EVEREST

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

cFDevOps: Workshop on DevOps Support for Cloud FPGA platforms @ FPL 2021

Climbing EVEREST: A design environment for extreme-scale big data analytics on heterogeneous platforms

---

CHRISTIAN PILATO

Politecnico di Milano, Scientific Coordinator

christian.pilato@polimi.it

MICHELE PAOLINO

Virtual Open System, Virtualization Leader

m.paolino@virtualopensystems.com

EVEREST: Using cloudFPGA for Big Data Analytics

H2020 project funded under the call – "**Big Data technologies and extreme-scale analytics**" [Kick-off on Oct 1, 2020][http://www.everest-h2020.eu]

- Big focus on FPGA acceleration in data centers and related issues ➔ **cloudFPGA**

Key idea:

- a coordinated action with the appropriate technology areas (e.g., AI, analytics, software engineering, HPC, Cloud technologies, IoT and edge/fog/ubiquitous computing) ➔ **FPGA acceleration in (federated) data centers**
- system engineering/tools to contribute to the co-design of federated/distributed systems ➔ **EVEREST system development kit**

adaptive memory management

architectures for collecting, managing and exploiting

virtualization

hardware acceleration

data security

high-level synthesis

standardized interconnection methods

domain-specific extensions

runtime management

Application Concepts

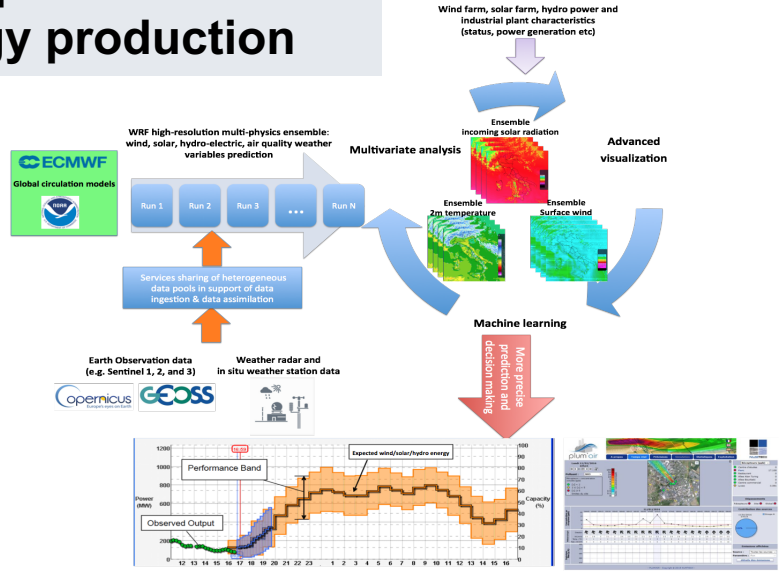
Three use cases **provided by the application partners**

- Looking for hardware acceleration (intense data computation) with efficient and secure data management (distributed data sources)
- Possibility of AI/ML-based decision making
- Combination of the tasks in different pipelines

Air-quality monitoring in industrial sites

Traffic modelling for intelligent transportation

Weather-based prediction of renewable energy production



EVEREST

EVEREST cloudFPGA: Key Features

Network-attached solution composed of:

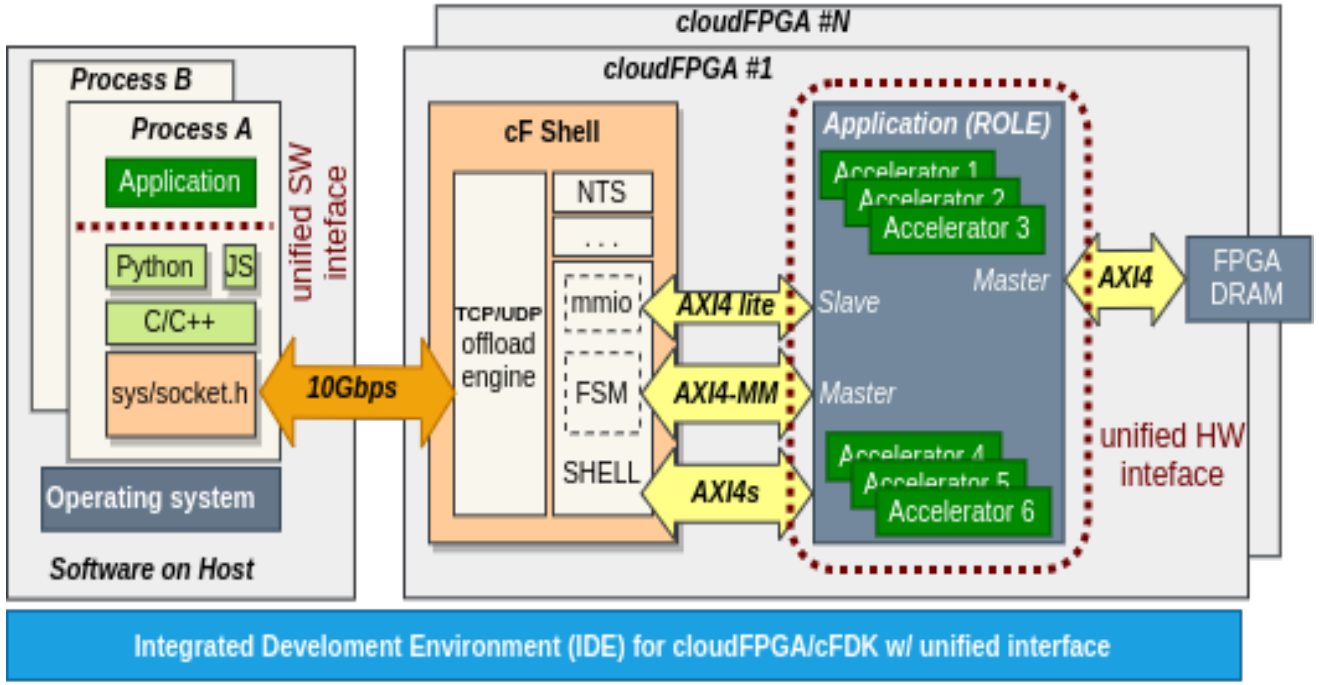
- Interface logic already designed (**cF Shell**) to support system integration
  - TPC/UDP communication is managed transparently to the user logic
- User logic (**ROLE**) that can be easily designed and customized) with traditional HLS tools

Application code running on host

- Low-level libraries for host-FPGA communication

Possibility to use a cluster of FPGAs

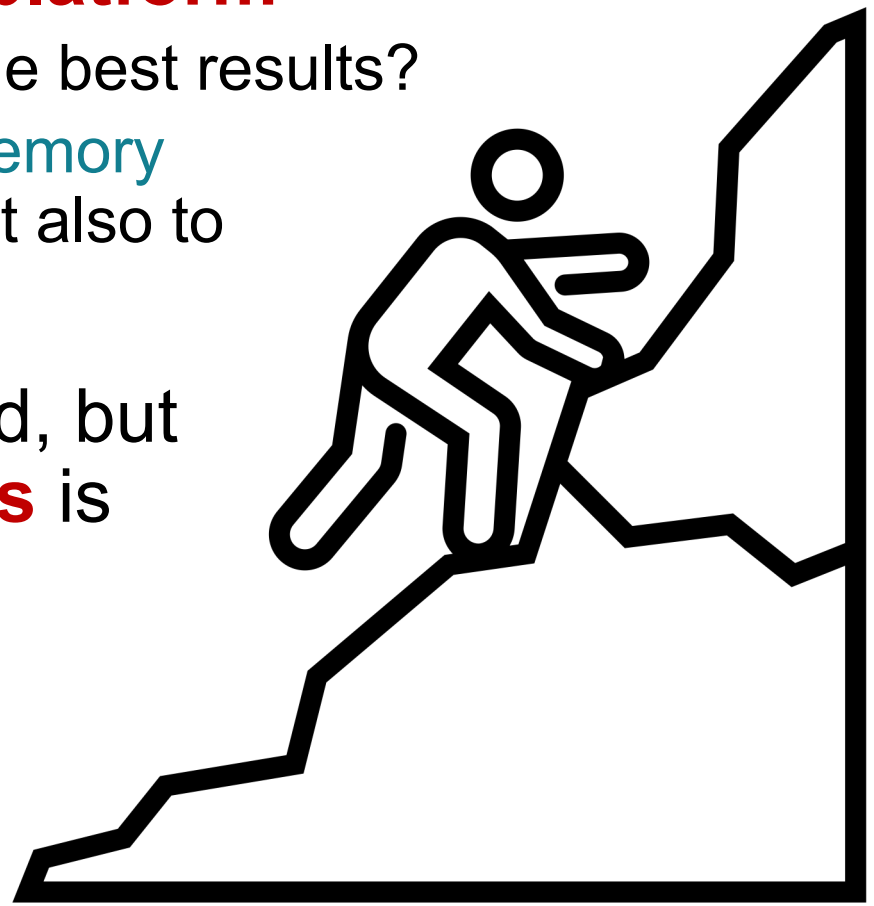
- IDE for allocation and management of resources



Climbing EVEREST: Obstacles on the Road

Programmability: Application developers have often **limited hardware skills** and **limited knowledge of the target platform**

- How to specify the application functionality to get the best results?
- How to design the hardware accelerator and the memory subsystem not only to optimize the performance but also to avoid bottleneck



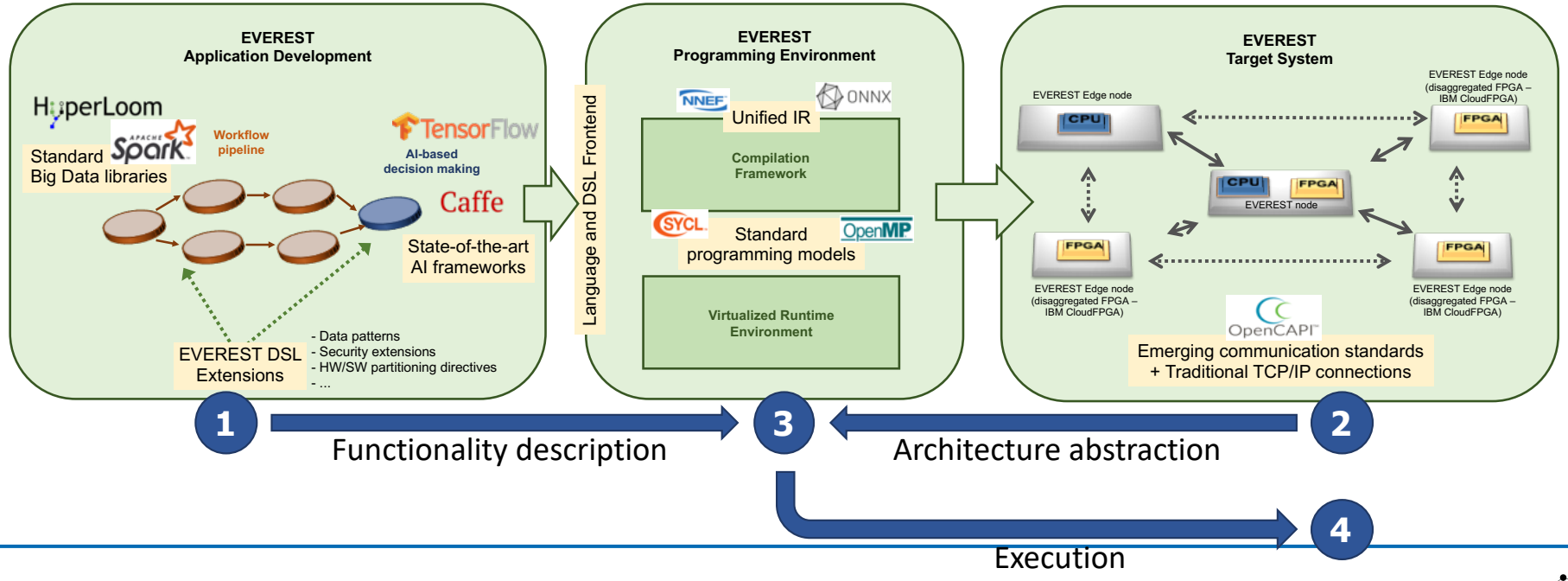
Portability: Designing a FPGA system is hard, but **designing an application for many systems** is even harder

- How to specify a platform-agnostic functionality?
- How to match such functionality with the actual hardware?
- How to deal with dynamic changes?

EVEREST SDK: System Development Kit

Coordinated design environment composed of **four major phases:**

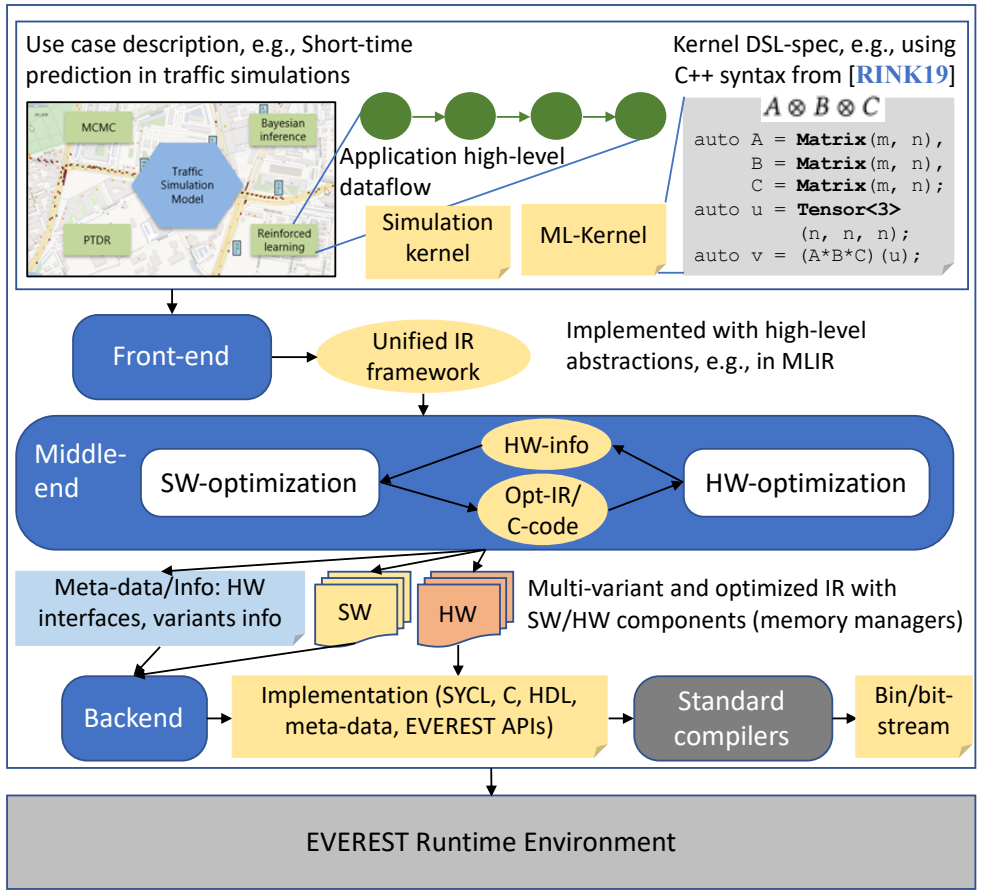
1. **Application specification** (data ▻ application and requirements)
2. **Architecture abstraction** (target system ▻ arch. description)
3. **Programming environment** (app+arch+reqs desc. ▻ hw/sw bin.)
4. **Execution monitoring and management** (hw/sw bin. ▻ execution)



EVEREST

EVEREST Programming Environment

1. **Compilation Environment:** analyzes application and creates all "variants" based on *architecture abstraction* and *application/data requirements*
   - **Unified IR framework** (MLIR)
   - **Hardware acceleration and High-level synthesis** (Bambu, Vivado/Vitis HLS)
   - **Integration of non-functional properties with domain-specific extensions**



Standard IR format and exchange files

Novel domain-specific extensions (format)

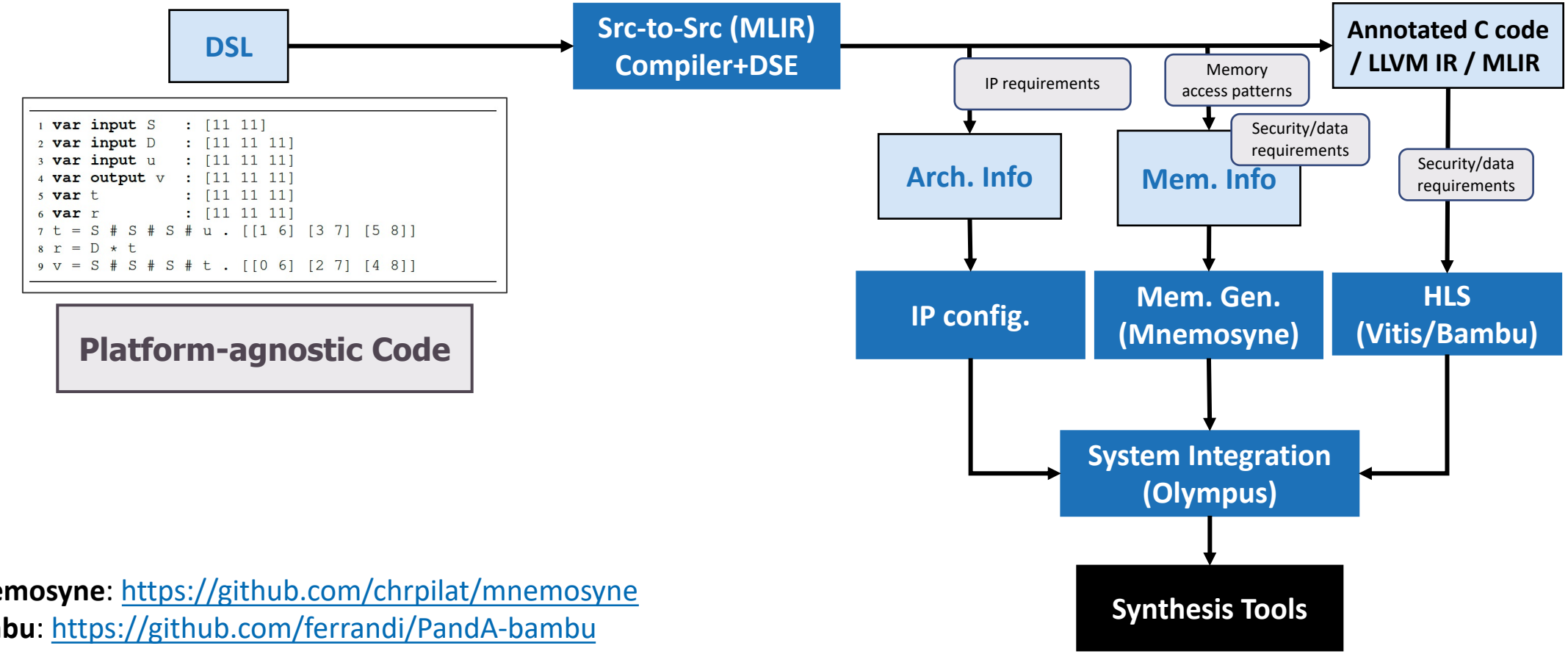
System and resource description (format)

Possibility of using different (ML) frameworks

Interoperability with different HLS tools

Hardware Compilation Flow

Automated **DSL-to-bitstream generation** for accelerating selected application kernels with specialized memory architectures

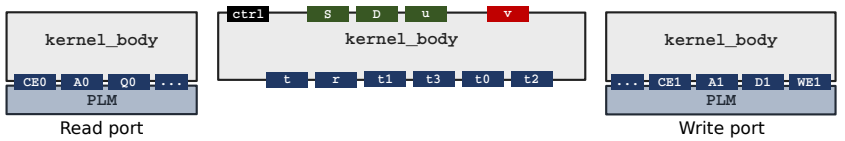


Mnemosyne: https://github.com/chrpilat/mnemosyne
Bambu: https://github.com/ferrandi/PandA-bambu

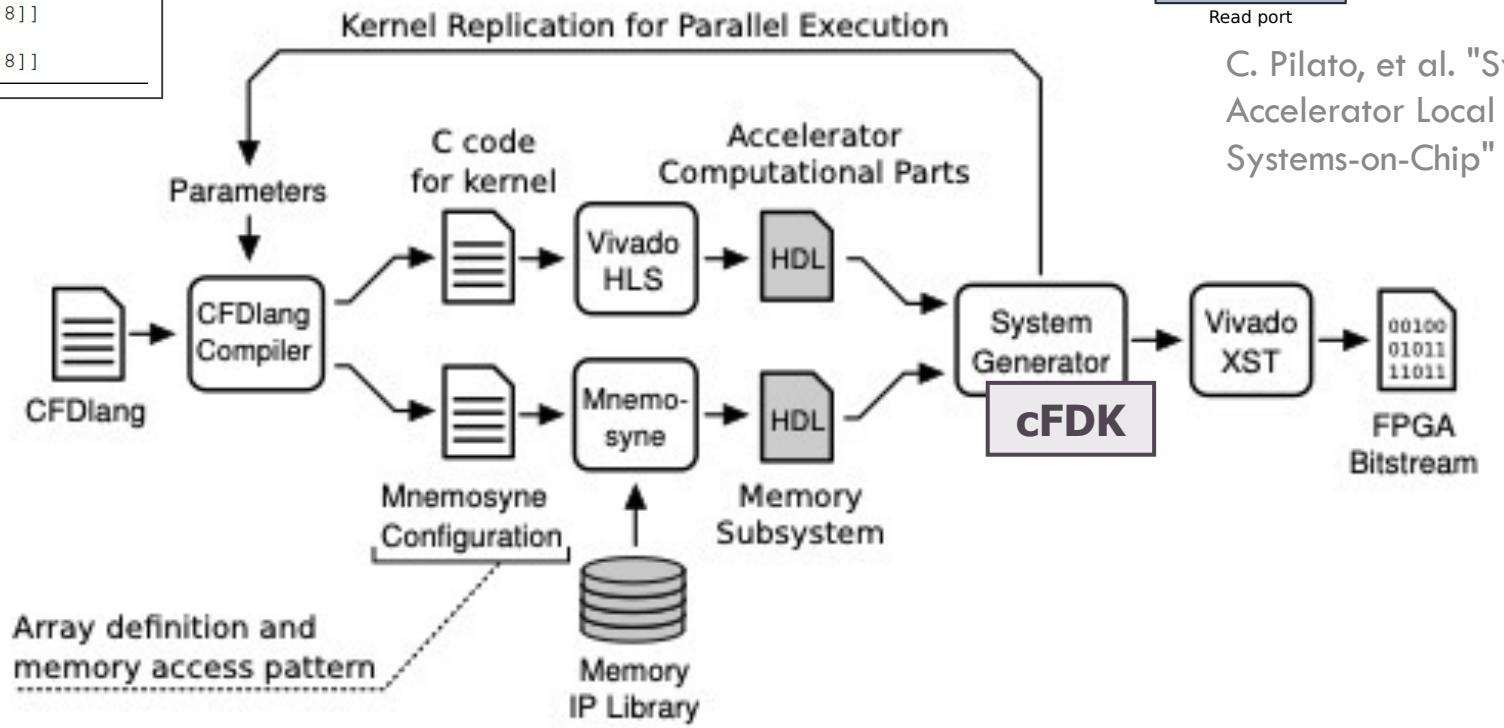
From DSL to Bitstream

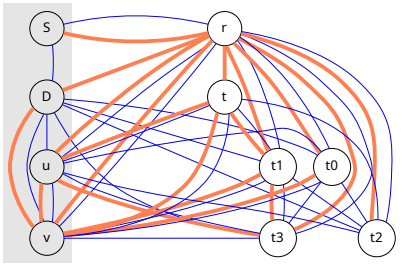
```
var input S    : [11 11]
var input D    : [11 11 11]
var input u    : [11 11 11]
var output v   : [11 11 11]
var t          : [11 11 11]
var r          : [11 11 11]
t = S # S # S # u . [[1 6] [3 7] [5 8]]
r = D * t
v = S # S # S # t . [[0 6] [2 7] [4 8]]
```

```
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])
```

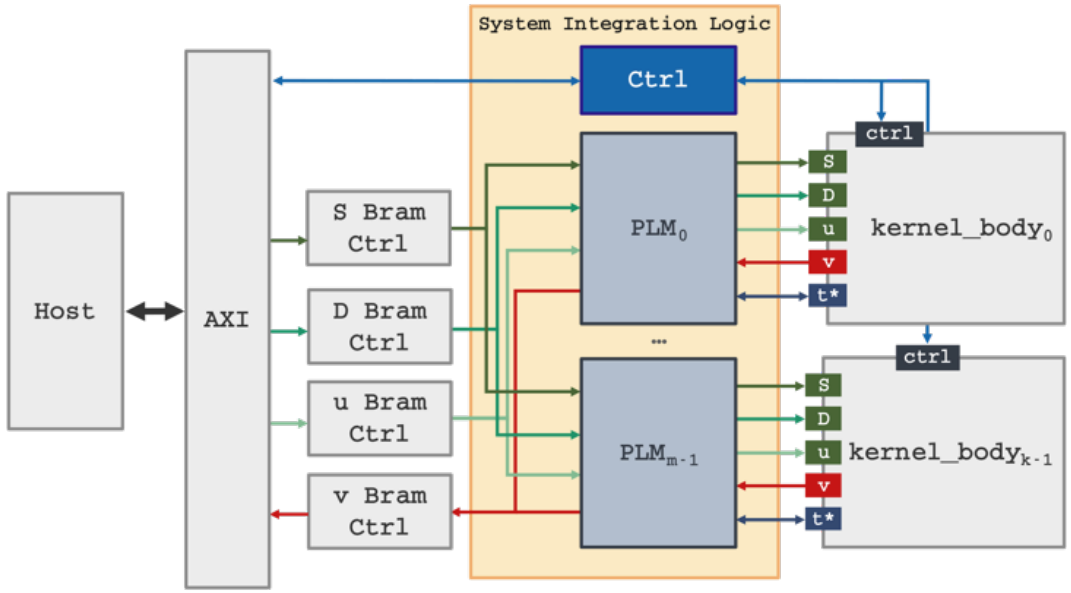


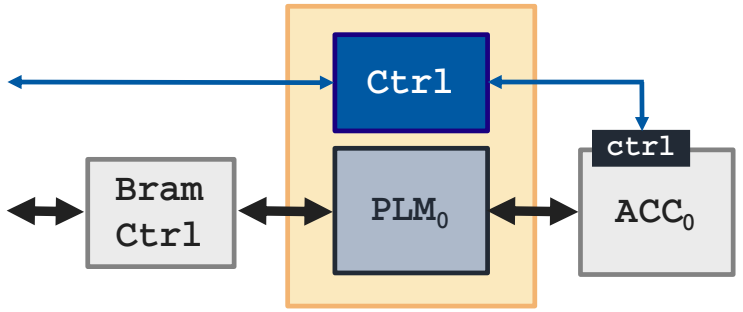
C. Pilato, et al. "System-Level Optimization of Accelerator Local Memory for Heterogeneous Systems-on-Chip" TCAD'17

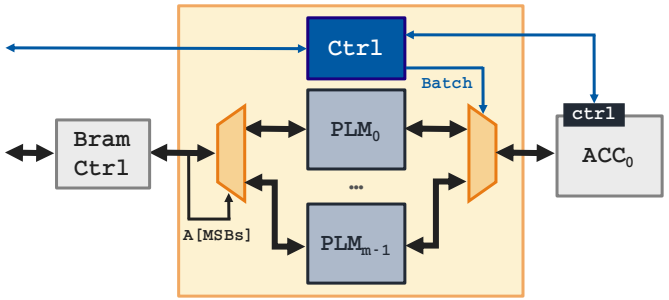


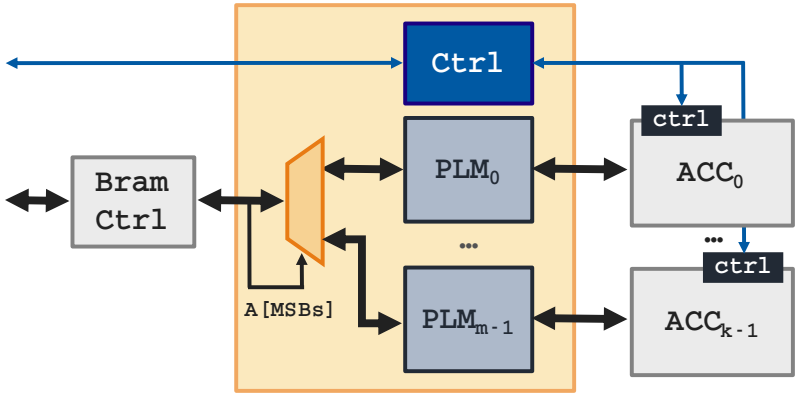


Creation of Parallel Architectures



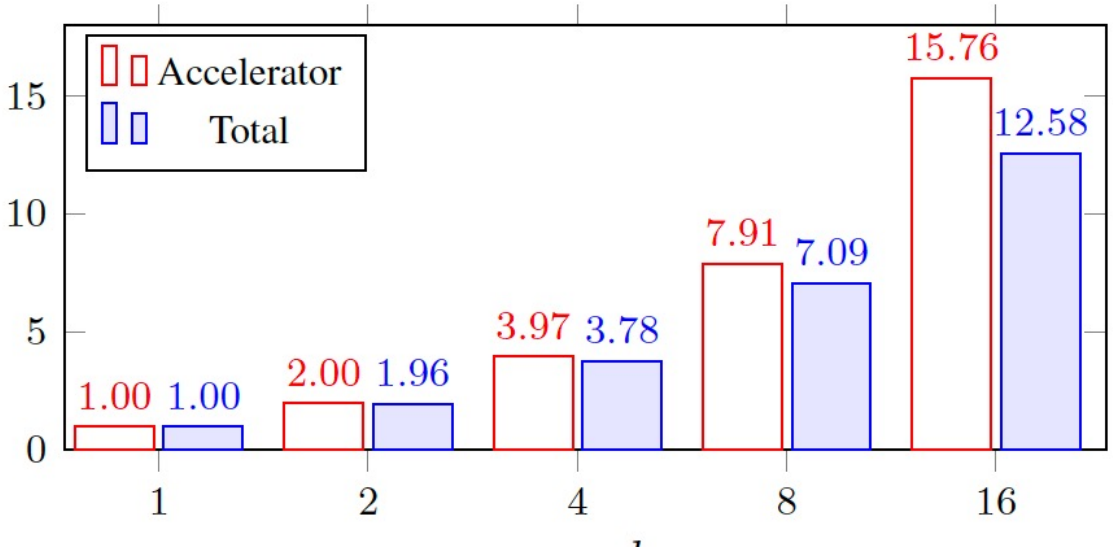


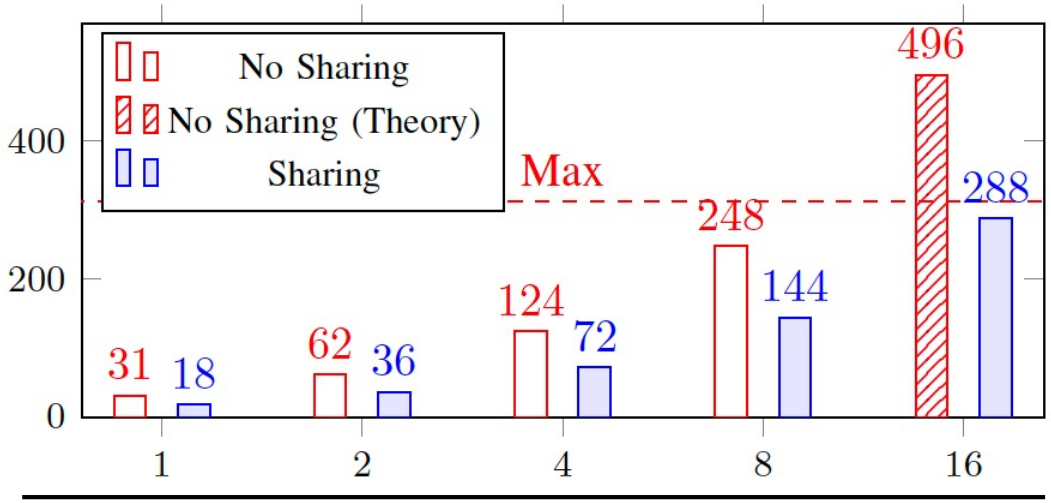




Preliminary Evaluation

- Xilinx Zynq UltraScale+ MPSoC ZCU106 board
  - CFD simulation of 50,000 elements
- Preliminary comparison with embedded ARM





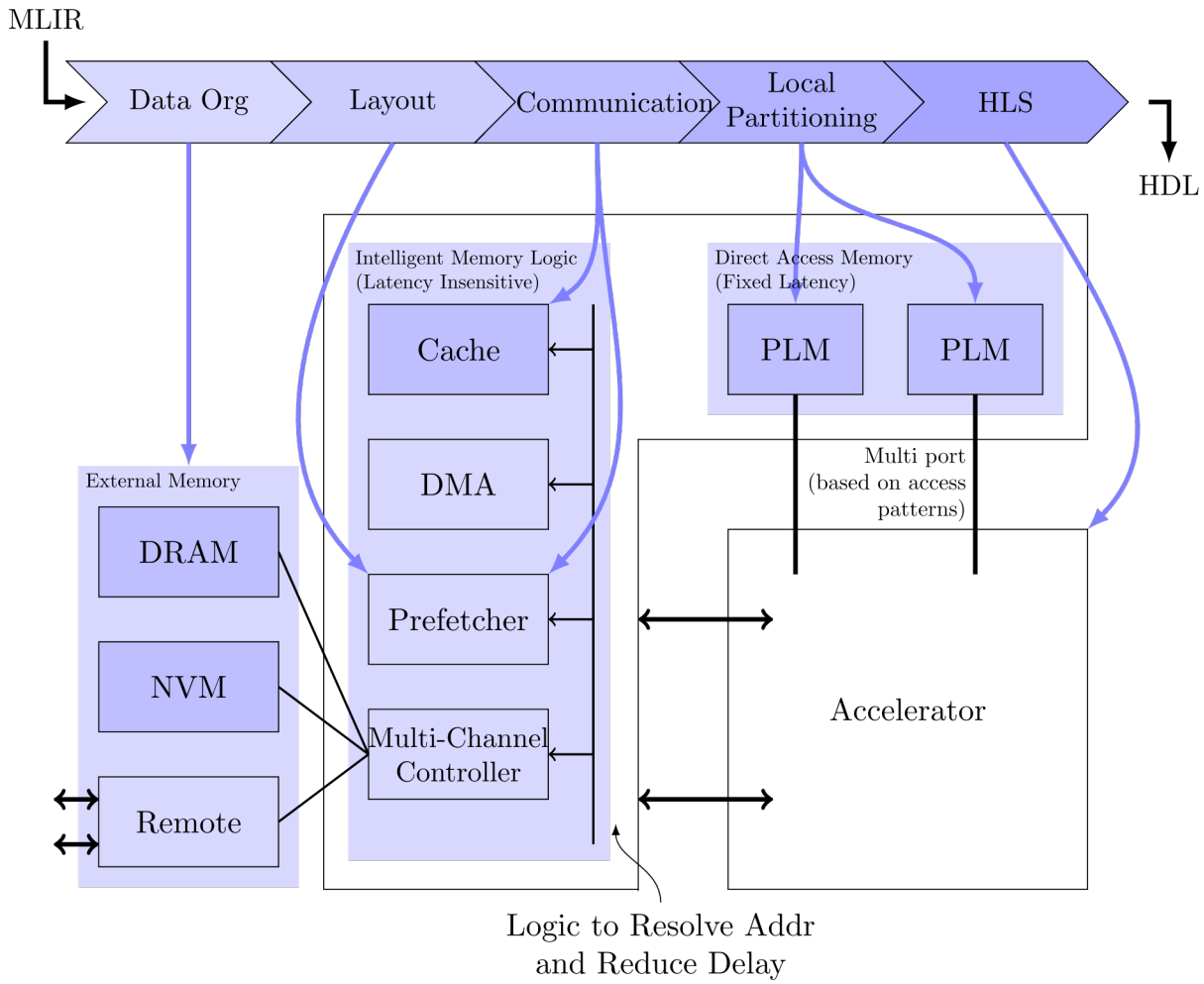
	m, k	LUT		FF		DSP	
No Sharing	1	11,318	(4.9%)	9,523	(2.1%)	15	(0.9%)
	2	15,929	(6.9%)	12,583	(2.7%)	30	(1.7%)
	4	25,728	(11.2%)	18,663	(4.1%)	60	(3.5%)
	8	42,679	(18.5%)	30,795	(6.7%)	120	(6.9%)
Sharing	1	11,292	(4.9%)	9,533	(2.1%)	15	(0.9%)
	2	15,572	(6.8%)	12,596	(2.7%)	30	(1.7%)
	4	24,480	(10.6%)	18,663	(4.1%)	60	(3.5%)
	8	42,141	(18.3%)	30,782	(6.7%)	120	(6.9%)
	16	77,235	(33.5%)	55,053	(12.0%)	240	(13.9%)

K. Friebel, et al. "From Domain-Specific Languages to Memory-Optimized Accelerators for Fluid Dynamics" HPCFPGA'21

Next Step: Let's Put Memory First

We are building a **compilation flow** based on **LLVM MLIR** for **automatic specialization**:

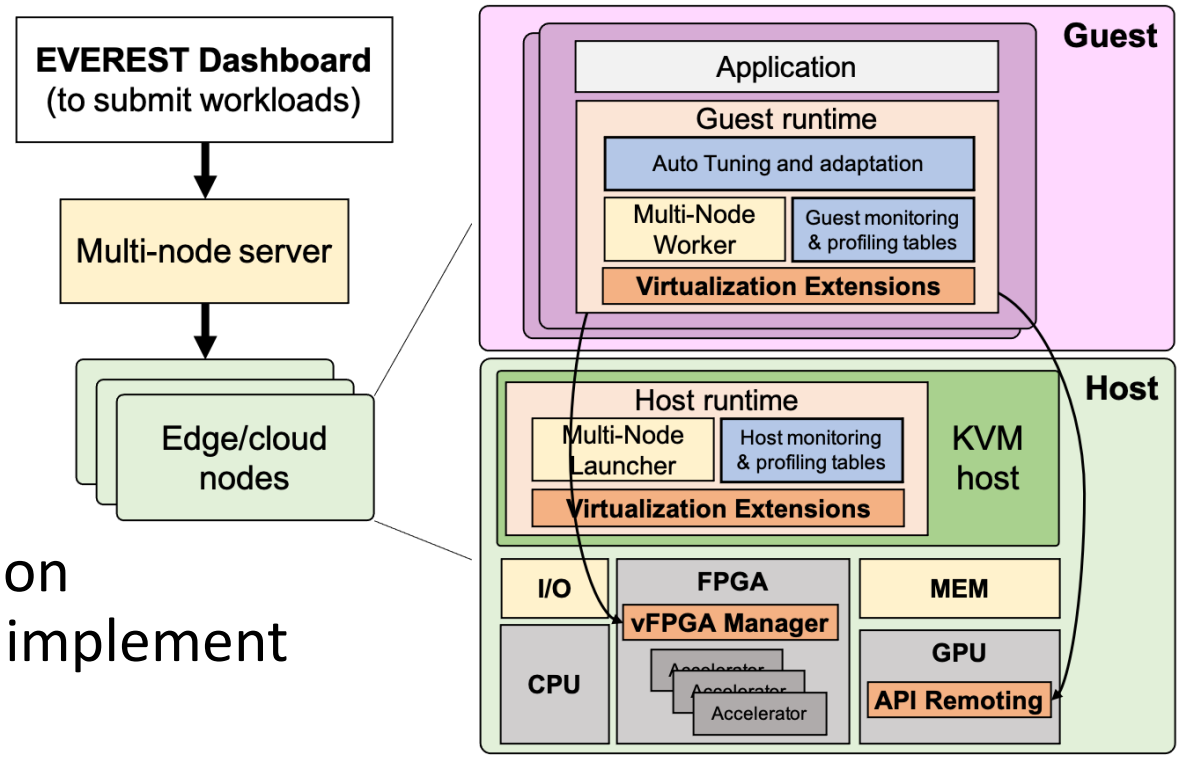
- **MLIR Input** – From DSL descriptions of the system functionality
- **Data Organization** – Determine which data resides off chip (also based on user/compiler annotations)
- **Layout** – Reorganize communication to exploit local memories (cache/PLM)
- **Communication** – Configure prefetcher to hide transfer latency
- **Local Partitioning** – Determine multi-bank PLM architecture (Mnemosyne[1])
- **HLS** – Generate computation part (interfacing with existing HLS tools, e.g., open-source Vitis HLS frontend)
- **HDL Output** – Automated code generation and system-level integration based on the target platform



S. Soldavini and C. Pilato. "Compiler Infrastructure for Specializing Domain-Specific Memory Templates" LATTE'21

EVEREST Runtime Environment

2. **Runtime Environment:** implements the *selection of "variants"* and the *hardware configuration* based on the *system status*
   - **Dynamic adaptation and autotuning** (**mARGOt**)
   - **Two-level runtime** for (1) **virtualization** of hardware resources regardless their distribution and the low-level details of the platforms; (2) implement **functional decisions** (**mARGOt**, **HyperLoom**)



How to collect system status and expose it to the runtime?

Runtime API

Autotuning API

Hiding communication latency (e.g., prefetching)

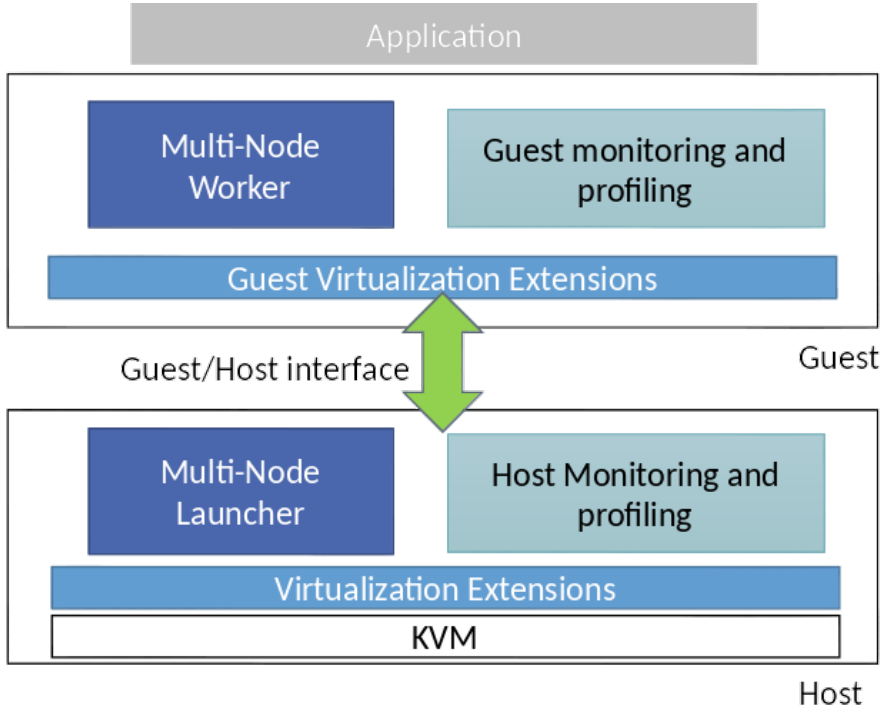
Seamless execution when varying the system configuration (resources, nodes, data, etc.)

EVEREST Runtime Virtualization

The **EVEREST runtime virtualization** is based on specific host and guest extensions:

- **Host**
  - The hypervisor of reference is KVM, and the Virtualized Infrastructure Manager is OpenStack
  - Libvirt will be extended to recognize harwdare accelerators and allocate them to the VMs. It will be used as driver for OpenStack Nova

- **Guests**
  - Will be extended with multi-node and monitoring/profiling applications that communicate with the host
  - Guests will benefit from optimized communication mechanisms based on shared memory

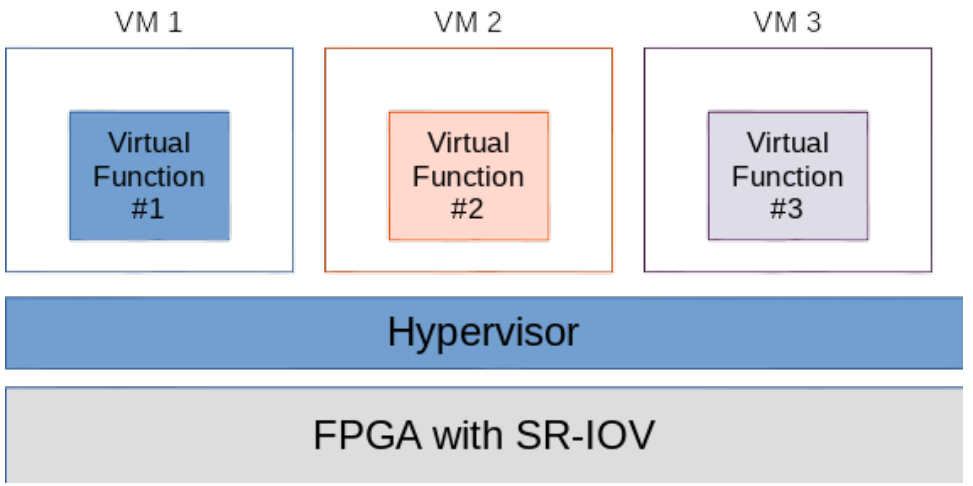


EVEREST Runtime Virtualization

The EVEREST runtime **key virtualization techniques** will be **FPGA Virtualization** and **API remoting**:

FPGA Virtualization: we consider both network-based and host-attached approaches

- **Network based**: using cloudFPGA mentioned before
- **Host attached**: SR-IOV and mediated pass-through used to allow guests direct access to the FPGA partitions
  - Single Root Input-Output Virtualization (SR-IOV) enables the creation of multiple virtual functions
  - Each of them will be mapped to a VM (and a portion of the FPGA)

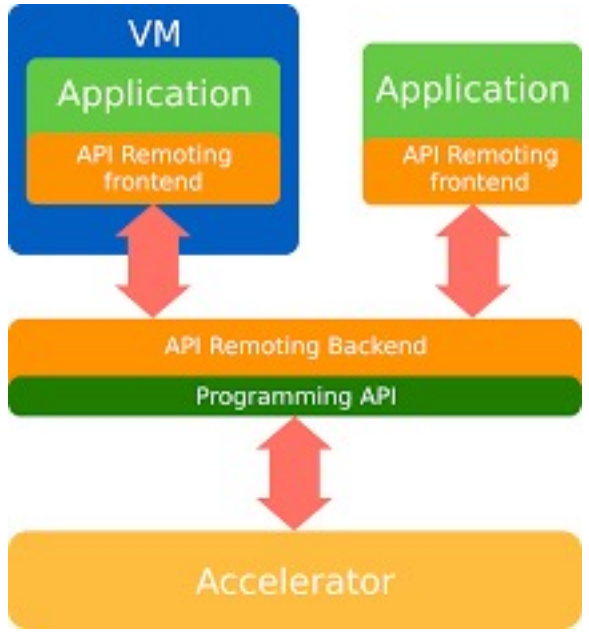


EVEREST Runtime Virtualization (ii)

The EVEREST runtime **key virtualization techniques** will be **FPGA Virtualization** and **API remoting**:

API remoting

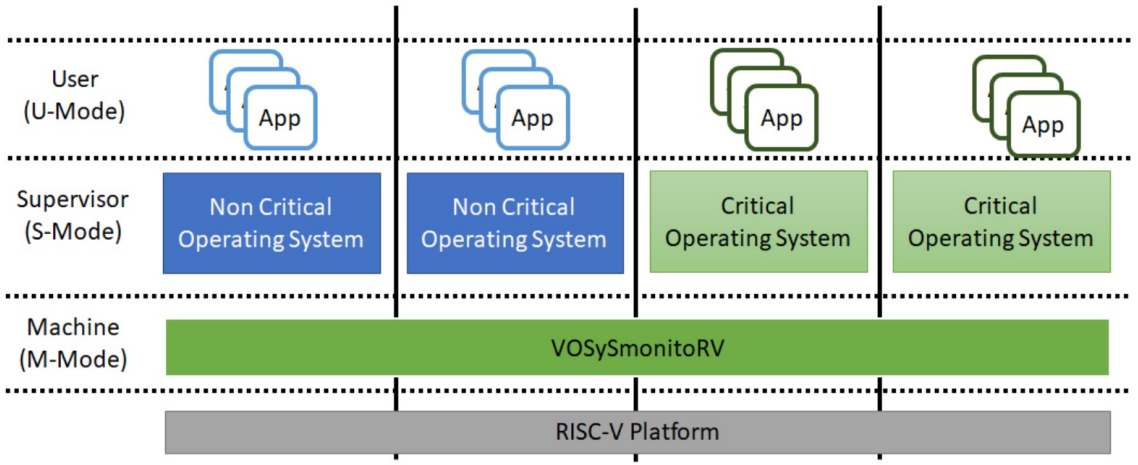
- Virtualization at a library level will be implemented through API remoting
- API calls in the guests will be redirected to the host
- A zero-copy shared memory mechanism is in place between the host and guests OSes



EVEREST Runtime Virtualization at the edge

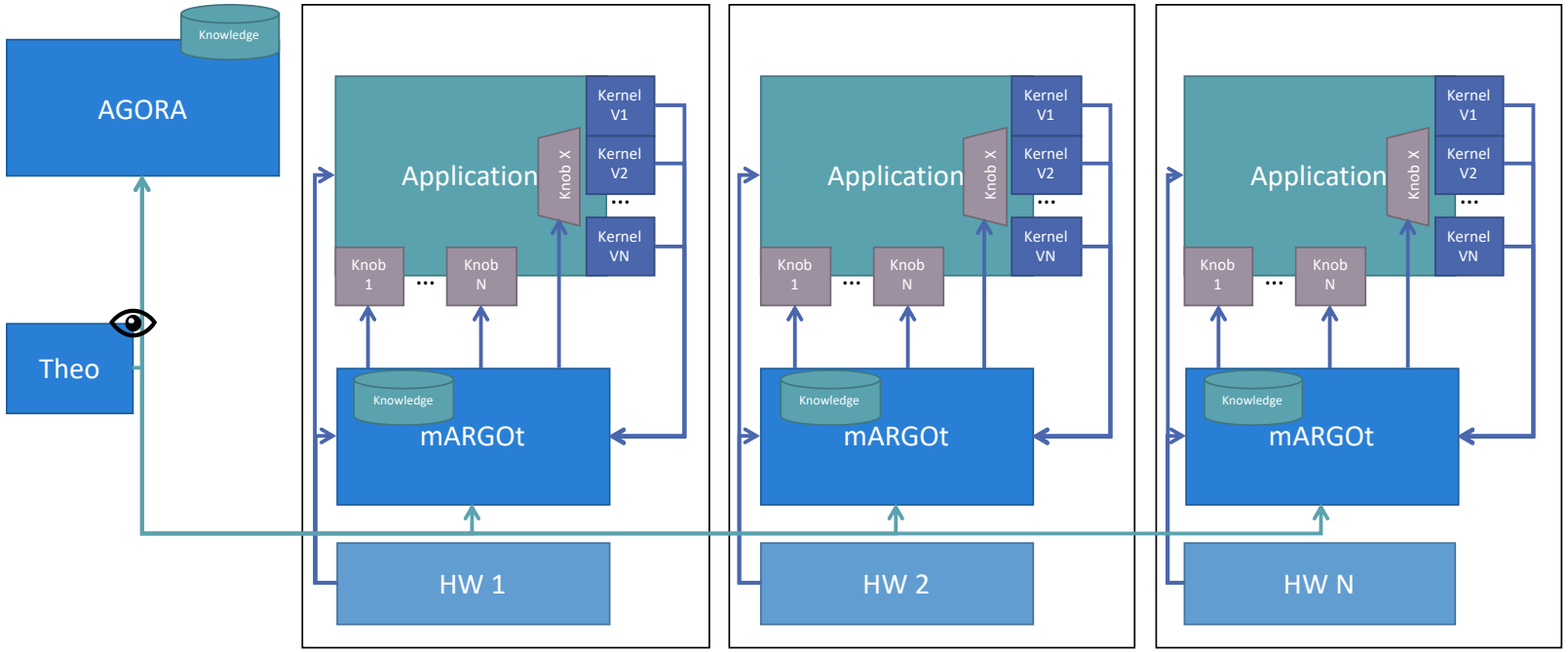
The EVEREST runtime key virtualization focuses on **both edge and cloud sides**:

- Arm, RISC-V, and x86 supported for the cloud side
- Specific virtualization solutions based on system resource partitioning are developed to cope with security and safety requirements of edge systems
- **VOSySmonitor** and **VOSySmonitoRV** by Virtual Open Systems are based respectively on Arm TrustZone and RISC-V PMP zones



EVEREST Runtime functional decision

The EVEREST FPGA systems include a **monitoring and decision infrastructure** for **dynamic autotuning** based on workload conditions



- **Application variants** (either software or hardware) are generated at design time (compilation and hardware synthesis), and selected at run time based on the actual available hardware resources

Conclusions

EVEREST is a new H2020 project that aims at **simplyfing the use of FPGA** for the acceleration of Big Data applications

- **Data-centric approach** focusing on **domain-specific extensions**, **high-level synthesis**, and **dynamic adaptivity**
- **Three application use cases**: *weather-based renewable energy prediction*, *air-quality monitoring of industrial sites*, and *intelligent traffic management*
- **CloudFPGA** will be used as primary target
- Looking for **interoperability** with existing solutions
- **EVEREST SDK** will be released as open-source to the community

The main goal is to **simplify the description of complex Big Data applications** and **improve the programmability of distributed FPGA-based systems**

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