# Bringing Hardware Acceleration closer to the programmer

*Iakovos Mavroidis*

*Telecommunication Systems Institute*
*iakovosmavro@gmail.com*

*EcoScale-ExaNest joint workshop*
*Rome, 17 February 2017*

*ECOSCALE has received funding from the EU Union's Horizon 2020 research and innovation programme under the grant agreement No 671632*

## Outline

- Introduction : HPC Systems
  - Exascale challenges
  - Approach: Accelerators (GPUs vs FPGAs)
- FPGA in HPC
  - Programming Model
  - Limitations & Solutions (ECOSCALE)
    - UNIMEM/UNILOGIC architecture
    - Reconfiguration toolset
    - Runtime System
    - Execution Environment
- Conclusion

# Top HPC Servers Today

- **SunWay TaihuLight** (Chinese)
  - 10M cores, 93 PFLOPS, 15 MW
    - Performance: 10 GFLOPS/core
    - Efficiency: 6.6 GFLOPS/W, 1.5 W/core
- **Tianhe-2 (Xeon E5, Phi)**
  - 3M cores, 33 PFLOPS, 17MW
    - Performance: 10 GFLOPS/core
    - Efficiency: 1.96 GFLOPS/W, 5.1W/core
- **Titan (Opteron, NVIDIA K20)**
  - 0.5M cores, 17 PFLOPS, 8 MW
    - Performance: 34 GFLOPS/core
    - Efficiency: 2.13 GFLOPS/W, 16W/core

SunWay TaihuLight

Tianhe-2

# Problem: Just scaling not a viable solution to reach ExaFlop

- **Energy Efficiency**
  - 2-6 GFLOPS/W ⇨ 1EFLOPS/0.15-0.5 GW

- **Cost & Space**
  - 10-30 GFLOP/ core ⇨ 1 EFLOP / 30-100 Mcores
  - 10-50x more space and cost

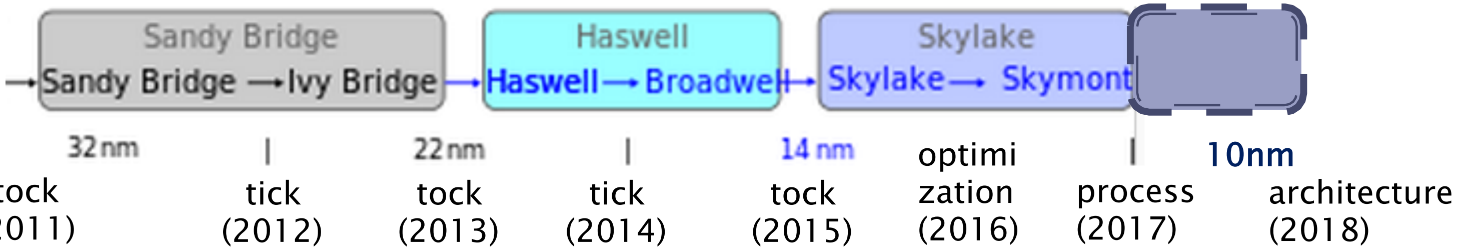
- **Resiliency**
  - projected MTBF less than 1 hour

- **Scalability & Management**
  - Manage 30-100 Mcores

## HPC Approach: Improve Technology, Architecture, Software

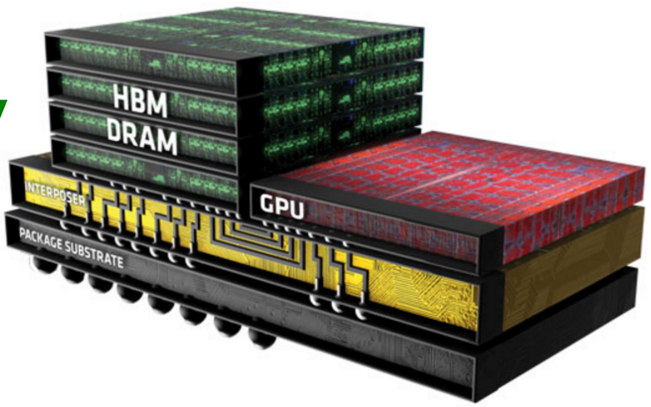
- ✓ **Technology**
  - Transistor shrinking
  - Bring transistors closer: 3D technology
- ✓ **Architecture**
  - Reduce data movements
  - Multi-level memory: scratchpad, cache, Flash
  - Increase parallelism
- ✓ **Software**
  - Programming Languages
  - System Software
  - Application

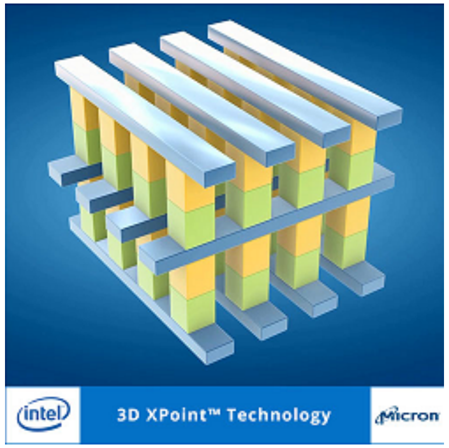
# Technology (in short)



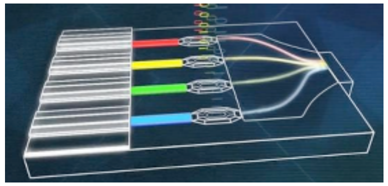
- Intel Announcement (March 2016): New technology **every 3 years instead of 2 years**

- **3D technology**
  - Interposer
  - Stacked





- **Silicon photonics**

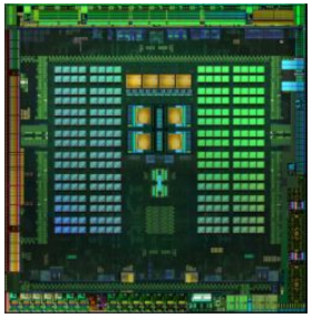


- **Flash memories**

# Common Architecture Approach: Use of Accelerators

- Accelerators are based on
  - ✓ parallel processing
  - ✓ synchronized accesses/processing
  - ✓ locality
- **Many-core**
  - Intel Xeon-Phi
  - KALRAY MPPA
- **GPUs**
  - NVIDIA
  - AMD
  - ARM
- **FPGAs**
  - Altera
  - Xilinx

Xeon-Phi

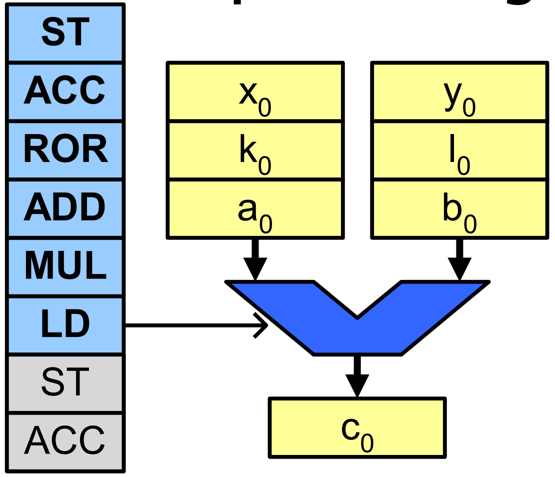


Tegra X1

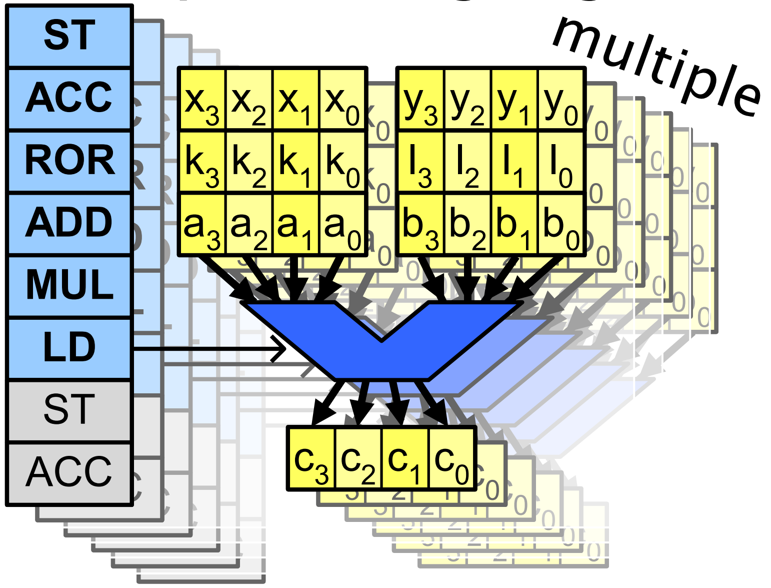
Ultrascale Board

# CPUs vs. GPUs vs. FPGAs

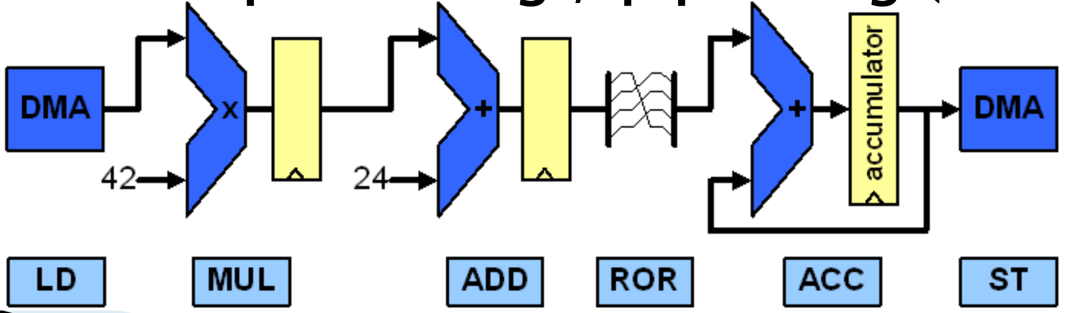
## Scalar processing



## SIMD processing (e.g. GPU)



## Dataflow processing / pipelining (FPGAs)



## Energy-Efficiency

Indicative numbers from various sources

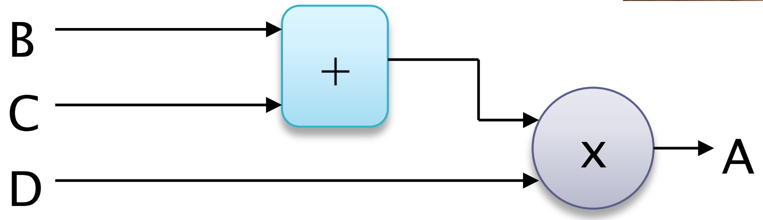
| Type | Device | GFLOPS (SP) | Cost (€) | Power (W) | GFLOPS/ € | GFLOPS/ W |
|---|---|---|---|---|---|---|
| Multi-core | Intel E5-2630v3 8x2.4GHz | 600 | 700 | 85 | 0.85 | 7.05 |
| | Intel E5-2630v3 10x2.3GHz | 740 | 1250 | 105 | 0.59 | 7.04 |
| Many-core | Xeon Phi, knights corner, 16GB | 2416 | 3500 | 270 | 0.69 | 8.94 |
| | Xeon Phi, knights landing, 16GB | 7000 | 3500 | 300 | 2.00 | 23.3 |
| GPU | Nvidia GeForce Titan X | 7000 | 1000 | 250 | 7.00 | 28 |
| | Nvidia Tesla K80 | 8740 | 7000 | 300 | 1.24 | 29.13 |
| | Nvidia Tegra X1 | 512 | 450 | 7 | 19.42 | 73 |
| | Radeon firepro S9150 | 5070 | 3500 | 235 | 1.44 | 21.5 |
| | ARM Mali T880 MP16 | 374 | ? | 5? | ? | 74 |
| FPGA | Altera Arria 10 | 1500 | 3000 | 30 | 1.00 | 50 |
| | Altera Stratix 10 | 10000 | 2000? | 125? | 5.00? | 80 |
| | Xilinx Ultrascale+ | 4600 | 2000 | 40? | 2.30 | 115 |

- *NVIDIA/ARM GPU's vs Altera/Xilinx FPGA's*
- Max Performance per Watt may not be the best metric

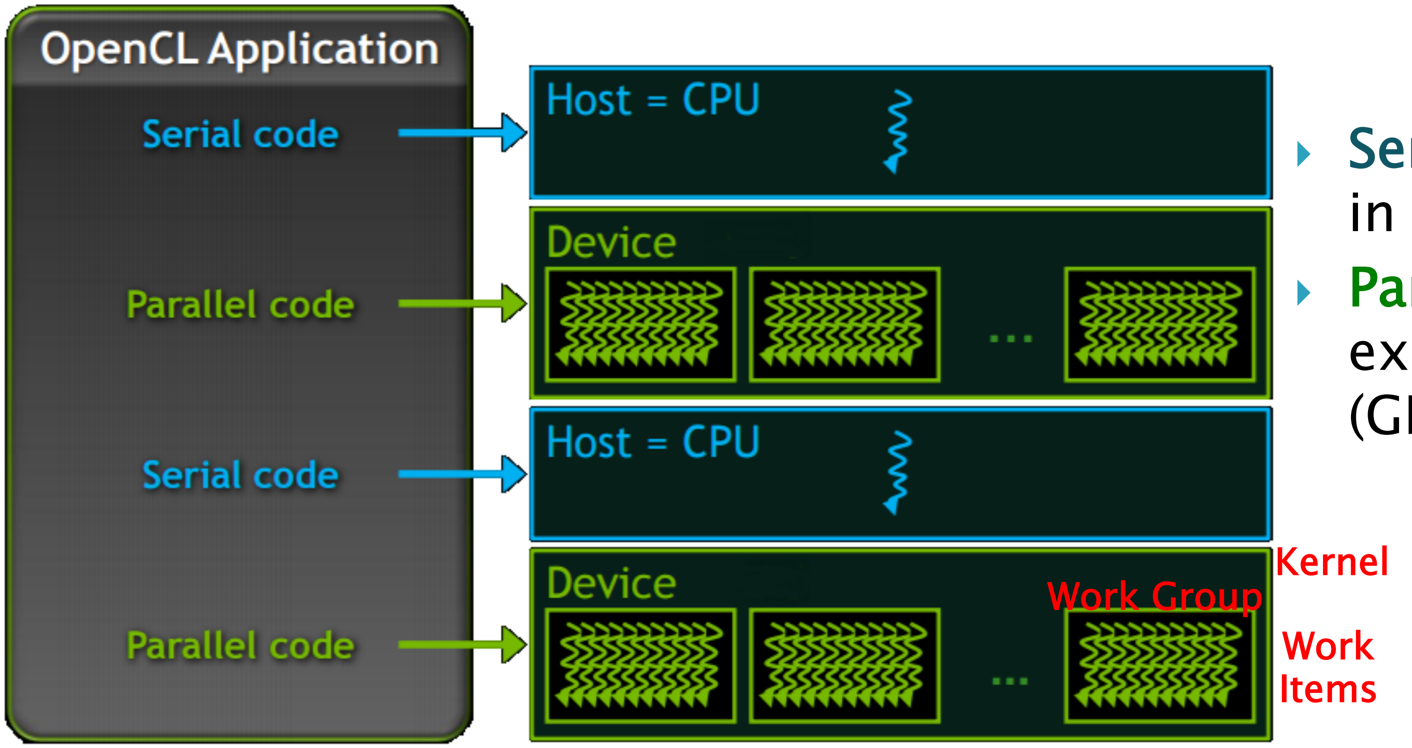
## Why are FPGA's so energy-efficient?

- Customized to the needs of the application
  - ✓ Instructions are part of the FPGA logic ☞ no IF, ID stages
  - ✓ Custom EX stage
  - ✓ Optimized data movements
  - ✓ Optimized control logic
  - ✓ Loop unrolling in HW

$$A = B + C$$
$$A = A * D$$



# OpenCL in a nutshell: SIMT



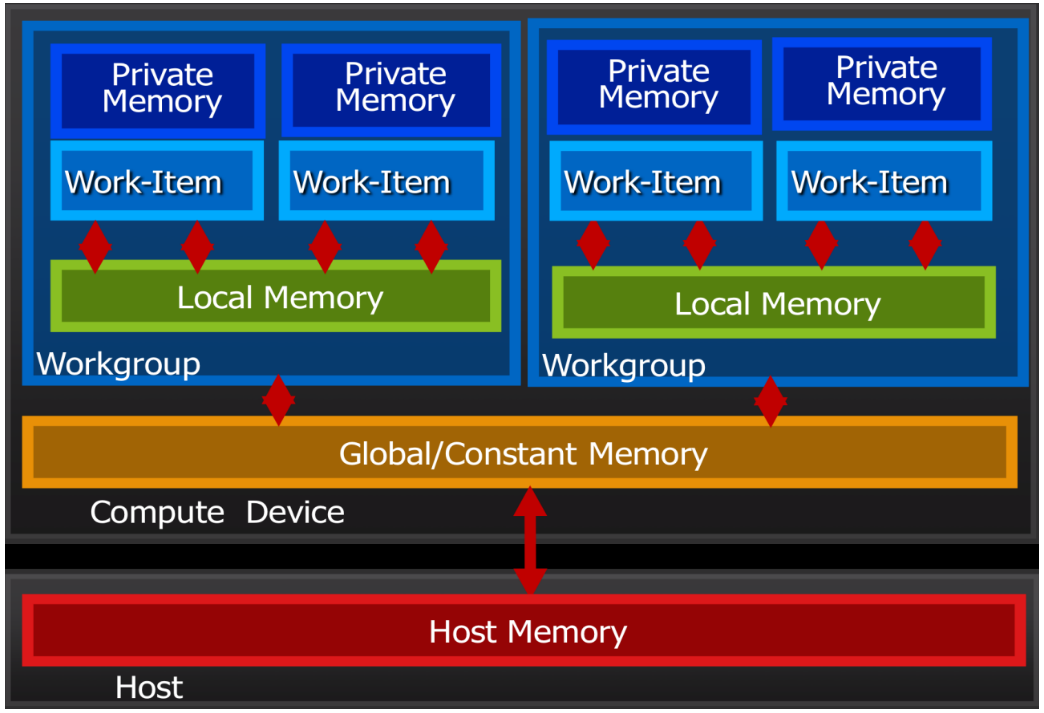
- **Serial** code executes in a **Host** thread
- **Parallel** code executes in **Device** (GPU/FPGA) threads

```
void
trad_mul(int n,
         const float *a,
         const float *b,
         float *c)
{
  int i;
  for (i=0; i<n; i++)
    c[i] = a[i] * b[i];
}
```

```
__kernel void
dp_mul(__global const float *a,
       __global const float *b,
       __global float *c)
{
  int id = get_global_id(0);

  c[id] = a[id] * b[id];
} // execute over n "work items"
```

# OpenCL in a nutshell: Memory Model



- Private Memory
  - Per work-item
- Local Memory
  - Shared within a workgroup
- Global/Constant Memory
  - Visible to all workgroups
- Host Memory
  - On the CPU

Explicit Memory Management

move data from host ⇨ global ⇨ local ⇨ private and back

# Language: Why OpenCL/CUDA for FPGAs?

✓ **Parallelism**
- Inside a work group (work items)
- Between work groups

✓ **Locality**
- Memory Hierarchy: Private, Local, Global memory

✓ **Simple and modular**
- Explicit Memory Transfers

✓ **Many applications already written in OpenCL**

☝ **But..**
- No Tree-like Hierarchy
- No Communication between work groups

# FPGAs in Data Centers

- Intel "Two Orders of Magnitude Faster than GPGPU by 2020":
  - Deep Learn. Inference Accel. (DLIA) with Altera Arria 10
  - Broadwell Xeon with Arria 10 GX
- EC2 F1 instance with Xilinx
  - Up to 8xUltraScale+ VU9P per instance
- IBM SuperVessel OpenPOWER development cloud using Xilinx SDAccel
- Microsoft Bing with Altera Stratix V
- FPGA networking for Azure cloud
- Xilinx SDAccell on Nimbix cloud
- Xilinx OpenStack support
  - Libraries: DNN, GEMM, HEVC Decoder & Encoder, etc.

## Still FPGA's are not used extensively – Why?

- Programmability
  - Huge Design Space ☞ hard to optimize application
    - Many FPGAs and choices
    - When/Where/What/How to accelerate
  - SW and HW skills required
- Compile/Synthesis time
  - Several hours to synthesize/PnR
  - Long time to program
- FPGA Size
  - FPGA Size directly affects speed and number of accelerated tasks
- Data movements (common in most accelerators)
  - High-latency links, such as PCIe
  - Several memory transfers between host and accelerator
- Cost
  - 2-3K € per FPGA
  - Acceleration as a service (Amazon)

## What should we do?
## Help the programmer!

- Improve Programming Language
  - Support of OpenCL 2.0
    - Shared virtual memory
    - Pipes
    - Dynamic Parallelism, Atomics, etc.
- Improve CAD Tools
  - Intel FPGA SDK for OpenCL (supports part of OpenCL 2.0)
  - Xilinx Vivado HLS (supports OpenCL 1.1)
- Improve Architecture
  - Multi-FPGA support
  - Multi-user support/shared resources
  - OS support (device drivers)
  - Runtime System
    - Dynamic Scheduling
    - Dynamic reconfiguration

# Ongoing efforts by ECOSCALE

- Improve Programming Language
  - Support of OpenCL 2.0
    - ✓ Shared virtual memory
    - Pipes
    - Dynamic Parallelism, Atomics, etc.

OpenCL

- Improve CAD Tools
  - Intel FPGA SDK for OpenCL (supports part of OpenCL 2.0)
  - Xilinx SDSoC, SDAccel (supports OpenCL 1.1)
  - OpenCapi

- Improve Architecture
  - ✓Multi-FPGA support
  - ✓Multi-user support/shared resources
  - ✓OS support (device drivers)
  - ✓Runtime System
    - Dynamic Scheduling
    - Dynamic reconfiguration

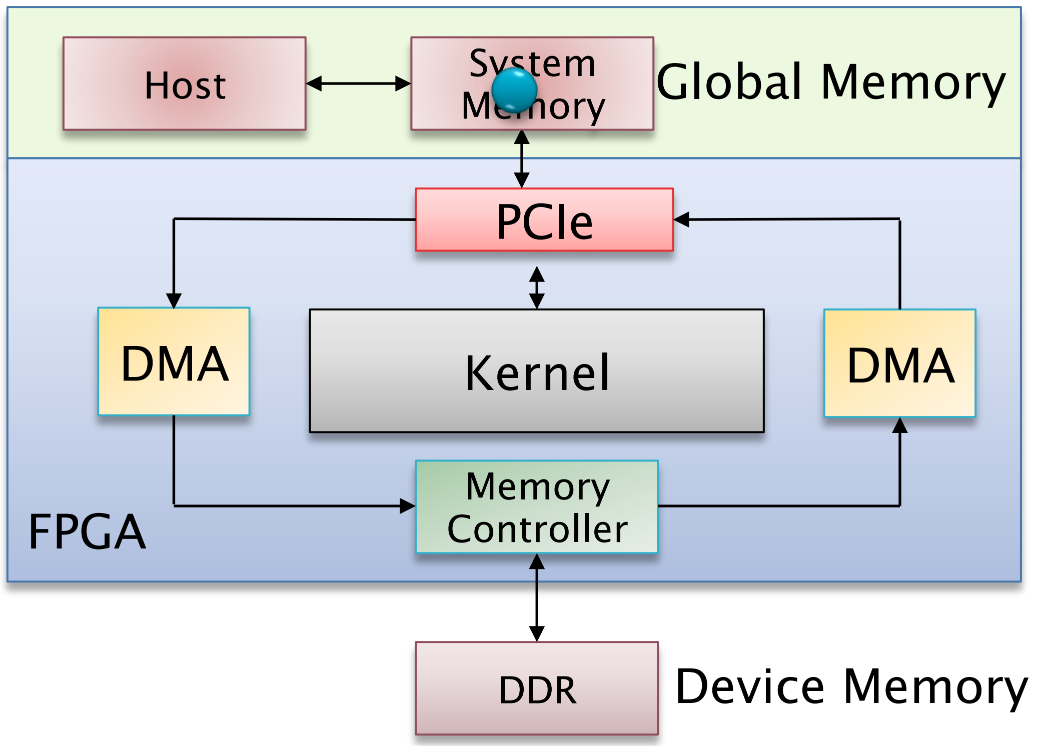
FAST CUDA

ECOSCALE

EURO SERVER

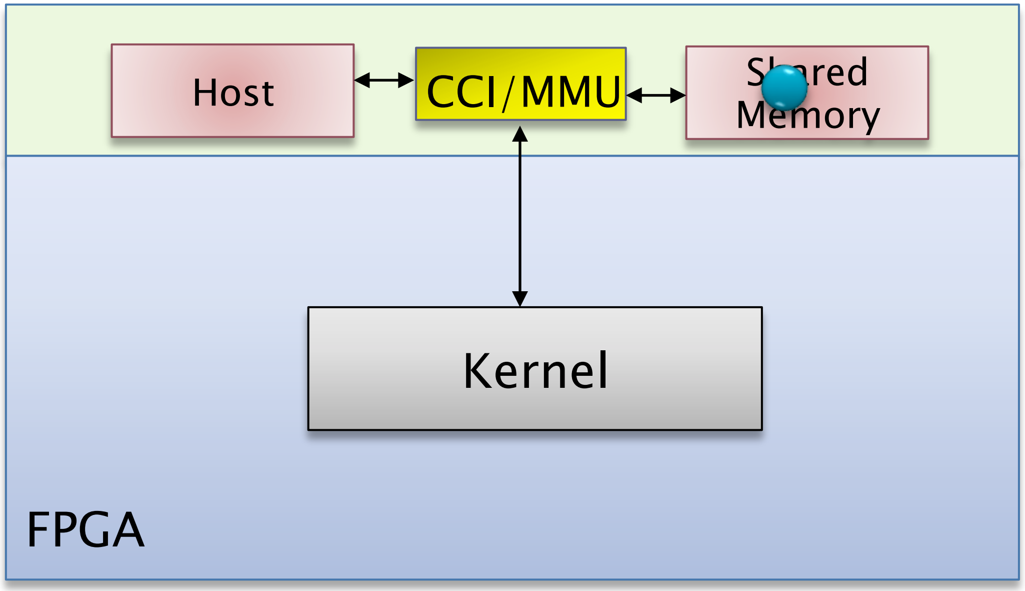
ExaNeSt

## Traditional OpenCL Data Movement



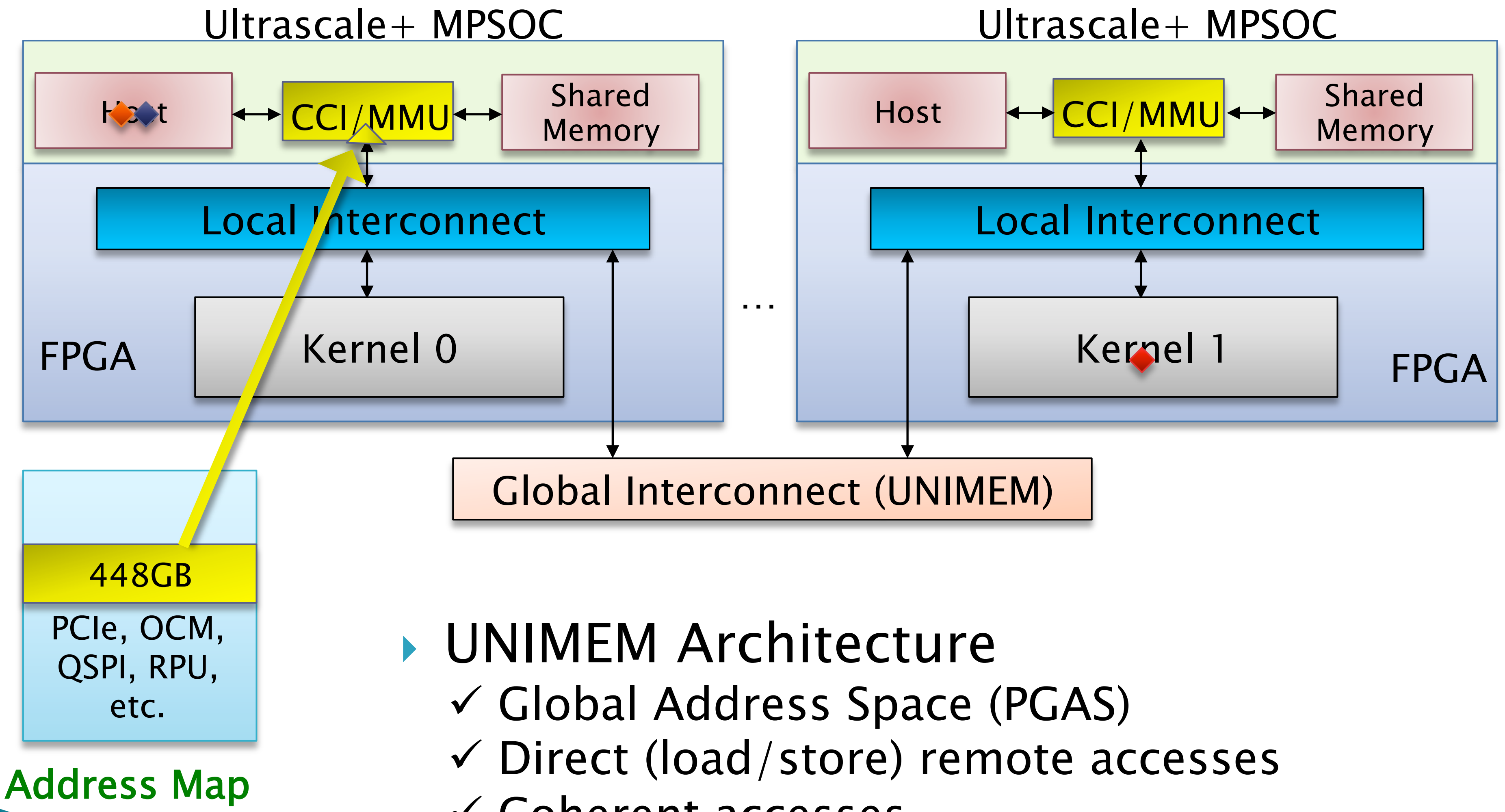
- Global Memory of Accelerator is Independent from Host/System Memory
- High-Latency PCIe

# OpenCL 2.0: Virtual Shared Memory



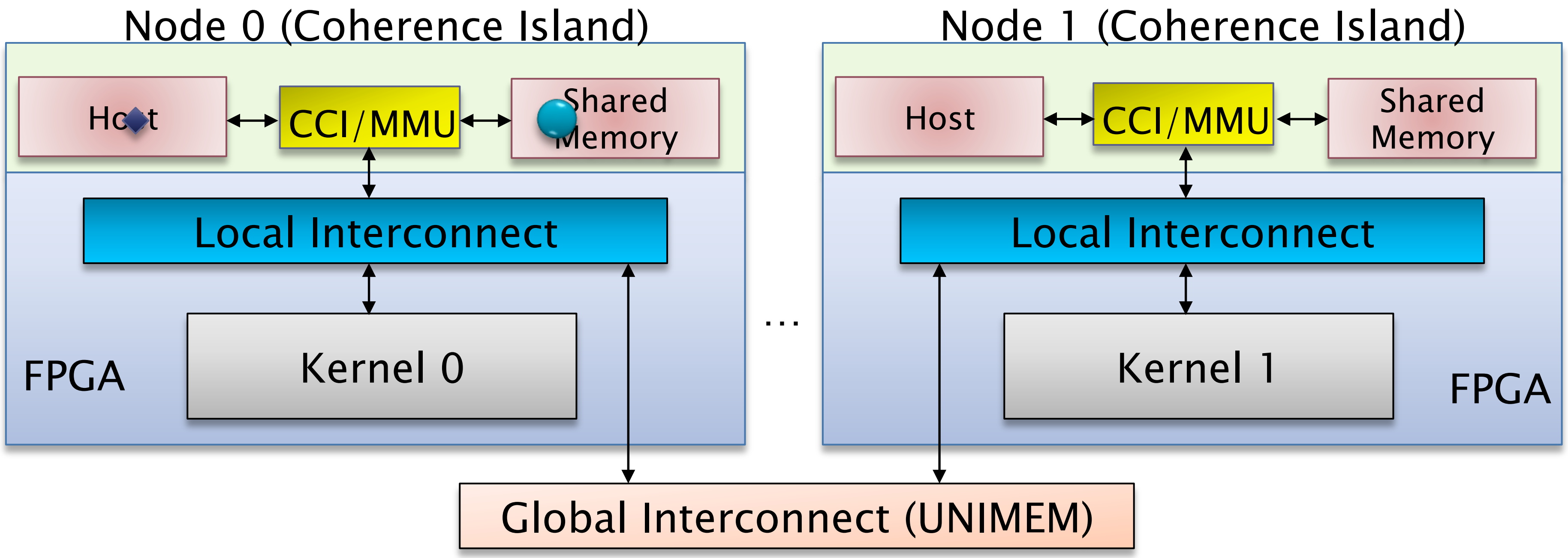
- **Cache Coherency**
  - ✓ ARM CCI (Ultrascale+)
  - ◦ Xilinx CCIX (next generation of Ultrascale+)
  - ◦ IBM CAPI (Intel QPI/CAPI)
- **IO MMU**
  - ✓ ARM SMMU (Ultrascale+)
  - ☞ Coherent Cache on the FPGA (Ultrascale+)?

# UNIMEM: Remote Coherent Accesses



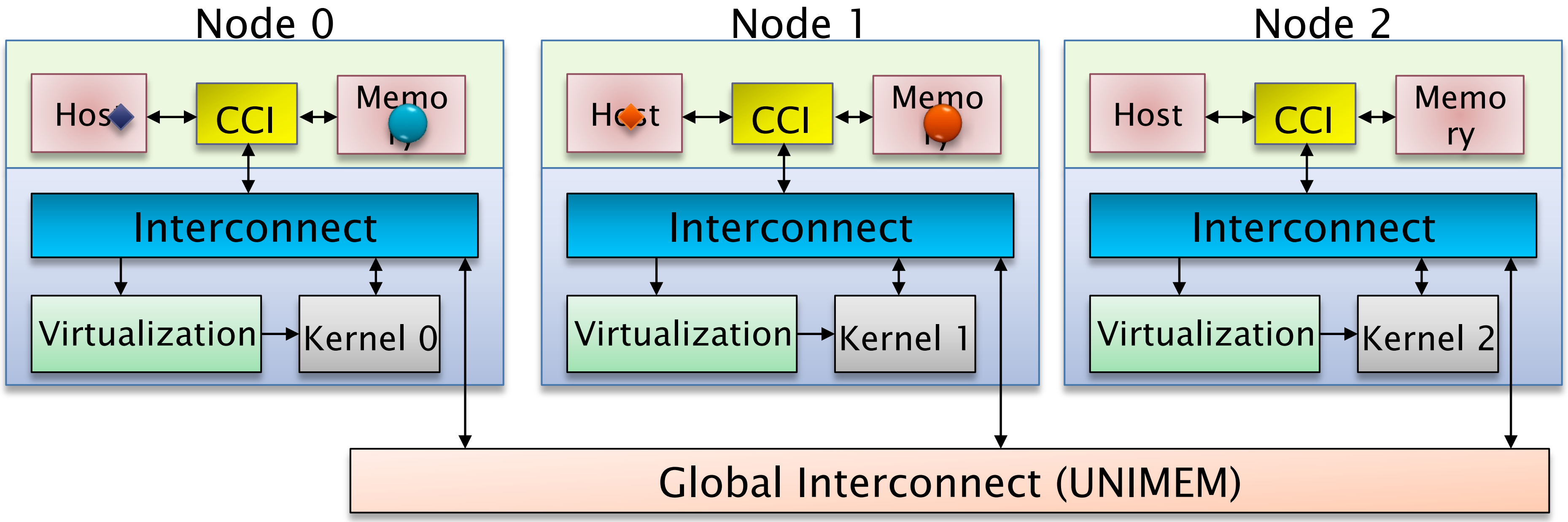
- UNIMEM Architecture
  - ✓ Global Address Space (PGAS)
  - ✓ Direct (load/store) remote accesses
  - ✓ Coherent accesses

# Multiple FPGAs



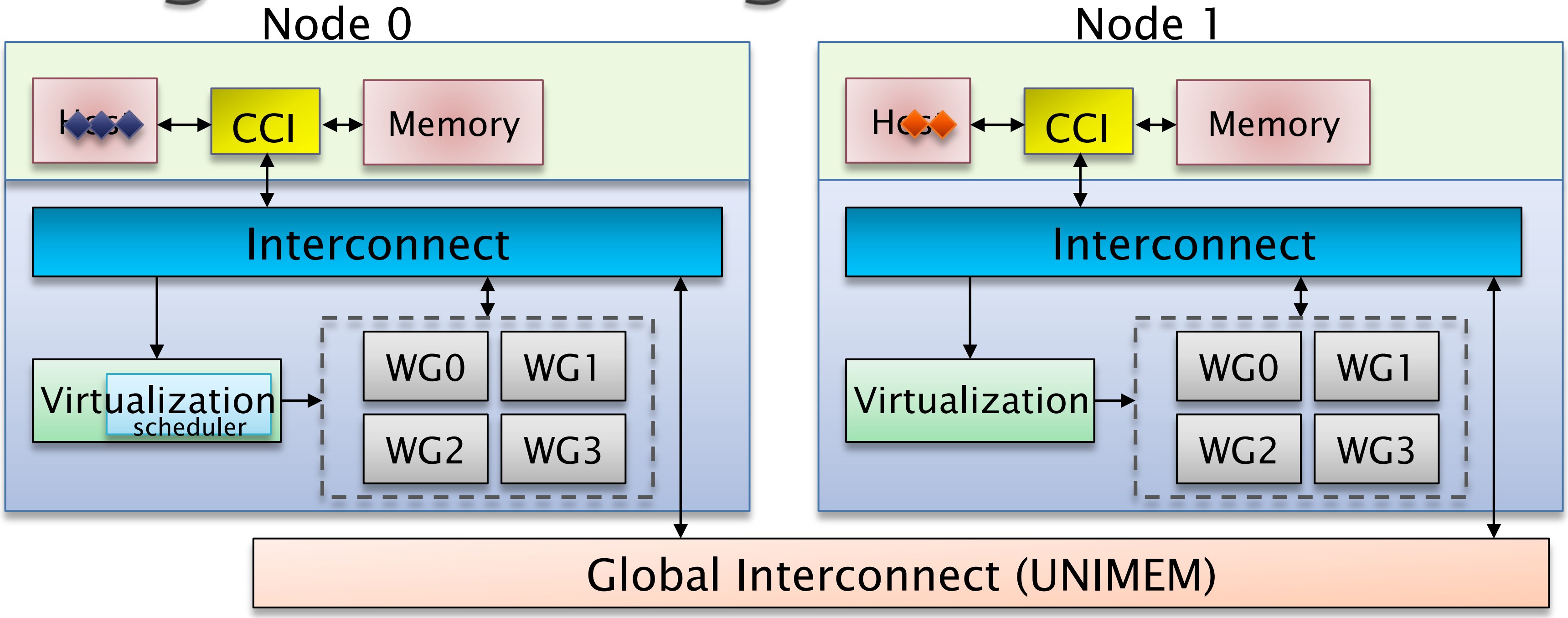
- Access any FPGA in the System
  - ✓ Remote Kernel calls
  - ✓ Remote Memory Accesses

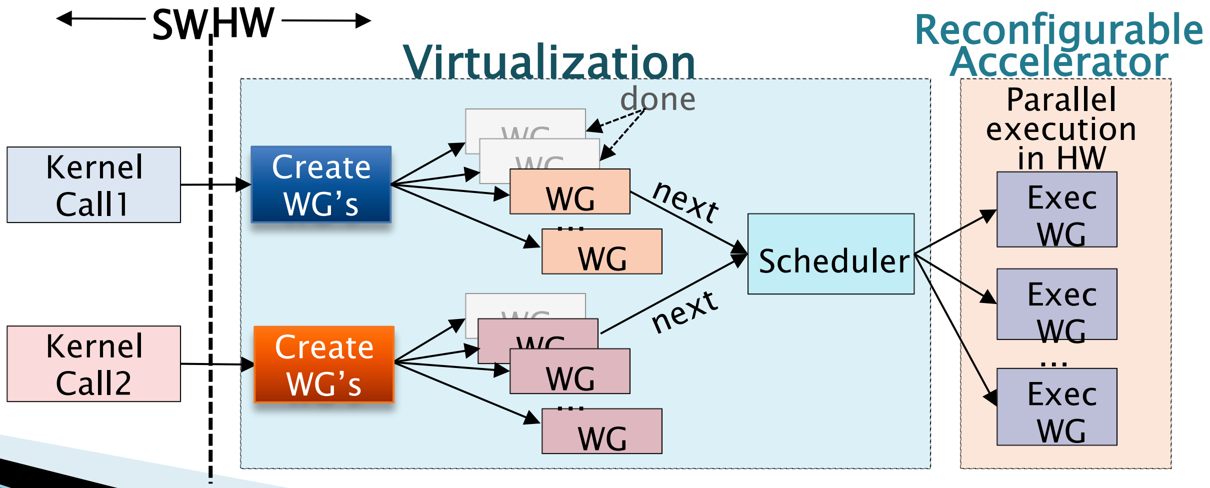
# Resource sharing



- Virtualization Block
  - Receives Kernel Execution command from any Node
  - Schedules commands and executes them locally

# Fine-grain sharing

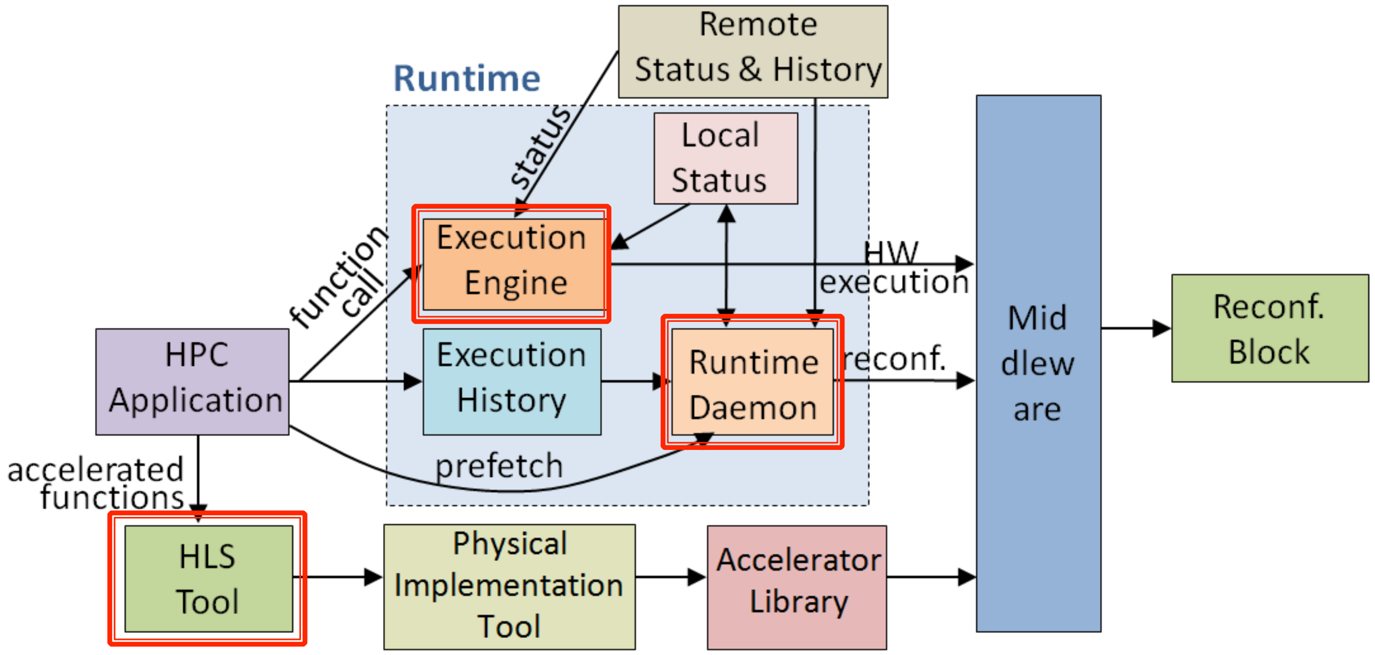




# Dynamic Reconfiguration: Challenges and Potential Solutions

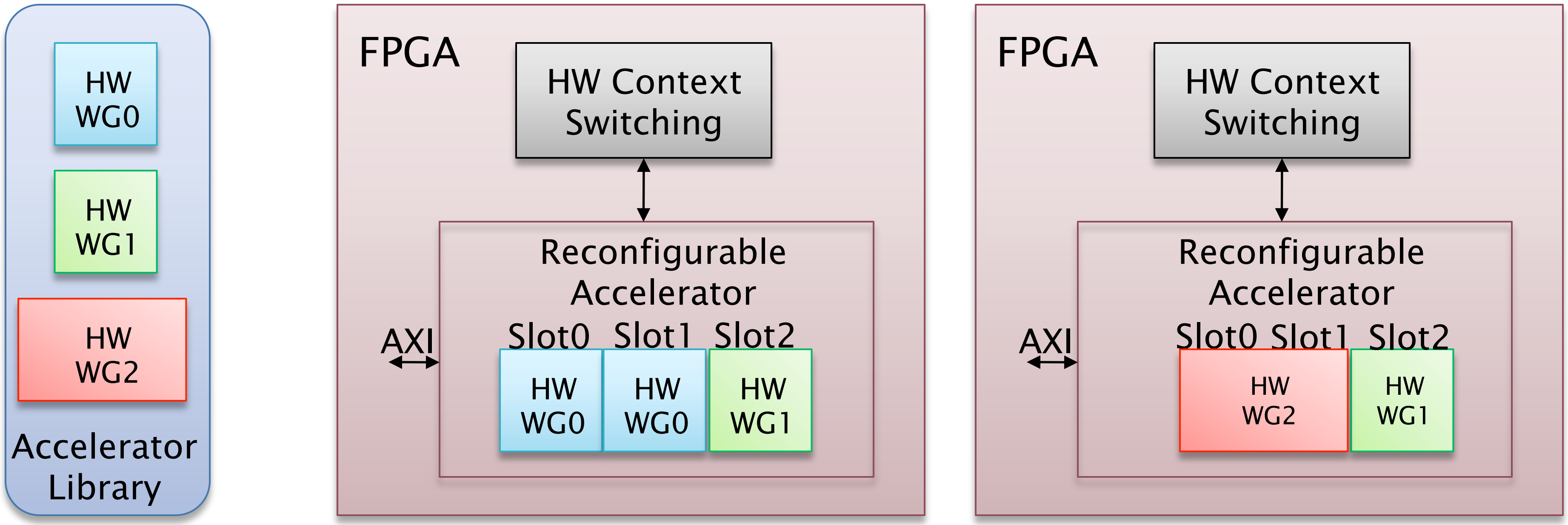
| | | |
|---|---|---|
| ◦ Synthesis/PnR too slow | $\Rightarrow$ | Synthesis at compile time |
| ◦ Bitstream per FPGA per location | $\Rightarrow$ | Improve Placement |
| ◦ Reconfiguration is slow | $\Rightarrow$ | Prefetching<br>GPU/CPU as alternative |
| ◦ Limited FPGA size | $\Rightarrow$ | Improve Architecture (UNIMEM) |
| ◦ Limited tool support | $\Rightarrow$ | Standardization/ Improve Tools |
| ◦ When/what/where to reconfigure | $\Rightarrow$ | Runtime System |

# Execution Environment



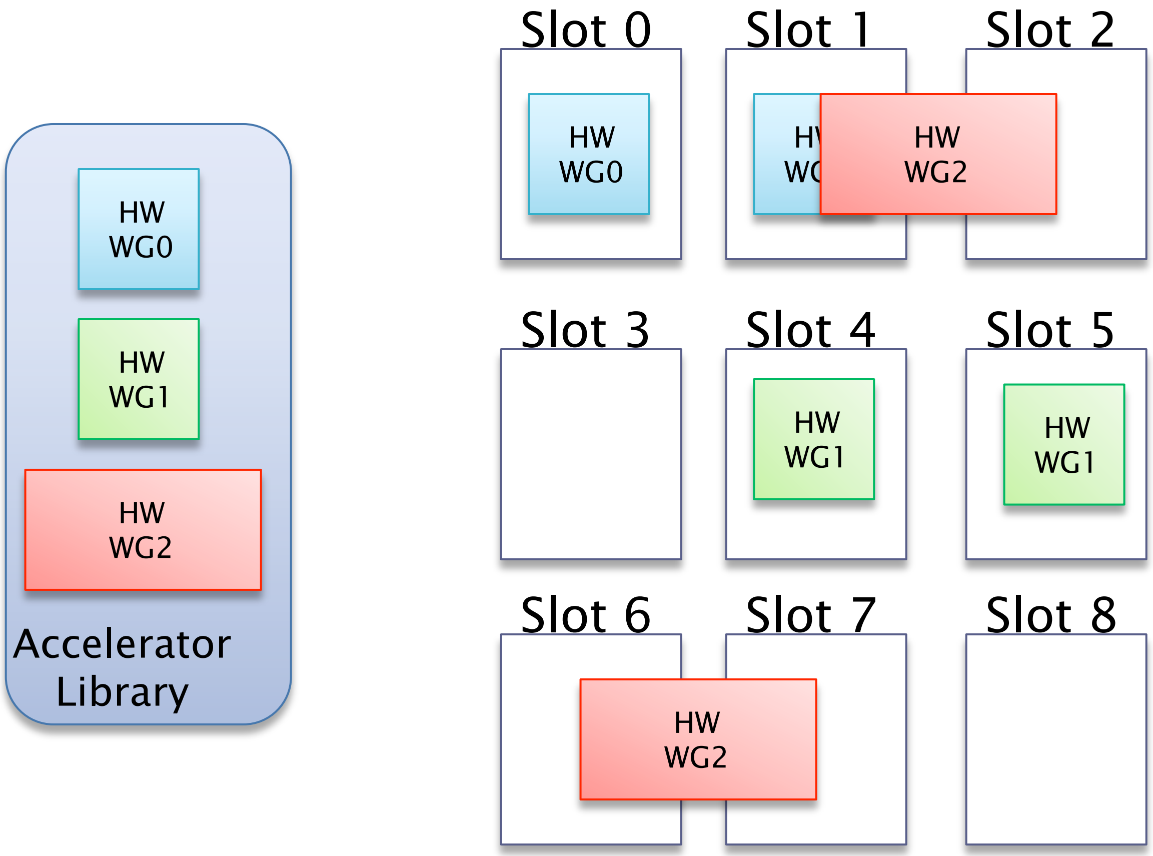
- Reconfiguration at runtime
- Task Execution Scheduling
- Task Execution Monitoring
- Locality Management

# Physical Implementation



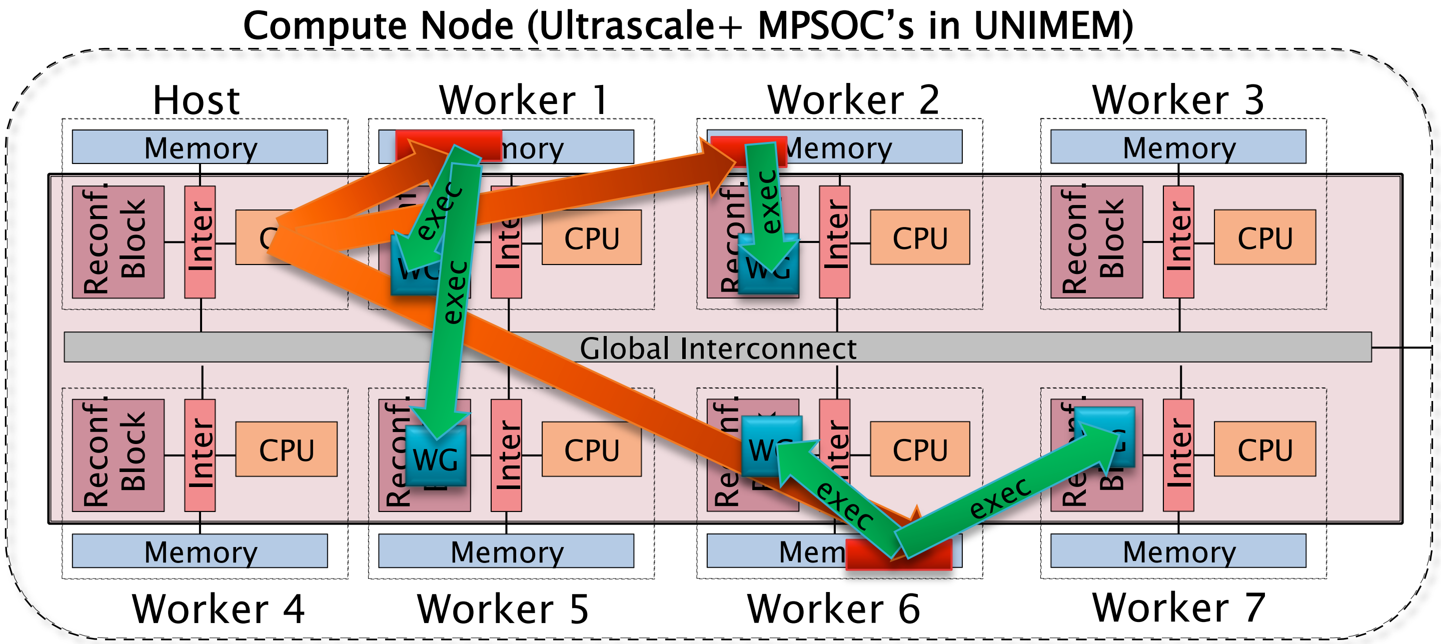
- **Resource-aware Reconfigurable Accelerator Floorplanning and Backend Tool**

# ECOSCALE Recofiguration



- Acceleration Library
- WG's of different size
- Relocation
  - Defregmentation
  - Move logic closer to data

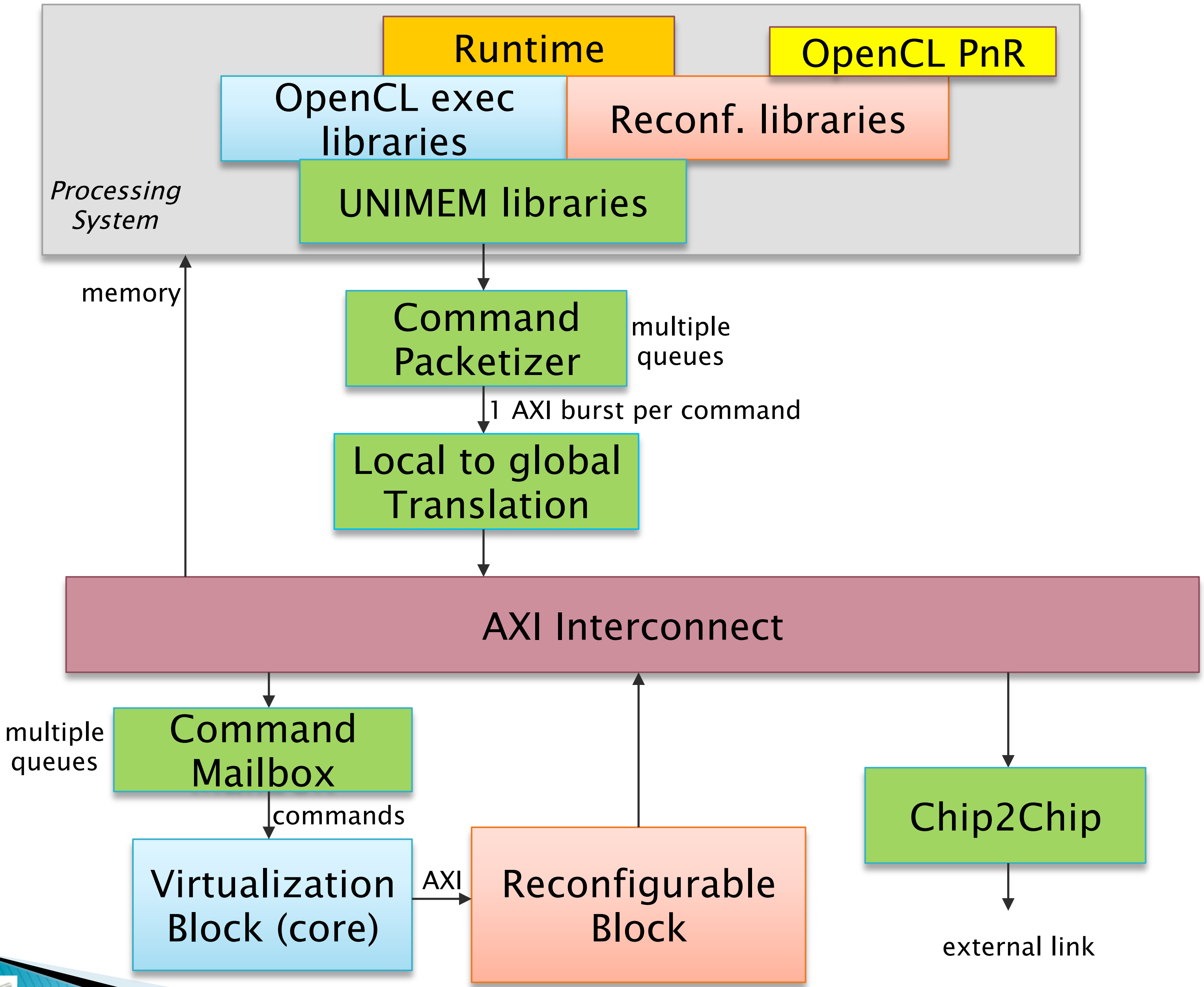
# Runtime System



- Runtime System
  - Reconfigure resources
  - Distribute Data and Computation ☞ Locality
  - Monitoring

# Conclusion

- Common HPC Approach
  - ☞ Use accelerators: GPUs vs FPGAs
- FPGAs main advantage
  - 👍 **Energy Efficiency**
- FPGAs soon in HPC servers
  - ☝ But hard to be used by programmers
    - Limited FPGA Size ☞ Sharing of resources
    - Reconfiguration ☞ Optimize tools/Standardization
    - Runtime system which automates/coordinates actions


Runtime
OpenCL PnR
OpenCL exec libraries
Reconf. libraries
Processing System
UNIMEM libraries
memory
Command Packetizer
multiple queues
1 AXI burst per command
Local to global Translation
AXI Interconnect
multiple queues
Command Mailbox
commands
Virtualization Block (core)
AXI
Reconfigurable Block
Chip2Chip
external link

www.ecoscale.eu

@ecoscale_H2020

Stay Tuned!