



The Abdus Salam
International Centre
for Theoretical Physics



1st Mesoamerican Workshop on Reconfigurable X-ray Scientific Instrumentation for Cultural Heritage

HyperFPGA heterogeneous computing on MPSoC-FPGA at ICTP

Antigua Guatemala, June 2025

Maynor Ballina

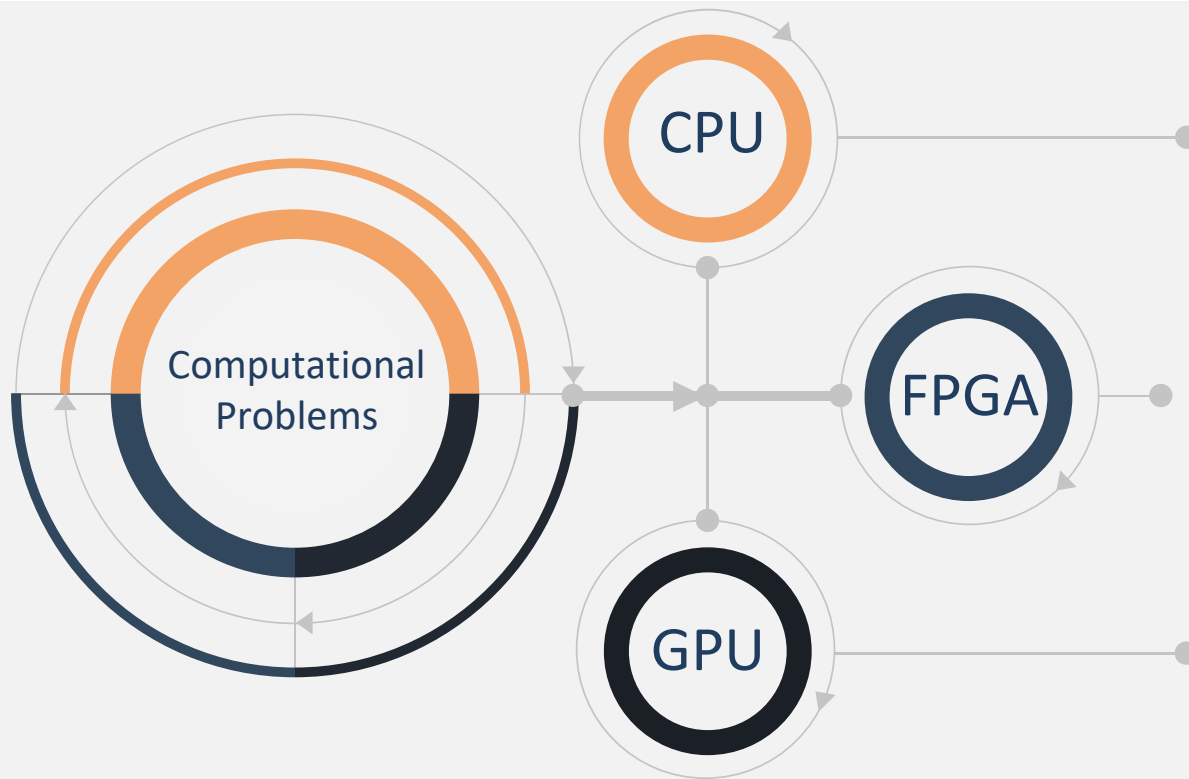


Introduction



- Heterogeneous computing
- Evolution of heterogeneous computing
- Distributed Computing
- Flynn Taxonomy
- Application of Heterogeneous Computing in Simulation
- The HyperFPGA cluster
- Infrastructure and management
- User Interface
- Jupyter Notebooks
- Broad spectrum of computational tasks
- Education and collaborative projects

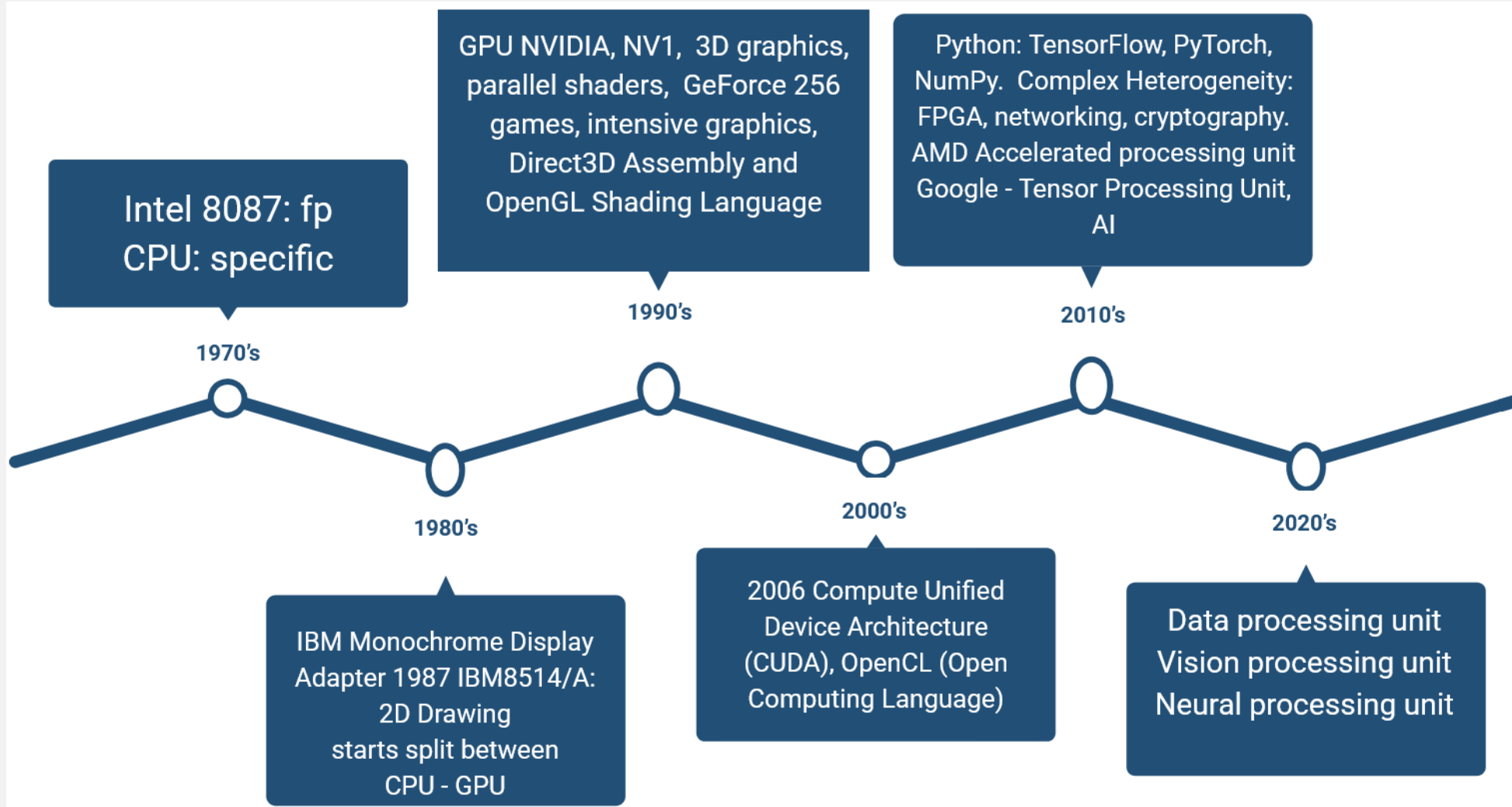
Heterogeneous Computing



It enhances computational performance by accelerating workloads such as graphics processing, artificial intelligence, and scientific simulations through specialized hardware.

The coordinated use of different types of processing cores within a single system to maximize performance and energy efficiency.

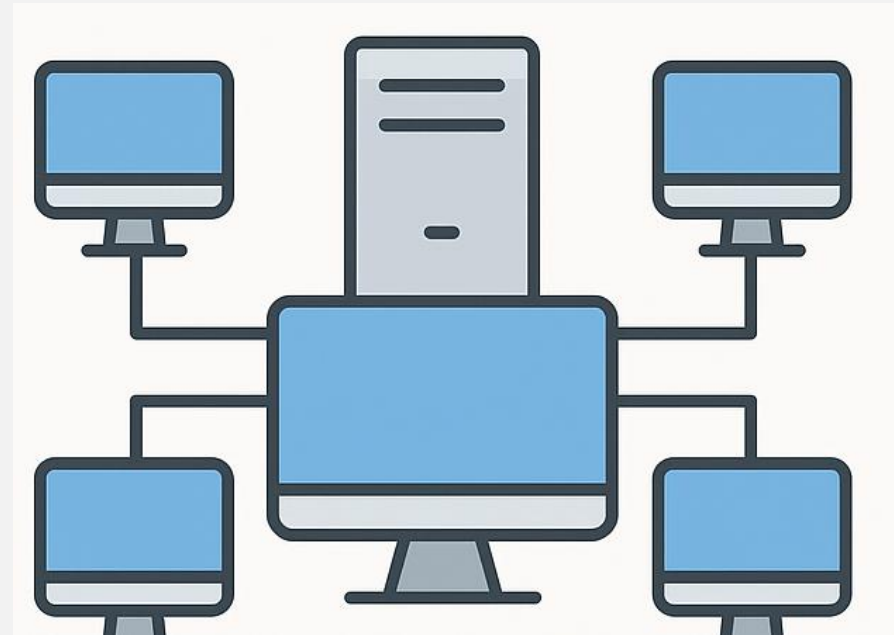
Evolution of Heterogeneous Computing



Distributed Computing

A model in which multiple nodes work together in a coordinated manner to solve a problem, sharing resources to improve performance, scalability, and fault tolerance.

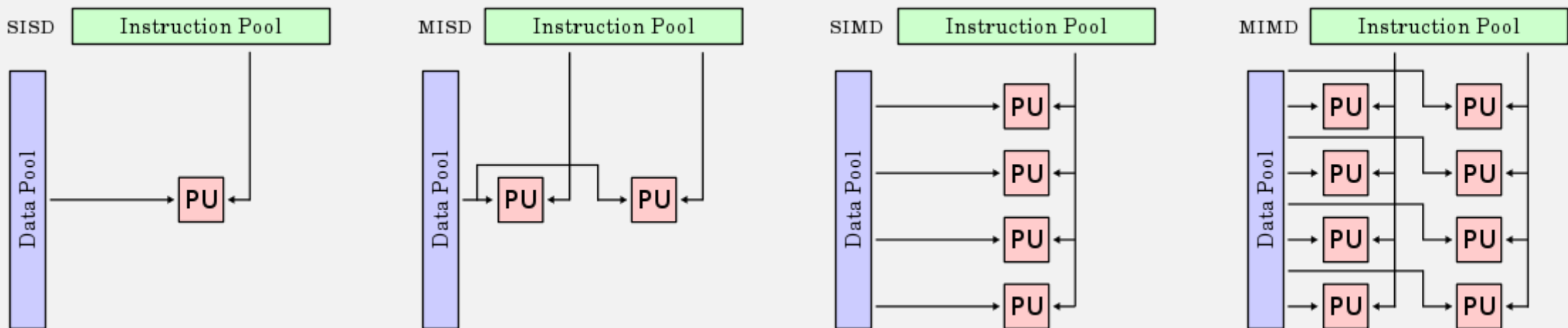
Distributed computing in clusters is a data processing approach that uses a set of interconnected nodes to work together as if they were a single machine.



Flynn's taxonomy

Classifies computer architectures based on the number of instruction streams and data streams.

- **SISD**: Single Instruction, Single Data – Conventional CPU
- **SIMD**: Single Instruction, Multiple Data – GPUs, instruction set extensions in CPUs
- **MISD**: Multiple Instructions, Single Data – Parallel processing, error detection
- **MIMD**: Multiple Instructions, Multiple Data – Multiprocessors, supercomputers



Application of Heterogeneous Computing in Simulation

Fluid Dynamics

CPU, GPU, FPGA



Molecular Dynamics

CPU, GPU, FPGA



Quantum Computational Simulation

CPU, GPU, TPU, FPGA



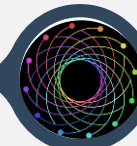
Cybersecurity

CPU, GPU, TPU, FPGA, DTU



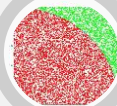
Cosmological Models and Gravity Simulations

CPU, GPU, FPGA (Barnes-Hut)



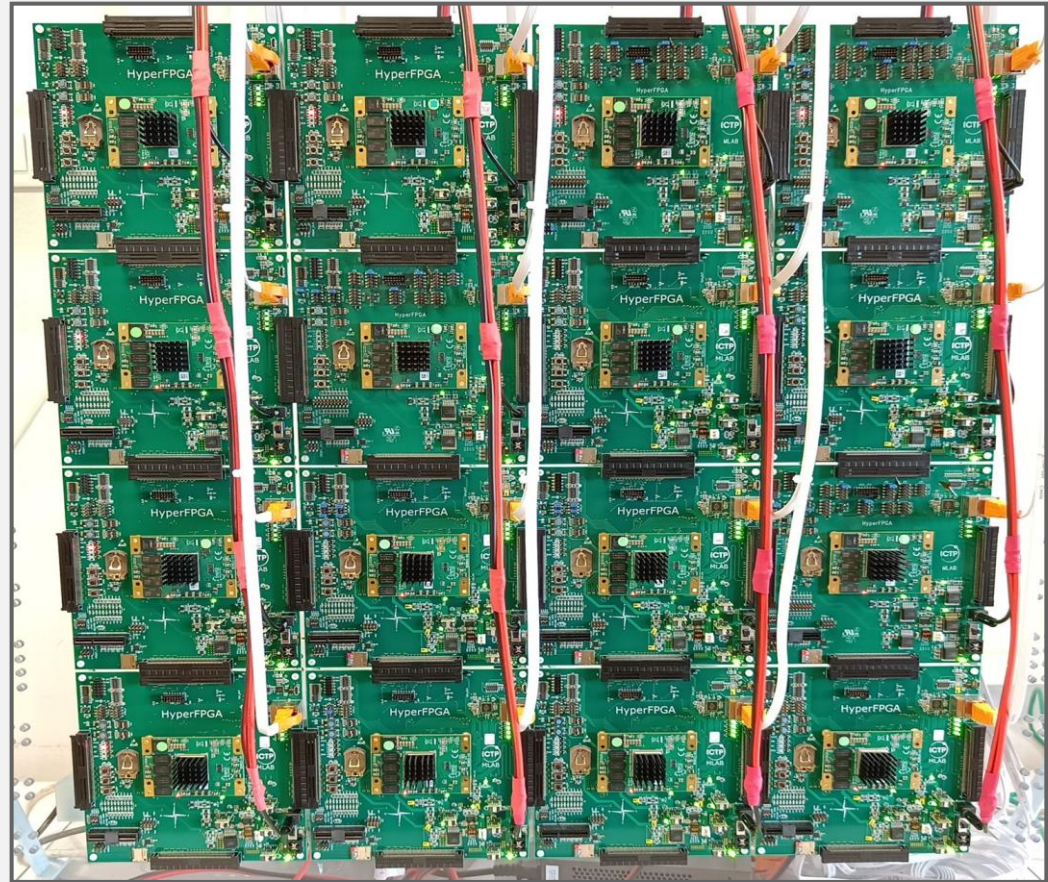
Montecarlo

CPU, GPU, FPGA

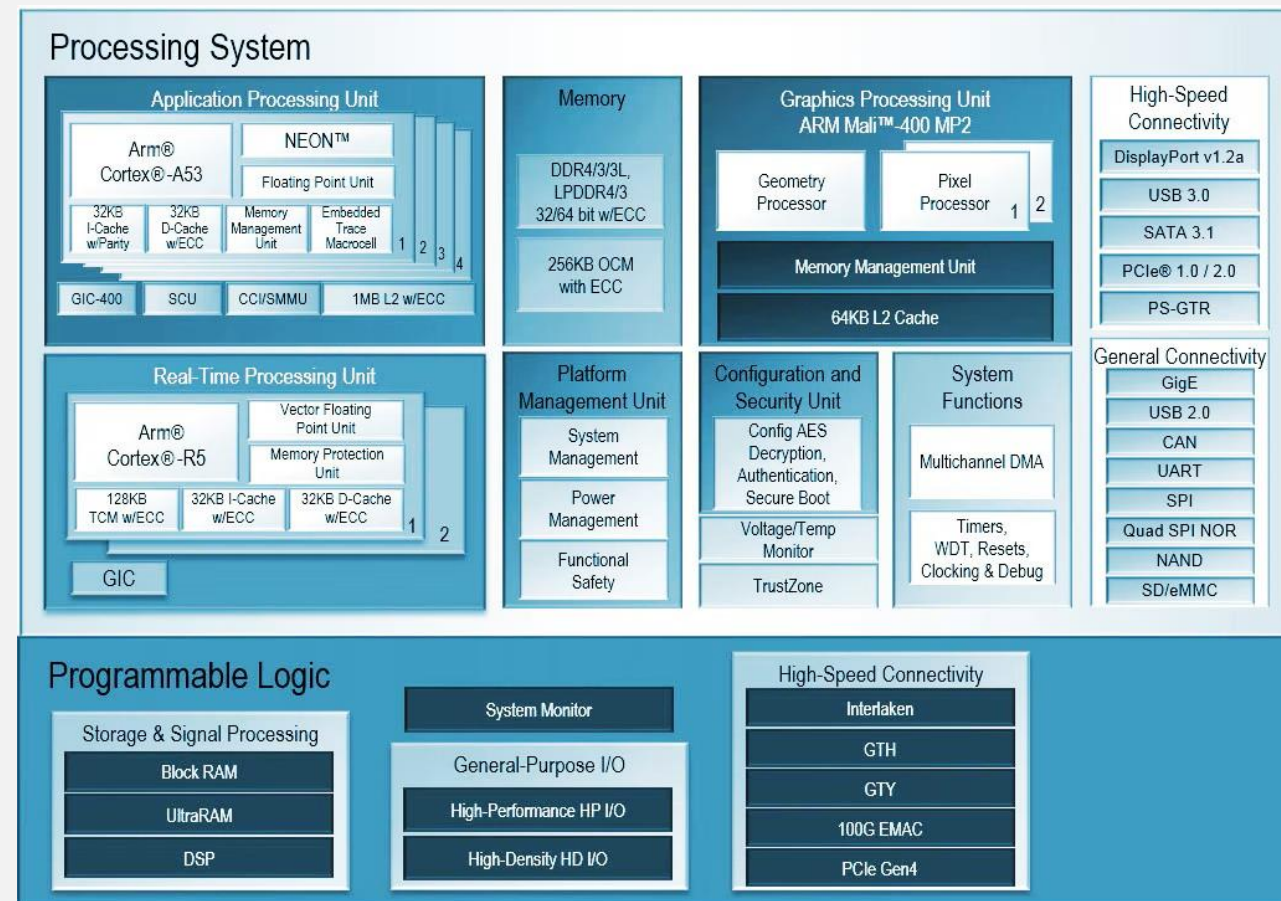
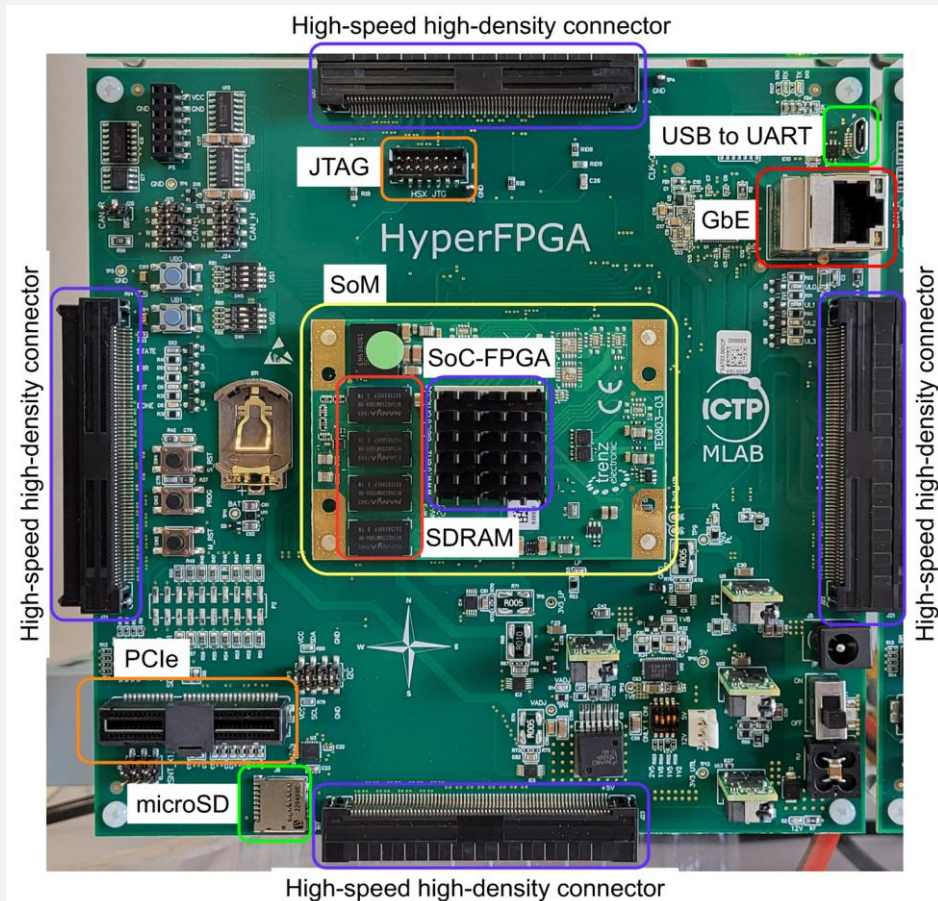


The HyperFPGA cluster

- MPSoC-FPGA-based experimental cluster.
- Zynq UltraScale+ MPSoC with APU, RPU, GPU.
- Fully connected, HP, HD, GTH, MIO, Ethernet.
- Debian OS

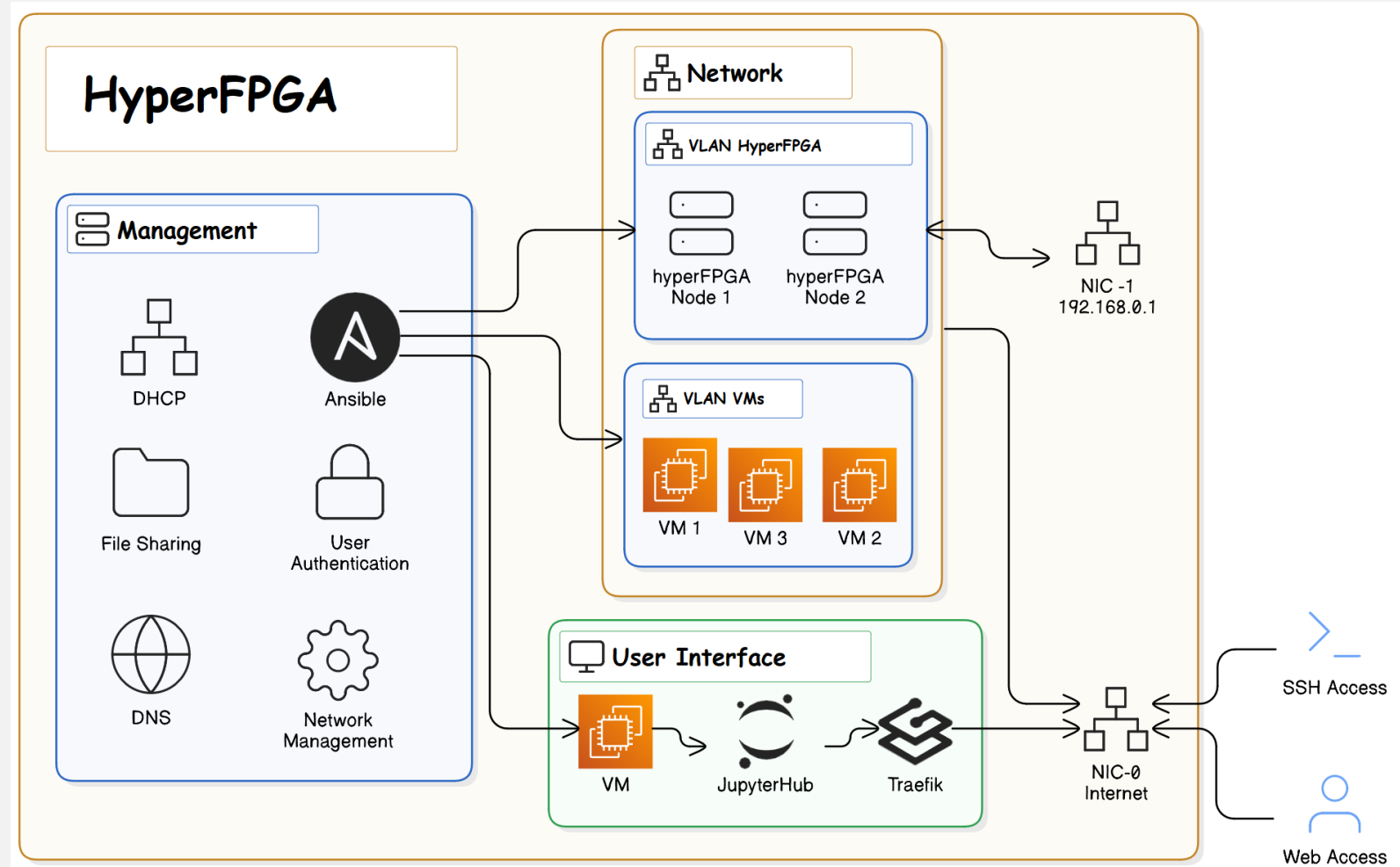


The HyperFPGA cluster



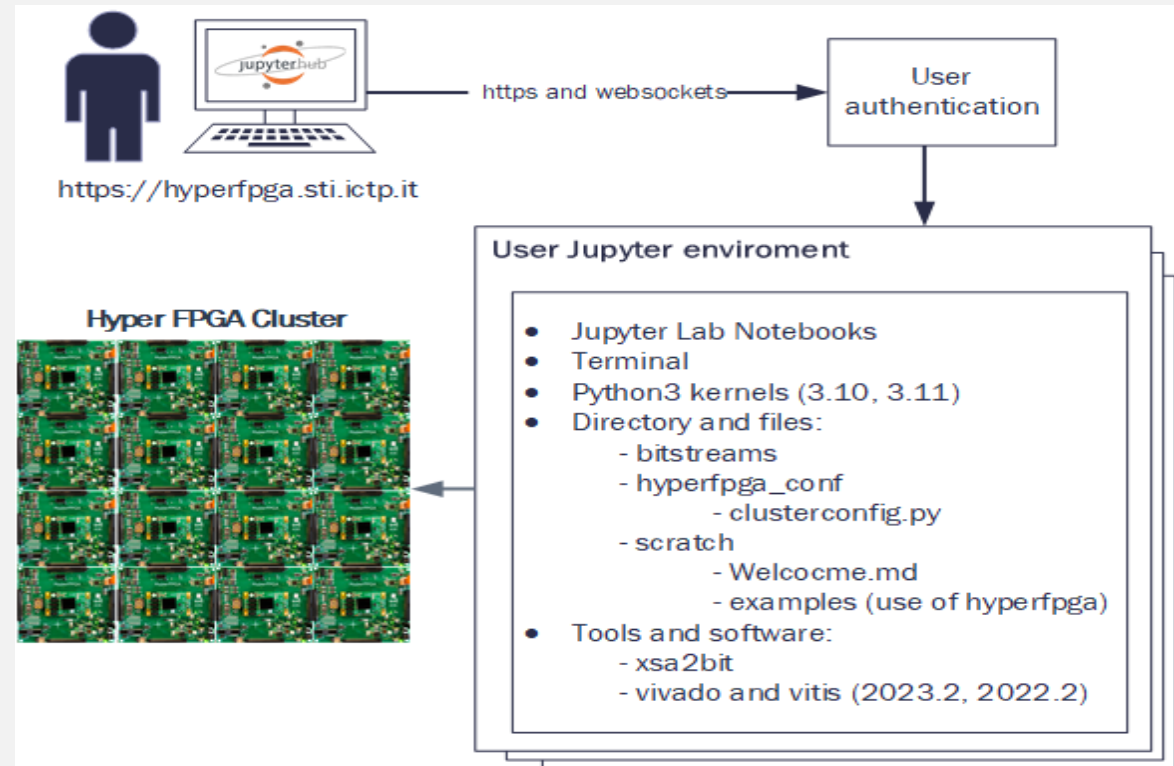
Infrastructure & Management

- Managed through IaC principles.
- Automated using Ansible for provisioning and updates.
- Flexible resource allocation: 1:1, 1:N, N:1 user-node configurations.



User Interface: JupyterHub

- Web-based, interactive platform.
- Roles: Admin (course content setup), Student (notebook interaction).
- Includes Vivado, Vitis, custom Python libraries for FPGA interaction.



Jupyter Notebooks

The screenshot shows the JupyterLab interface. On the left is a file explorer with a sidebar containing icons for home, search, and other functions. The main pane shows a directory structure with files: 'bitstreams' (2mo ago), 'hyperfpga_conf' (3mo ago), 'scratch' (3mo ago), and 'nodes.json' (2mo ago). The right pane shows a Jupyter notebook with a single code cell. The cell's title is '01-Tutorial_configuration.ipyn'. The code cell contains the following text:

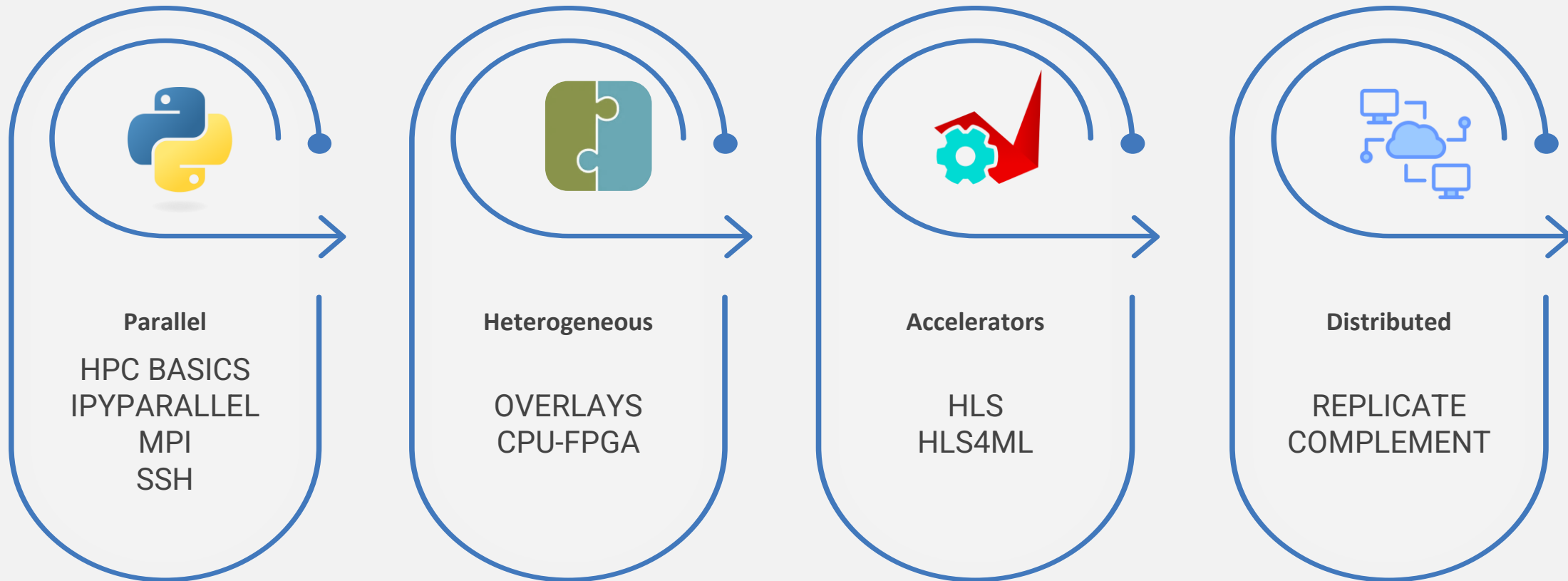
```
[3]: nodes = get_nodes()
```

Below the code cell, the output is displayed as a grid of 16 images, each showing a circuit board (FPGA) with a white label overlay. The labels are as follows:

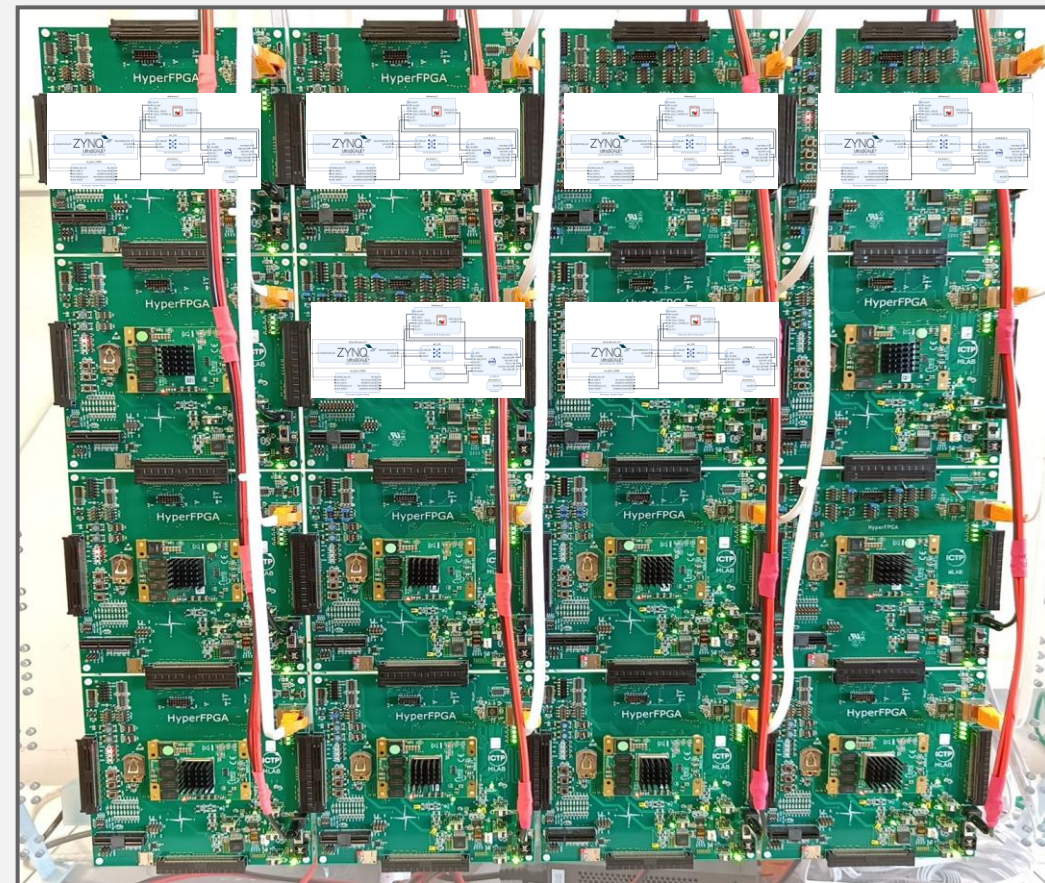
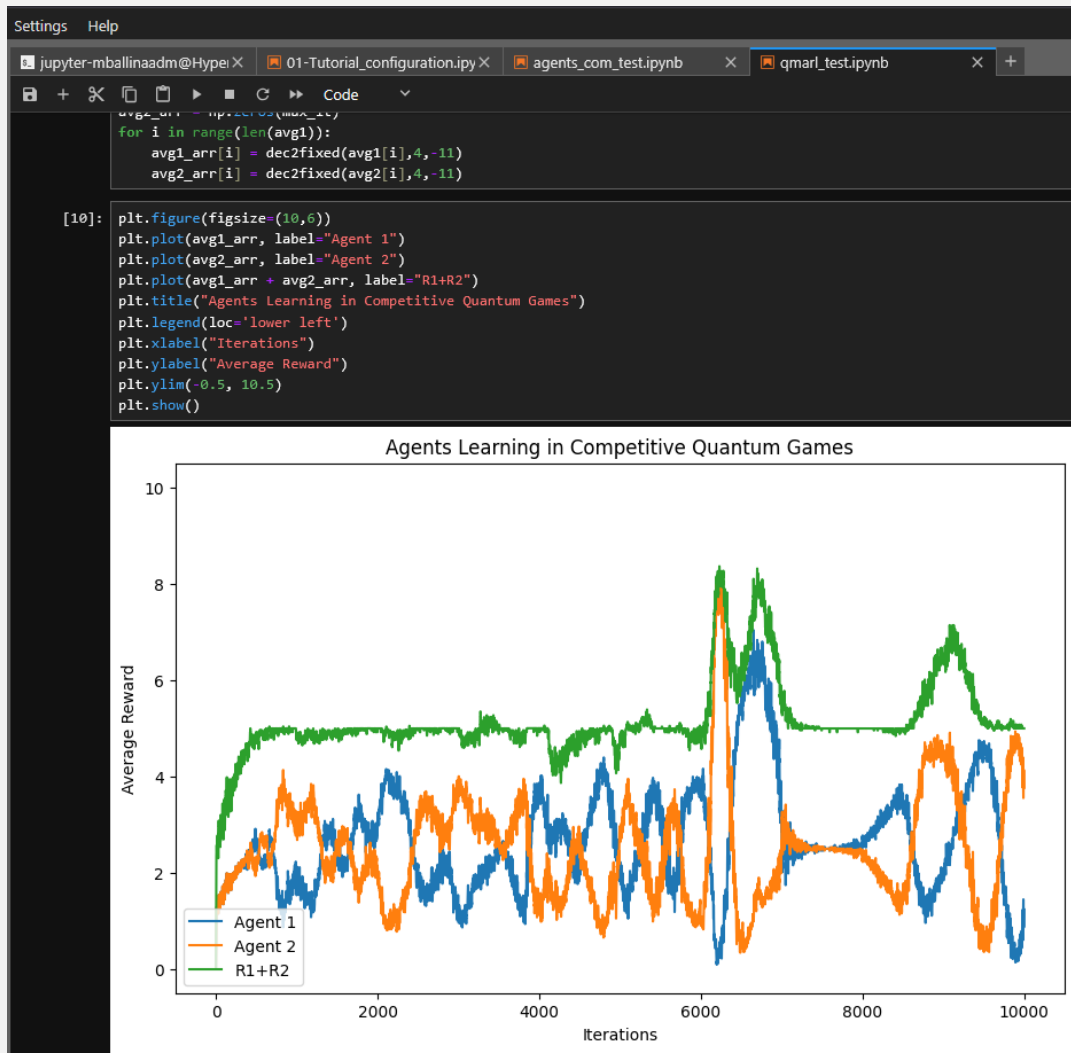
node	model	ip
node: 0	model: 4ge21	ip: 192.168.0.2
node: 1	model: 4ge21	ip: 192.168.0.3
node: 2	model: 4ge21	ip: 192.168.0.4
node: 3	model: 3be11	ip: 192.168.0.5
node: 4	model: 4ge21	ip: 192.168.0.6
node: 5	model: 4ge21	ip: 192.168.0.7
node: 6	model: 4ge21	ip: 192.168.0.8
node: 7	model: 3be11	ip: 192.168.0.9
node: 8	model: 3be11	ip: 192.168.0.10
node: 9	model: 3be11	ip: 192.168.0.11
node: 10	model: 3be11	ip: 192.168.0.12
node: 11	model: 3be11	ip: 192.168.0.13
node: 12	model: 3be11	ip: 192.168.0.14
node: 13	model: 3be11	ip: 192.168.0.15
node: 14	model: 3be11	ip: 192.168.0.16
node: 15	model: 3be11	ip: 192.168.0.17

The screenshot shows the Hyperfpga IDE interface. On the left is a file explorer with a list of files: bitstreams, hyperfpga_conf, scratch, and nodes.json. The main workspace displays a block design titled 'basic_test'. The design features two 'comblock' blocks, 'comblock_0' and 'comblock_1', which are interconnected with an 'UltraScale_and_AXI_interconnect'. The interconnect is also connected to an 'sconstant_O' block, a 'c_counter_binary_O' block, and a 'CLK' block. The comblock blocks are connected to an 'adder_subtractor' block and a 'Multiplier_32_bit' block. A legend indicates: magenta for Register LoopBack, yellow for FIFO LoopBack, green for add 2, 32bit numbers, and blue for multiply 2, 32 bit numbers.

Broad spectrum of computational tasks



Distributing computing examples



Education and collaborative projects

Distributed
computing,
DSP,
HPC
fundamentals

From Algorithm
to Hardware:
Machine
Learning in
Embedded
Systems



Distributed
computing,
Heterogeneous
computing,

Distributed
computing

Used in 2 pilot courses with over 50 students across two countries and 3 different collaborative projects.

Conclusion:

Why is heterogeneous computing necessary?

Performance Optimization: GPUs excel at massive parallel processing, while CPUs are better suited for complex sequential and control tasks. FPGAs allow for specific customization of critical tasks and real-time optimizations.

Energy Efficiency: Some accelerators, such as FPGAs, are more energy-efficient than CPUs or GPUs, which is crucial for simulations requiring extensive computational resources over long periods.

Scalability: Accelerators enable simulations to run efficiently at larger scales. Without heterogeneous computing, some simulations requiring real-time data processing or massive scale would be unfeasible.

Latency Reduction: In certain problems, such as those needing real-time simulations or immediate feedback, FPGAs and other accelerators can significantly reduce system latency.

Questions

Thank you for your attention!

Maynor Ballina
mballina@ictp.it

