EVEREST

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

CPS Summer School – September 19, 2023

How to use HLS for building customized memory architectures

CHRISTIAN PILATO AND STEPHANIE SOLDAVINI

About Me

Associate Professor

Website: http://pilato.faculty.polimi.it

PhD Student
2008-2011

Research Assistant
2011-2013

Postdoc Research Scientist
2013-2016

Postdoc Research Assistant
2016-2018

Assistant Prof.
2018-*2023*

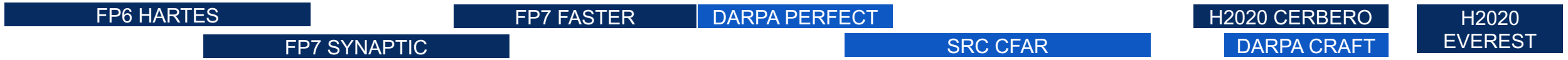
R&D Internship
6 months
NANGATE

Visiting Researcher
3 months
CHALMERS

Visiting Researcher
4 months
TU Delft

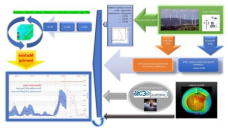
Visiting Researcher
9 months
NYU

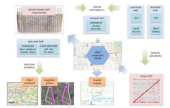
R&D Projects



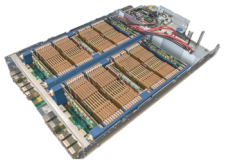
EVEREST

The EVEREST Project





Big data applicati
heterogeneous d



- App designers are not FPGA experts
- Hardware accelerators require many
- an have different

H2020 Project – 10 partners, 6 countries

Project Coordinator: Christoph Hagleitner, IBM Research Europe

Scientific Coordinator:

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

daptation to variants

Virtualization of resources

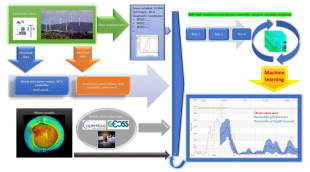
Multi-node support

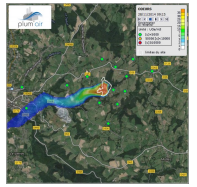
EVEREST

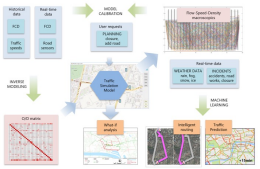
EVEREST Approach

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

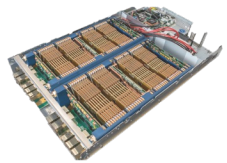
Open-source framework to support the optimization of selected workflow tasks

FPGA-based architectures to accelerate selected kernels

CPU-based infrastructure

+

Two FPGA-based clusters



EVEREST Partners

IBM Reseach Lab, Zurich (Switzerland)
Project administration, prototype of the target system
PI: Christoph Hagleitner

Università della Svizzera italiana (Switzerland)
Data security requirements and protection techniques
PI: Francesco Regazzoni

Centro Internazionale di Monitoraggio Ambientale (Italy)
Weather prediction models
PI: Antonio Parodi

Virtual Open Systems (France)
Virtualization techniques, runtime extensions to manage heterogeneous resources
PI: Michele Paolino

Numtech (France)
Application for monitoring the air quality of industrial sites
PI: Fabien Brocheton

Politecnico di Milano (Italy)
Scientific coordination, high-level synthesis, flexible memory managers, autotuning
PI: Christian Pilato

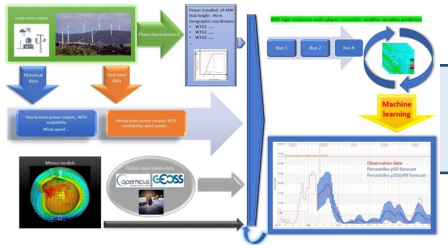
TU Dresden (Germany)
Domain-specific extensions, code optimizations and variants
PI: Jeronimo Castrillon

IT4Innovations (Czech Republic)
Exploitation leaders, large HPC infrastructure, workflow libraries
PI: Katerina Slaninova

Duferco Energia (Italy)
Application for prediction of renewable energies
PI: Lorenzo Pittaluga

Sygic A/S (Slovakia)
Application for intelligent transportation in smart cities
PI: Radim Cmar

EVEREST Use Cases

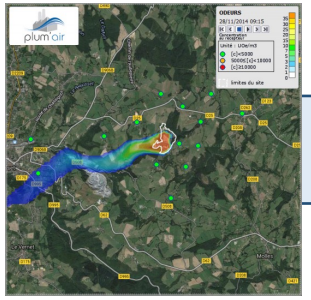


Renewable energy production prediction

★ Improve **quality of the predictions**

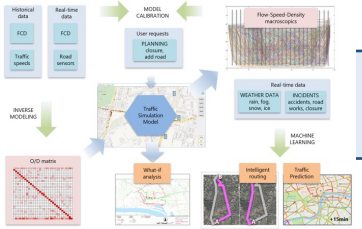
Weather prediction modelling (WRF)

★ Accelerate kernels to execute more tests



Air-quality monitoring of industrial sites

★ Improve the **response time of predictions**



Traffic modeling for intelligent transportation

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

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

EVEREST

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

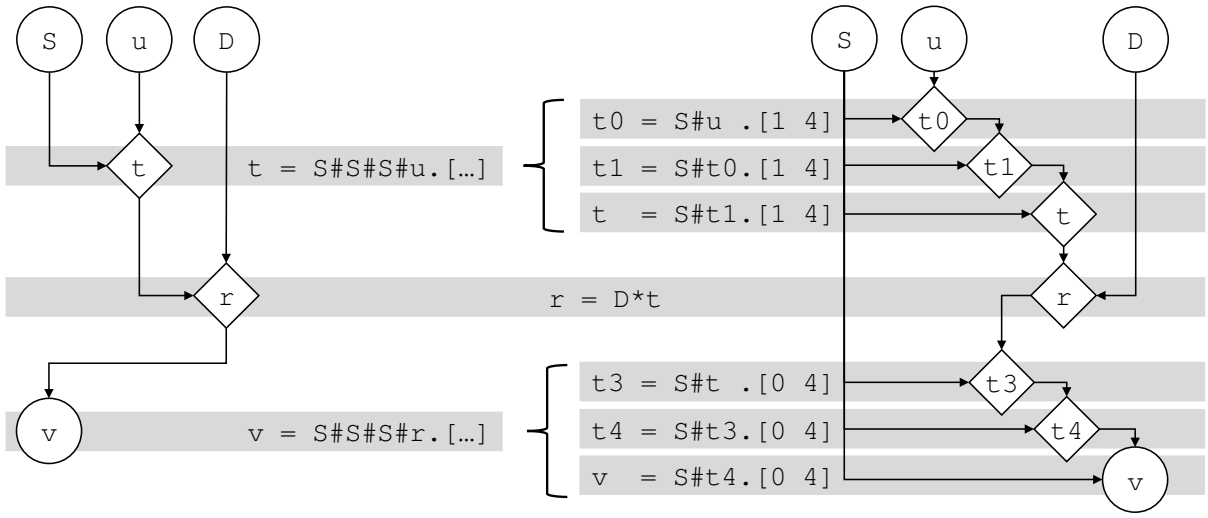
★***Inverse Helmholtz operator*** *(three tensor operators) repeated millions of times – parameters* ***p***

```
1 var input S    : [11 11]
2 var input D    : [11 11 11]
3 var input u    : [11 11 11]
4 var output v   : [11 11 11]
5 var t          : [11 11 11]
6 var r          : [11 11 11]
7 t = S # S # S # u . [[1 6] [3 7] [5 8]]
8 r = D * t
9 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)

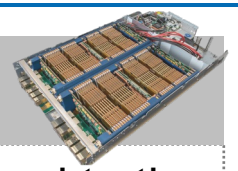


Total = 94.53 KB per kernel

EVEREST

EVEREST Target System

cloudFPGA



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

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**

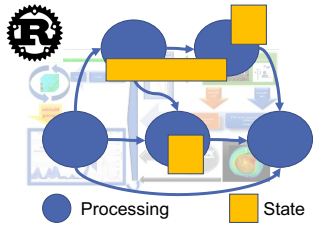
Different nodes to better **match applications**

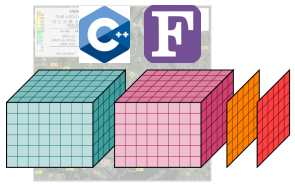
Data-intensive (memory-bound) applications

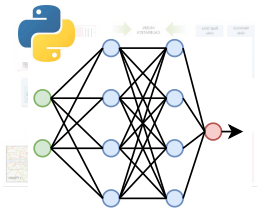
Seamless support for multiple nodes

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

EVEREST System Development Kit (SDK)

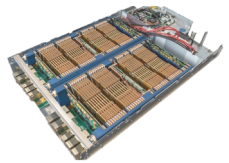






Different **input flows**
starting from different **input languages**

Support for **multiple target boards**



Collection of interoperable and open-source tools to create hardware/software systems that can adapt to the target system, the application workflow, and the data characteristics

- Compilation framework based on **MLIR** to unify the input languages
- **High-level synthesis** and **hardware generation flow** to automatically create optimized architectures
- Creation of hardware and software variants to match architecture features
- Use of state-of-the-art frameworks and commercial toolchains for FPGA synthesis

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

EVEREST

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?

Hardware HPC (Memory) Architectures

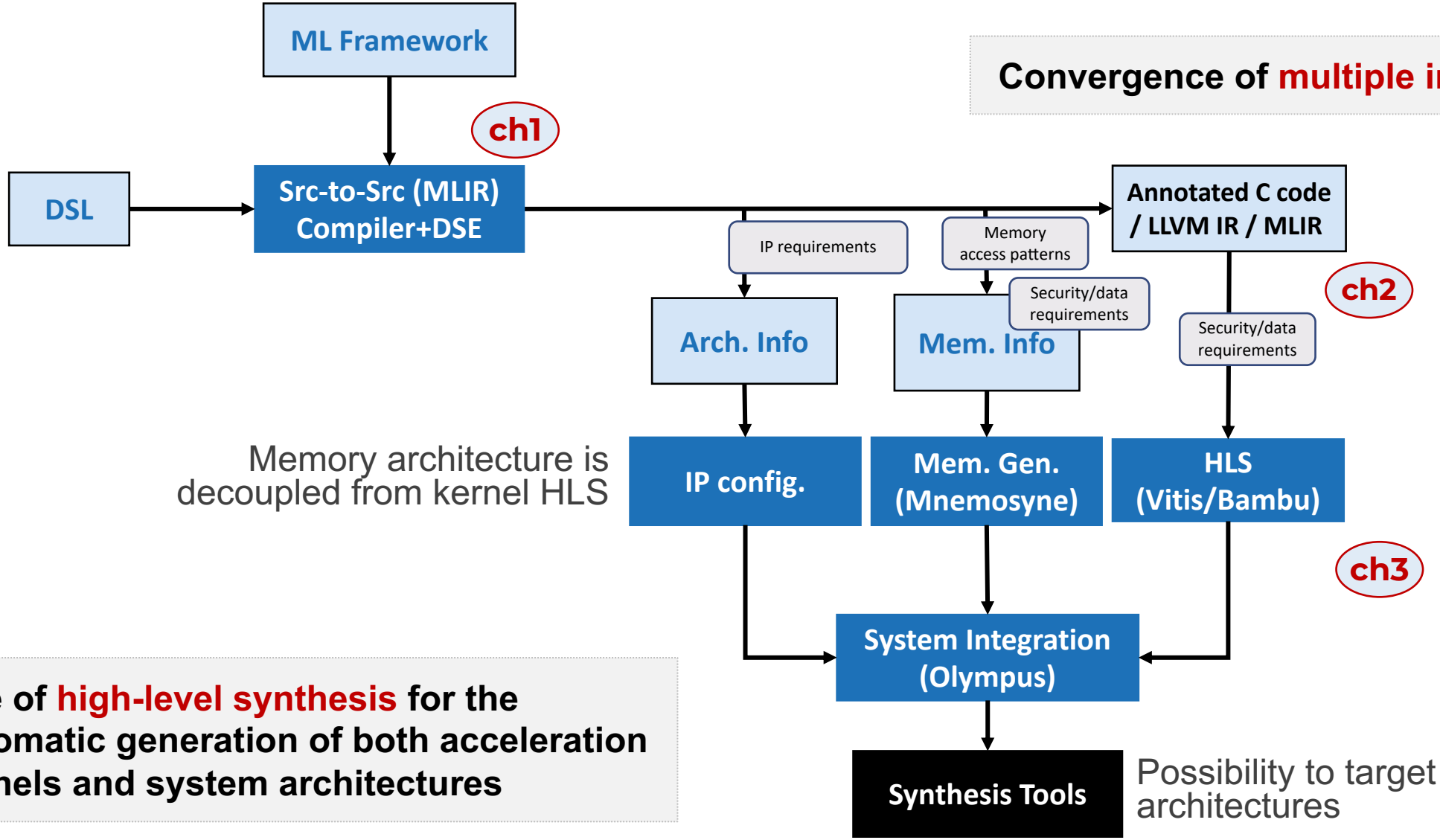
What do we mean with **memory architecture**?

Every hardware module that is responsible to provide data to the accelerator kernels

Additional issues:

- BRAM resources are limited
  - Helmholtz operator requires >94 KB of local data
  - If local storage is not optimized, the number of parallel kernels can be limited
- Application-specific details can be used to optimize the data transfers
  - In Helmholtz, one of the tensors is constant over all elements – how to match these details with platform characteristics?
  - Better to transfer data for a “batch” of elements and then execute them in series – how many? again, limited storage

MLIR-based Compilation Flow



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

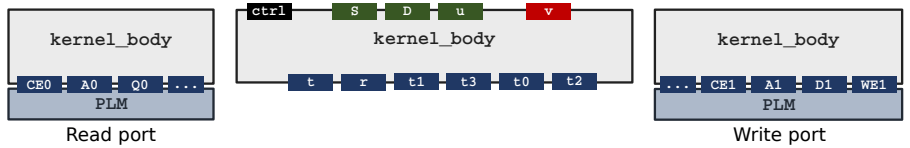
EVEREST

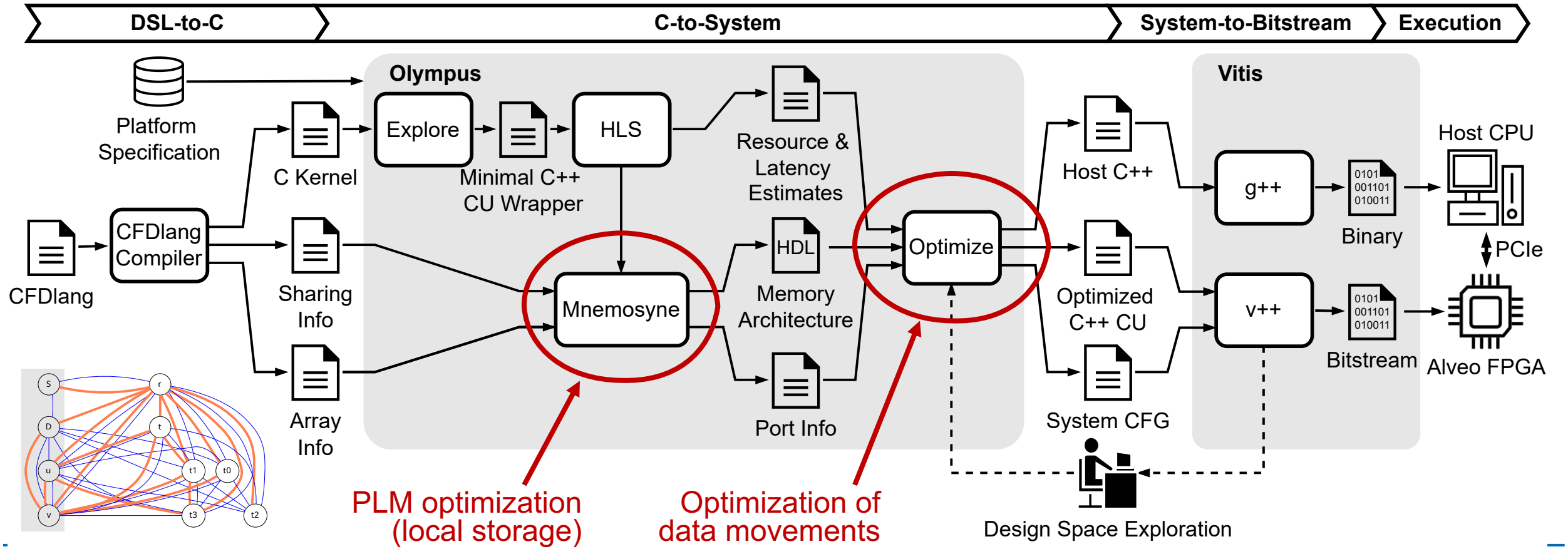
From DSL to Bitstream – Focus on Memory

```
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]]
```

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])
```





EVEREST

Mnemosyne – Reuse What is not Used

Generally, we use one **PLM unit** (possibly composed of many banks) for each **data structure** (array)

Reuse the same memory IPs
for several data structures

"Two data structures are compatible if they can be allocated to the same PLM unit (memory IPs)"

A common case: accelerator kernels never executed at the same time

- Possible only at system-level, when integrating the components
- Optimizations of accelerator logic and memory subsystem are independent

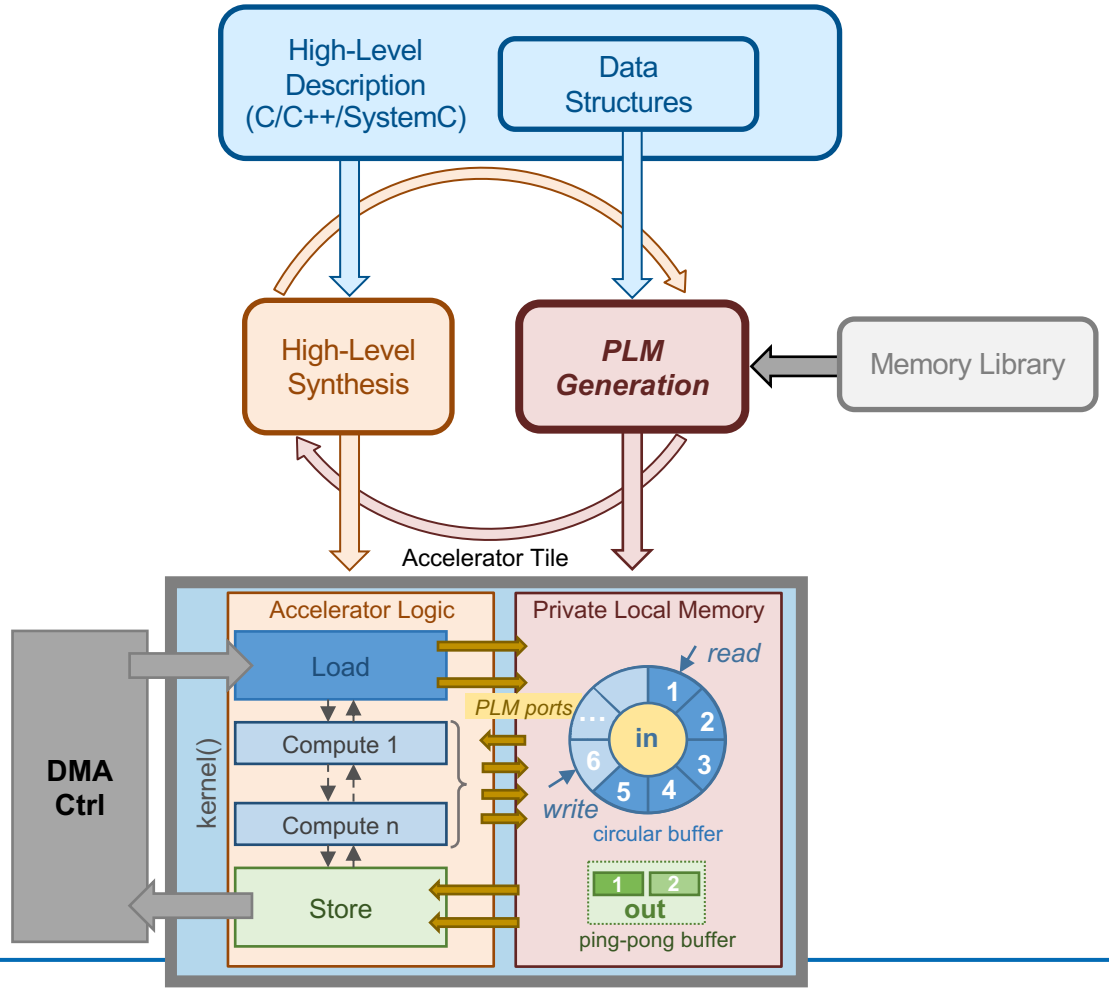
PLM Customization for Heterogeneous SoCs

High-Level Synthesis (HLS) to create the accelerator logic

- Definition of memory-related parameters (e.g., number of process interfaces)

Generation of **specialized PLMs**

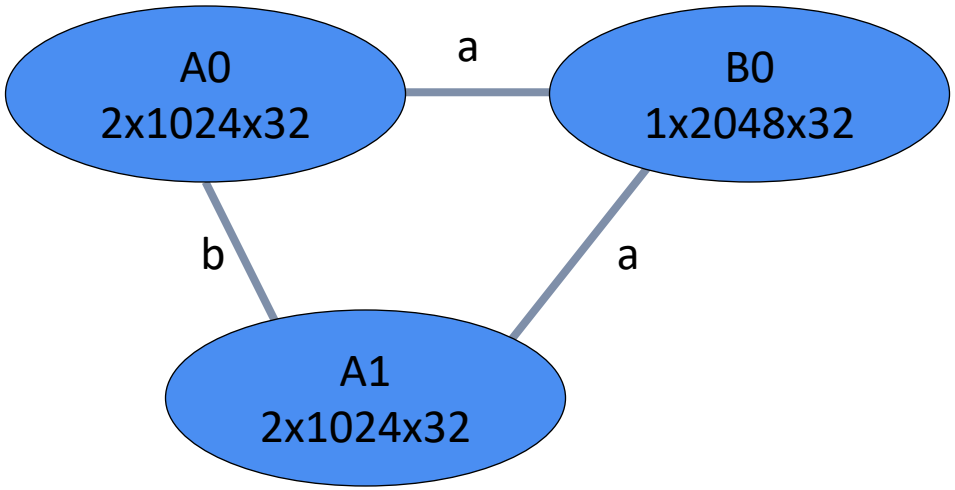
- Technology-related optimizations
- Possibility of system-level optimizations across accelerators



Memory Compatibility Graph (MCG)

Graph to represent the possibilities for optimizing the data structures

- Each node represents a data structure to be allocated, annotated with its data footprint (after data allocation)
- Each edge represents compatibility between the two data structures
- Can be automatically extracted from the MLIR-based compiler flow
  - Variant exploration to achieve the "best solutions"

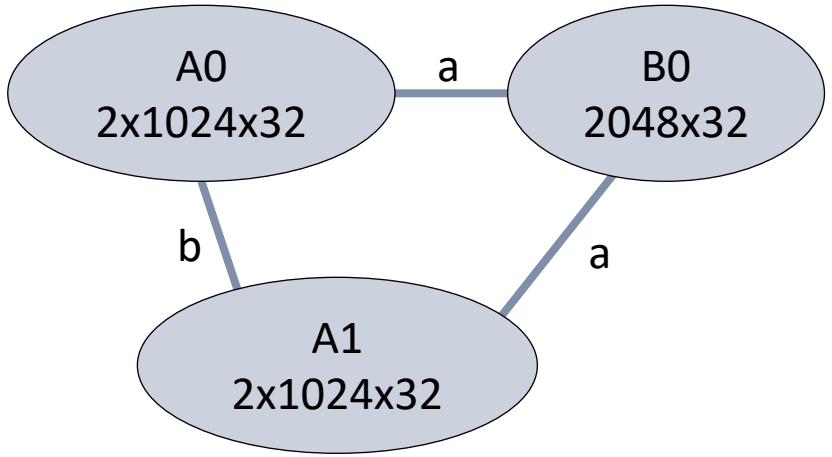


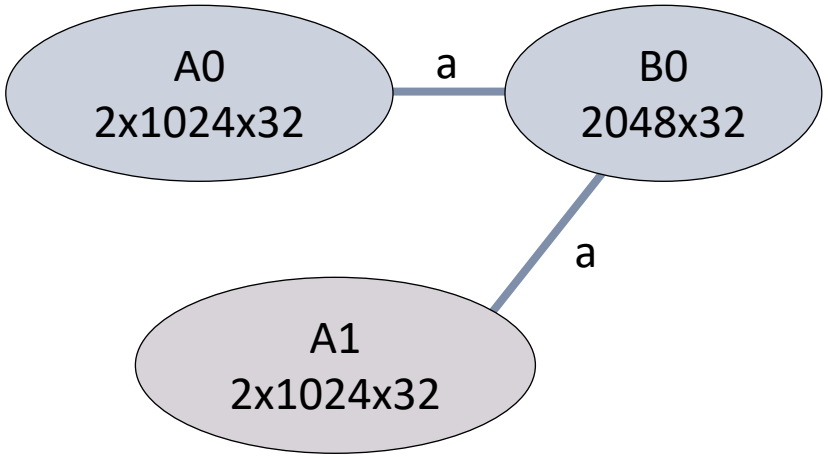
a) **Address-space compatibility**: the data structures are compatible and can use the same memory IPs
b) **Memory-interface compatibility**: the ports are never accessed at the same time and the data structures can stay in the same memory IP

Clique Definition

“A clique is a subset of the vertices of the memory compatibility graph such that every two vertices are connected by an edge”

A clique represents a **set of data structures** that can share the same memory IPs



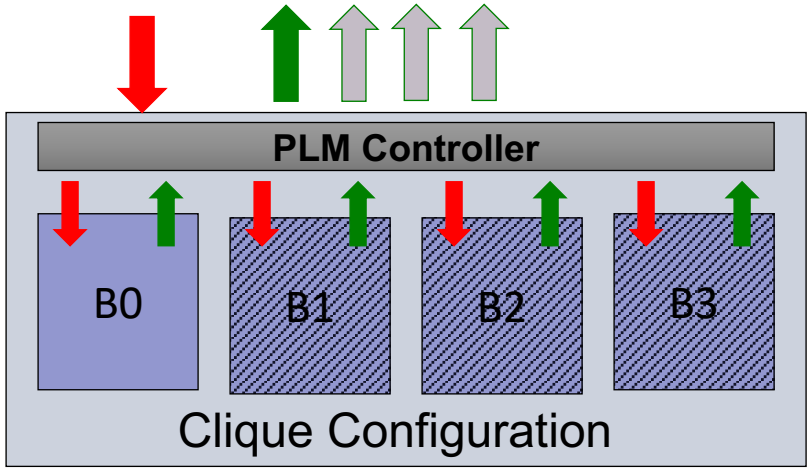


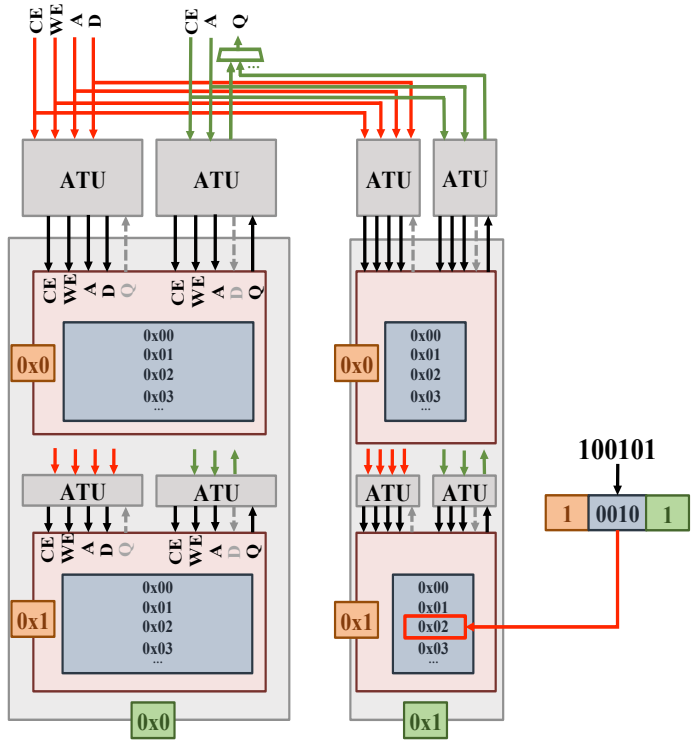
We need two distinct configurations!
{A0,B0} and {A1} or {A1,B0} and {A0}?

PLM Controller Generation

A **lightweight PLM controller** is created for each compatibility set (clique) based on the bank configuration

- Accelerator logic is not aware of the actual memory organization
- Array offsets need to be translated into proper memory addresses





Custom logic with negligible overhead, especially when the number of banks and their size is a power of two

Olympus – Automated System-Level Integration

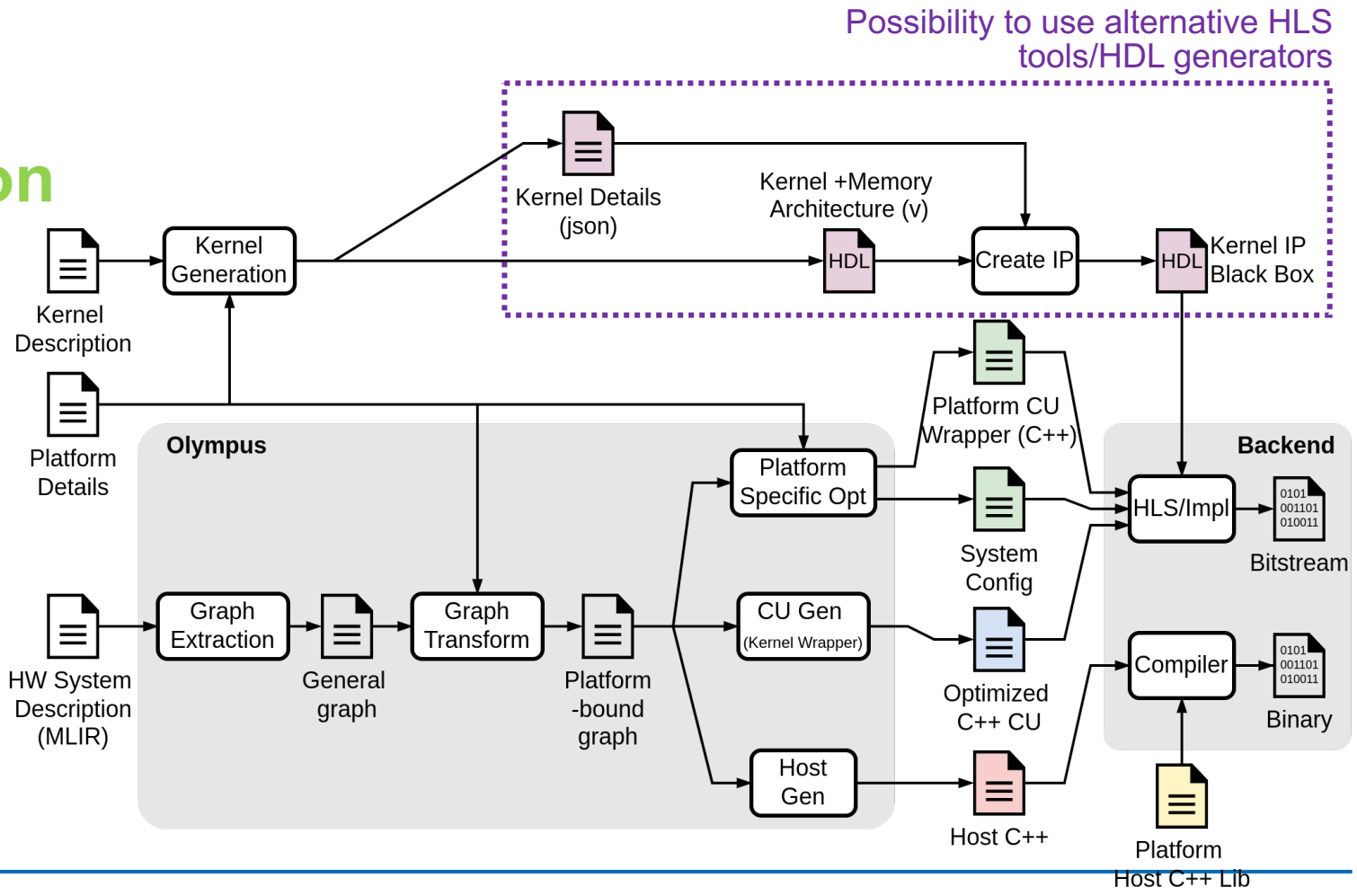
Complete **hardware architecture generation flow** from high-level specification

Platform-specific description

- HBM-based Xilinx Alveo
- IBM CloudFPGA
- …

Host code generation

- Based on **platform libraries** for the specific target

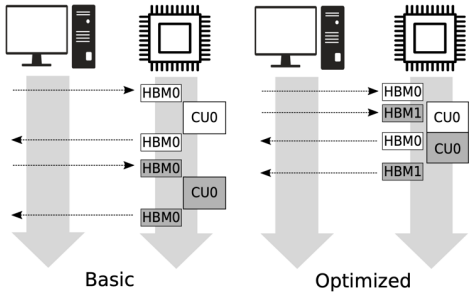


EVEREST

Olympus Optimizations

Double buffering

★ To hide latency of host-FPGA data transfers

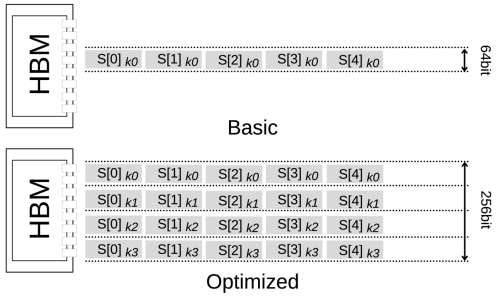


automatic batch sizing

Bus optimization and data interleaving

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

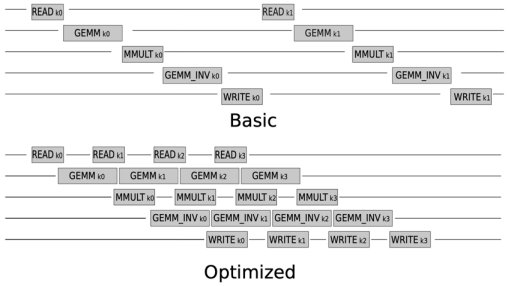
S. Soldavini, D. Sciuto, C. Pilato: **Iris: Automatic Generation of Efficient Data Layouts for High Bandwidth Utilization**. ASP-DAC (2023)



algorithms for efficient data layout on the bus

Dataflow execution model

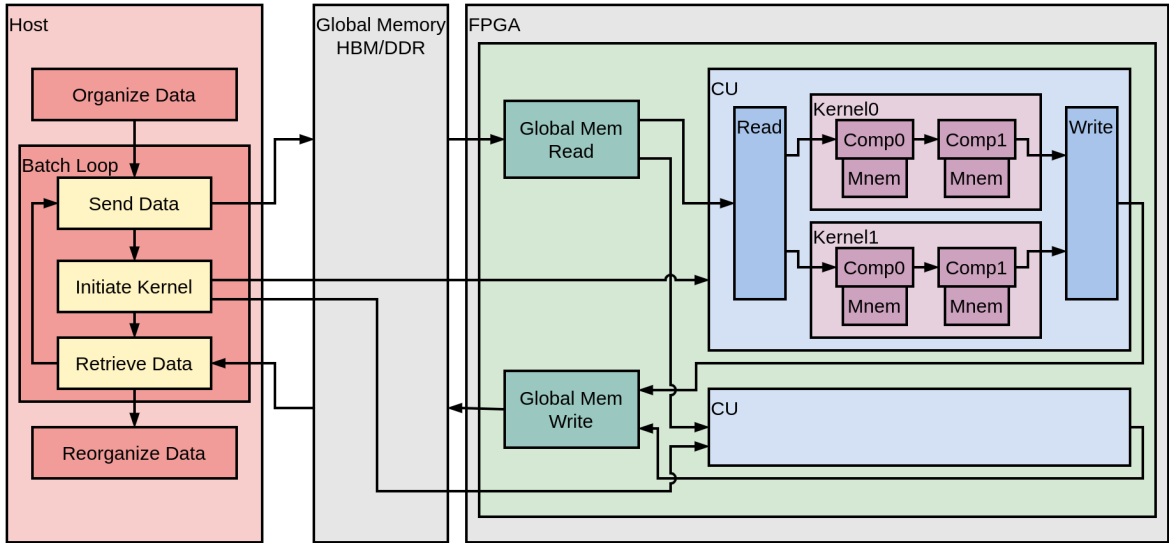
★ To enable kernel pipelining



automatic (pre-HLS) code transformations

EVEREST

Olympus – System Generation Flow



Parallel computing units

Inputs

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

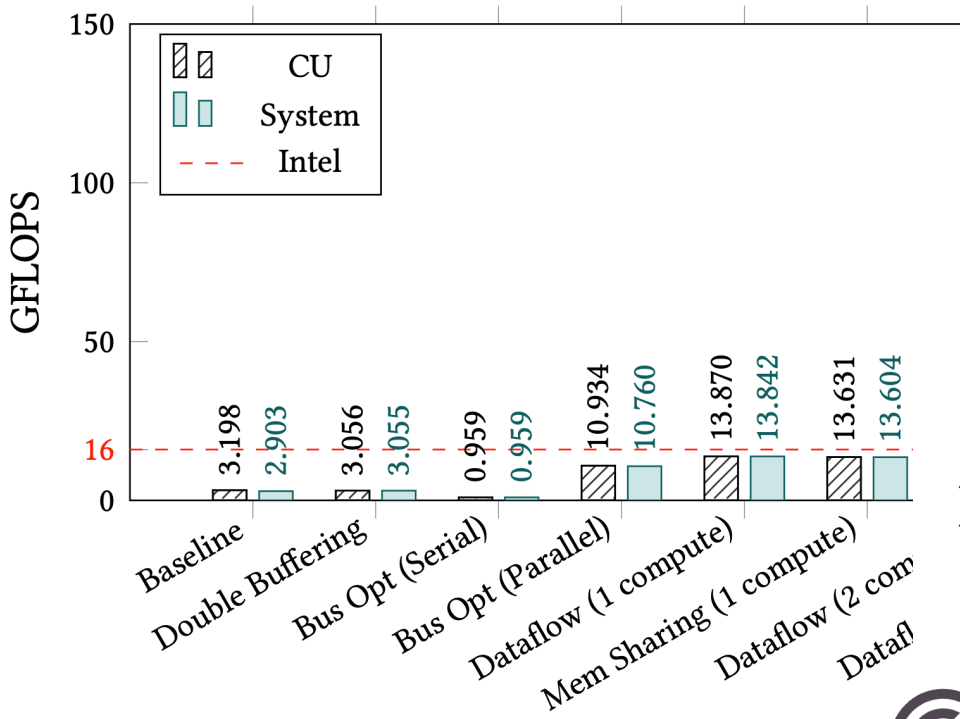
Outputs

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

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

EVEREST

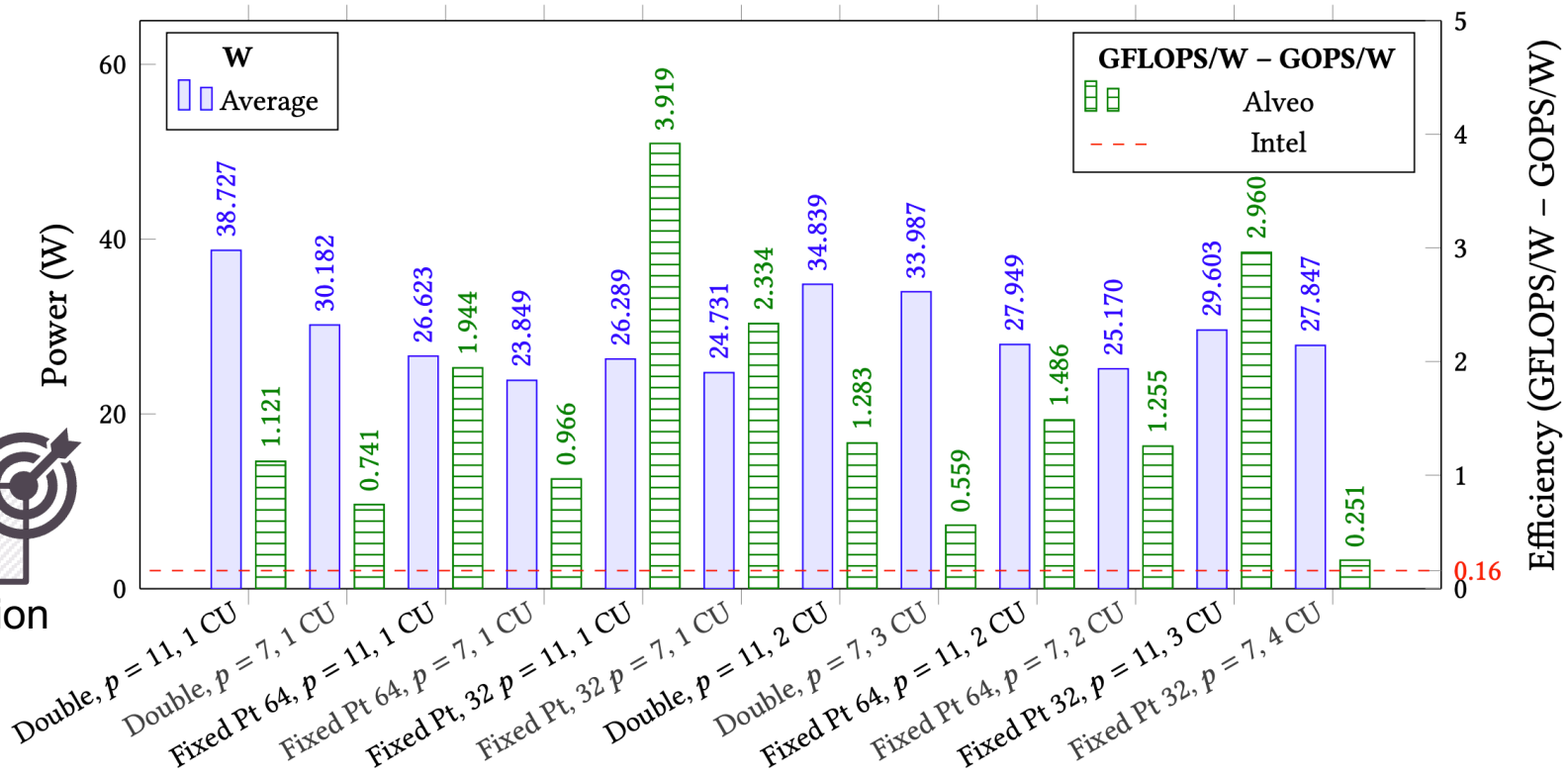
Results on HBM FPGA (Alveo u280)



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)

EVEREST

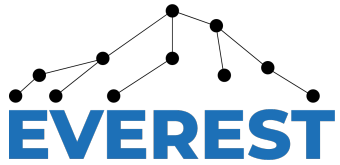
Conclusions

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**



Mnemosyne

Olympus

Thanks!

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