latency** and **high bandwidth** system
- **cFDK framework** for system generation
- Separation between **Shell** and **Role** modules

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

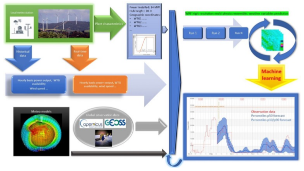
Multi-node support

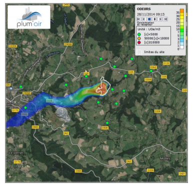
EVEREST

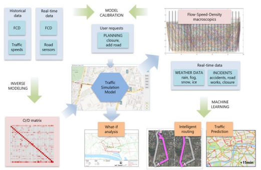
EVEREST Approach

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

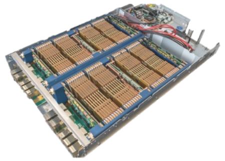
Open-source framework to support the optimization of selected workflow tasks

FPGA-based architectures to accelerate selected kernels

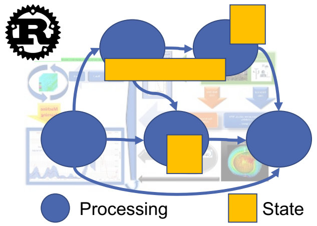
CPU-based infrastructure

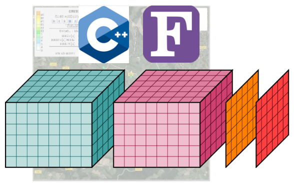
+

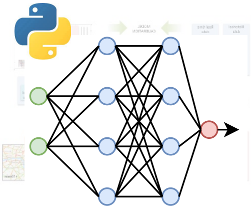
Two FPGA-based clusters



EVEREST System Development Kit (SDK)

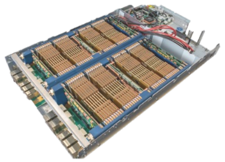






Different **input flows**
with different **input languages**

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

PandA

XILINX VITIS

cFDK

HEAppE

HyperLoom

(and more...)

Three Main Gaps

⚠ 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 FPGA tool flows*

how to talk with them?

⚠ 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?

⚠ System-level optimization

- ★ ***Gap between hardware designers and system designers***
- ★ *Accelerator design must consider also the implications for data transfers and FPGA synthesis*

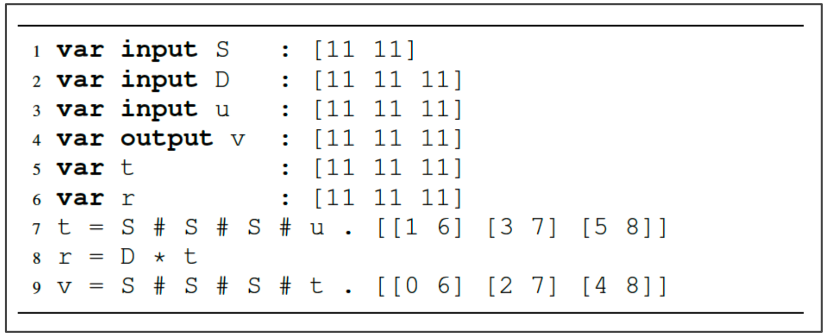
how to co-optimize the kernel and the system?

The Case of Computational Fluid Dynamics

Numerical simulation application that requires to solve **partial differential equations**

★Final result obtained by “small” contributions on independent data

★***Inverse Helmholtz operator*** *(three tensor operators) repeated millions of times – parameters **p***

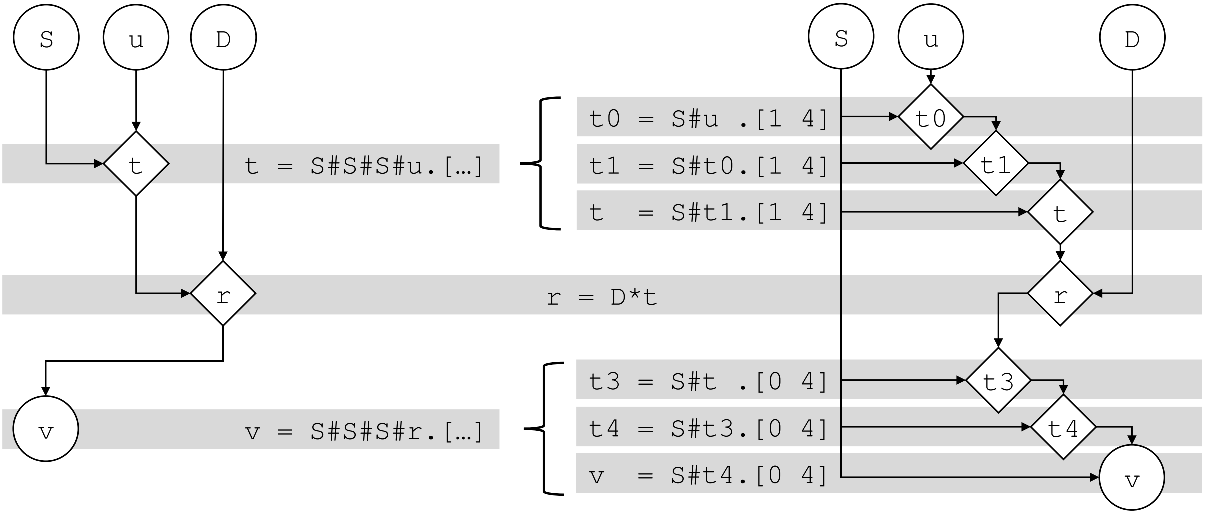


```
var input  S      : [11 11]
var input  D      : [11 11 11]
var input  u      : [11 11 11]
var output v      : [11 11 11]
var t             : [11 11 11]
var r             : [11 11 11]
t = S # S # S # u . [[1 6] [3 7] [5 8]]
r = D * t
v = S # S # S # t . [[0 6] [2 7] [4 8]]
```

$p^2 + 2 \cdot p^3$ *(double) elements* as input
21.74 KB (p = 11)

p^3 *(double) elements as output*
10.40 KB (p = 11)

$6 \cdot p^3$ *(double) elements as temporary*
62.39 KB (p = 11)



Total = 94.53 KB per kernel

Additional Data-Related Issues

⚠ **Application-specific optimizations**

★ For example, in Helmholtz, one of the tensors is constant over all elements and another is diagonal

how to specify them at the language level and exploit them during design?

⚠ **Limited BRAM resources**

★ *The **number of parallel kernels** can be limited*

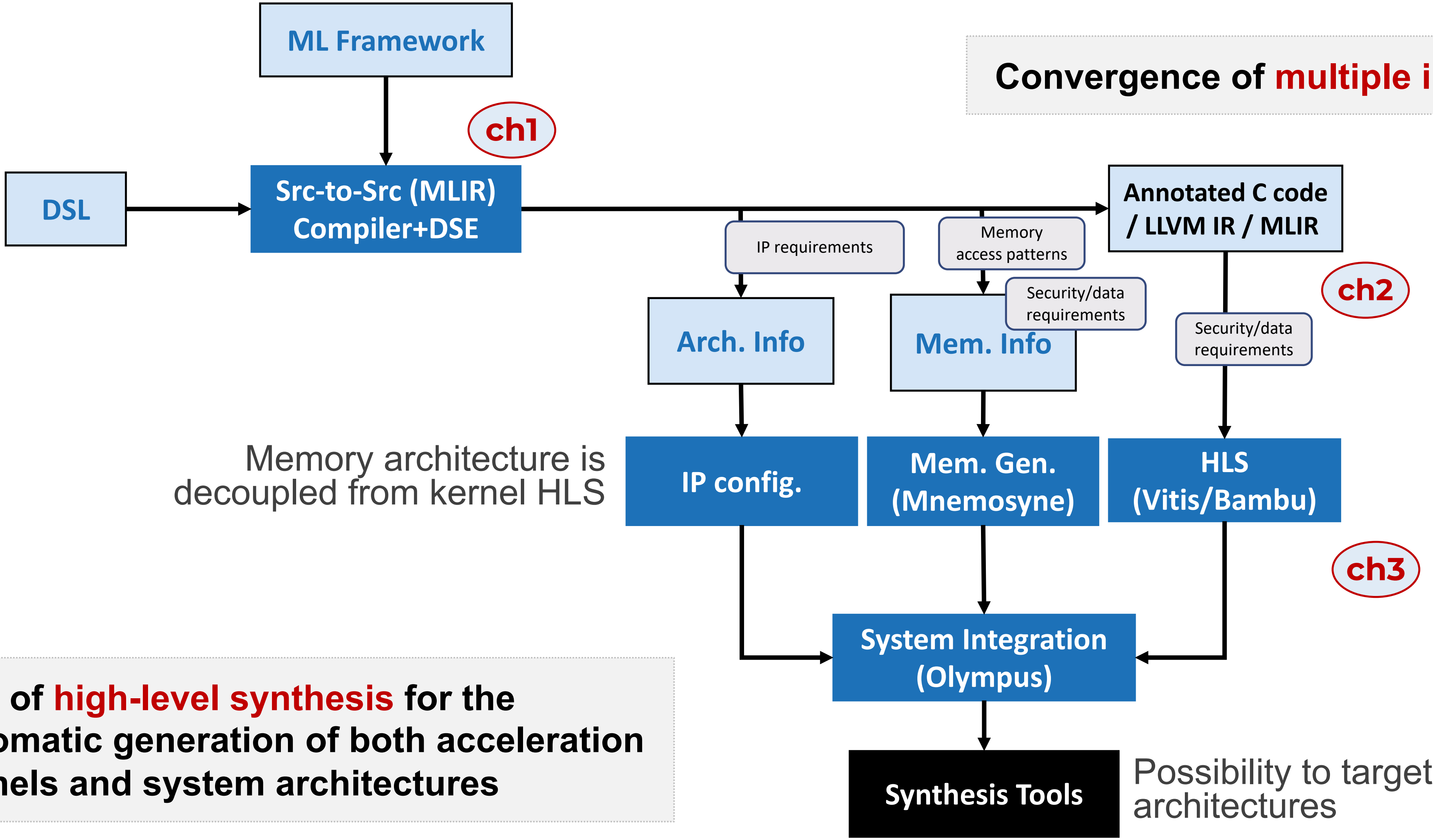
how to optimize local storage?

⚠ **System-level optimization**

★ *Creation of **batch of elements** to be executed in series by each kernel*

how to size the batches and hide communication latency?

MLIR-based Compilation Flow


ML Framework
ch1
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
ch2
Security/data requirements
IP config.
Mem. Gen. (Mnemosyne)
HLS (Vitis/Bambu)
ch3
System Integration (Olympus)
Synthesis Tools

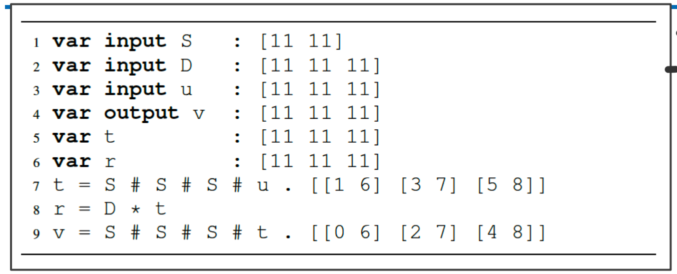
Convergence of multiple input flows

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

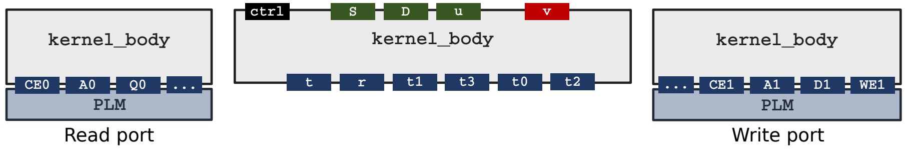
From DSL to Bitstream – Focus on Memory

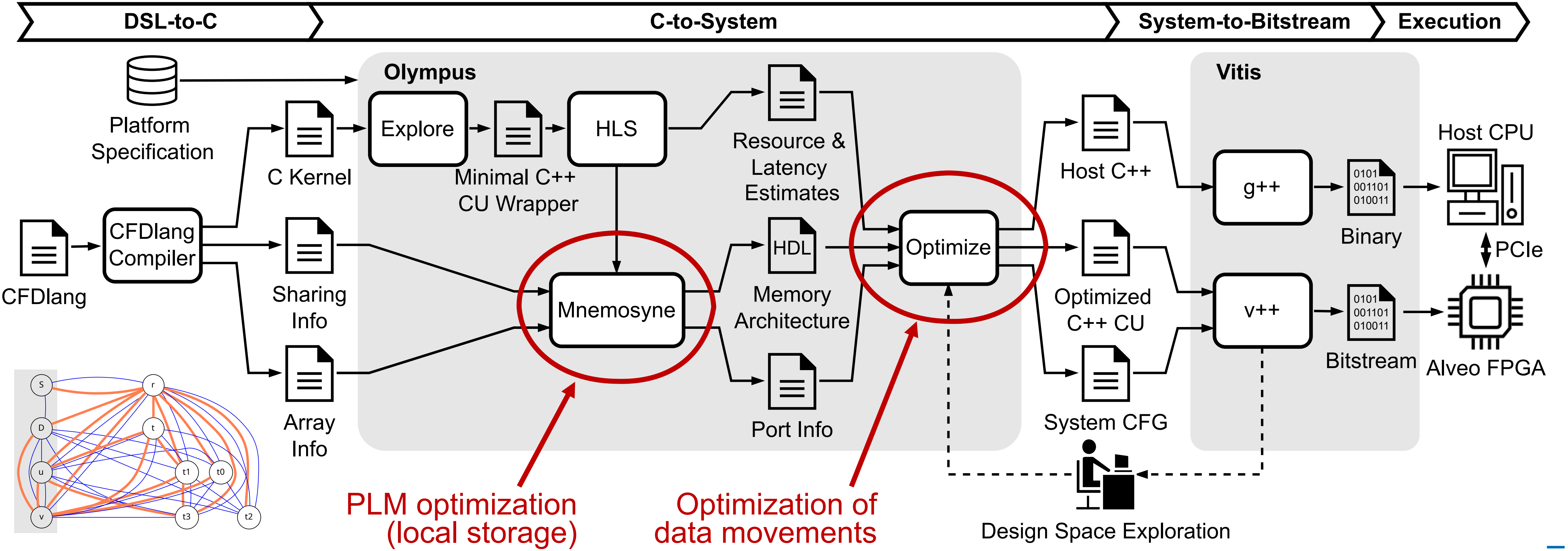


```
var input S   : [11 11]
var input D   : [11 11 11]
var input u   : [11 11 11]
var output v  : [11 11 11]
var t         : [11 11 11]
var r         : [11 11 11]
t = S # S # S # u . [[1 6] [3 7] [5 8]]
r = D * t
v = S # S # S # t . [[0 6] [2 7] [4 8]]
```

Memory architecture generation
is decoupled from kernel HLS

```
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])
```





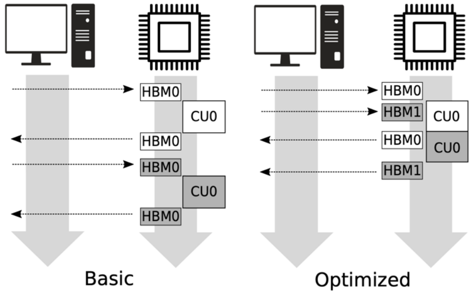
PLM optimization
(local storage)

Optimization of
data movements

Olympus Optimizations

Double buffering

★ To hide latency of host-FPGA data transfers

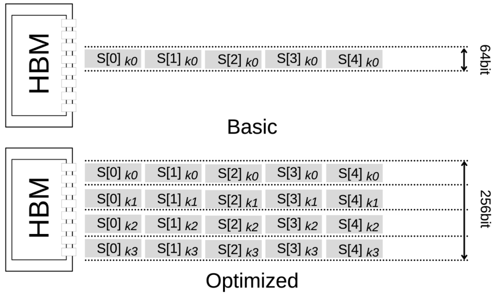


automatic batch sizing

Bus optimization and data interleaving

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

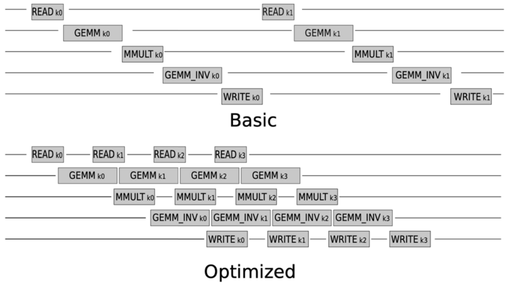
S. Soldavini, D. Sciuto, C. Pilato: **Iris: Automatic Generation of Efficient Data Layouts for High Bandwidth Utilization**. ASP-DAC (2023)



algorithms for efficient data layout on the bus

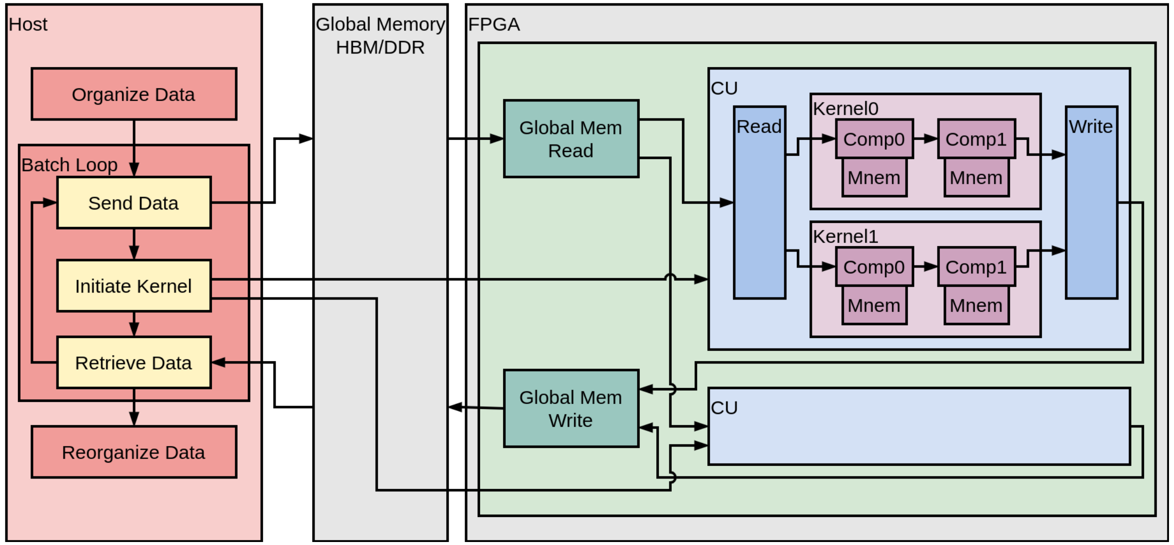
Dataflow execution model

★ To enable kernel pipelining



automatic (pre-HLS) code transformations

Olympus – System Generation Flow



Parallel computing units

Inputs

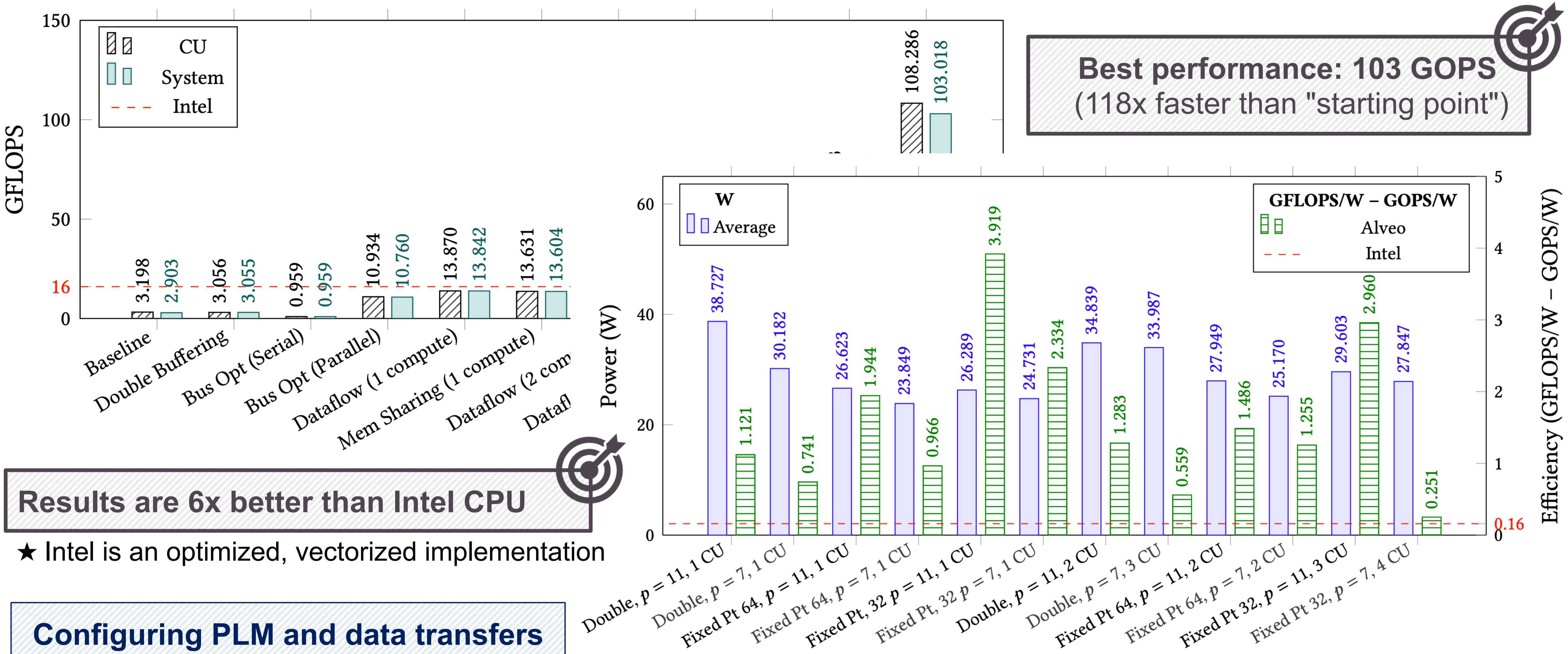
- ★ Algorithm parallelism
- ★ Characteristics of the target platform(s)
- ★ Interfaces of the modules (HLS tools)

Outputs

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

“Intelligent” policies to coordinate and/or protect data transfers

Results on HBM FPGA (Alveo u280)



Best performance: 103 GOPS
(118x faster than "starting point")

Results are 6x better than Intel CPU

★ Intel is an optimized, vectorized implementation

Configuring PLM and data transfers based on custom data formats

S. Soldavini, K. F. A. Friebel, M. Tibaldi, G. Hempel, J. Castrillón, C. Pilato: **Automatic Creation of High-Bandwidth Memory Architectures from Domain-Specific Languages: The Case of Computational Fluid Dynamics.** ACM TRETS (2022)

Conclusions

Accelerators are becoming **key components** in modern architectures

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

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

The increasing **design complexity** requires embracing **high-level synthesis** to increase productivity

★ Use of HLS to generate both **accelerator kernel** and the **associated memory architecture**

★ Possibility to **target different platforms**

EVEREST

Mnemosyne

Olympus

Thanks!

IBM

POLITECNICO
MILANO 1863

USI

cima
RESEARCH FOUNDATION

IT4I

NUMTECH

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