ESSPER
Elastic and Scalable System for High-Performance Re-configurable Computing

RIKEN R-CCS PROCT

Challenges for Reconfigurable HPC with FPGA Cluster "ESSPER" Connected to Supercomputer Fugaku

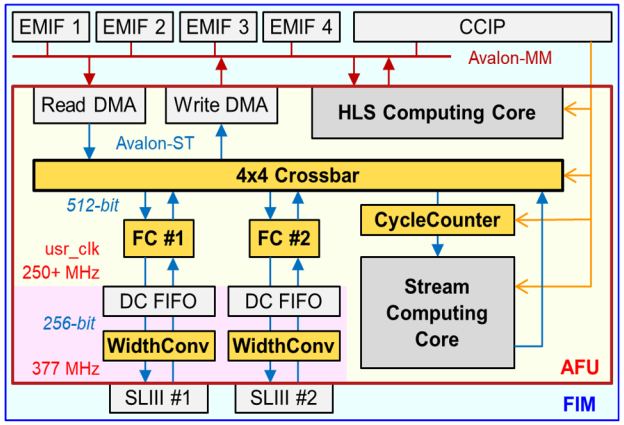
Kentaro Sano

Processor Research Team
RIKEN Center for Computational Science

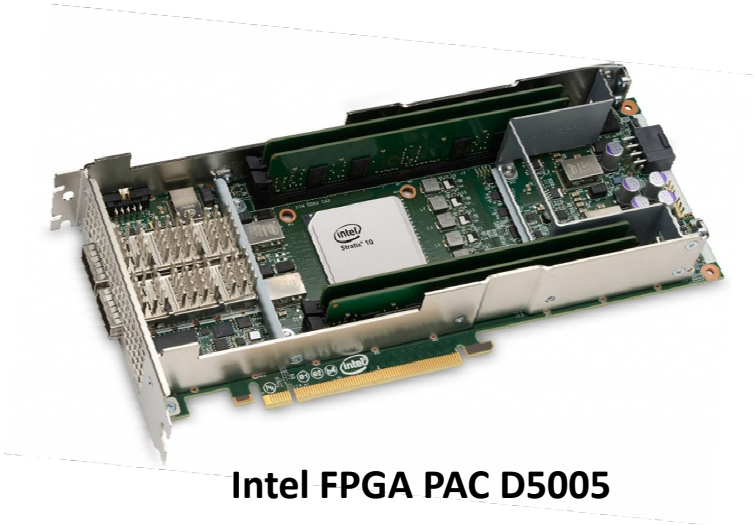
Sep 7, 2021

Outline

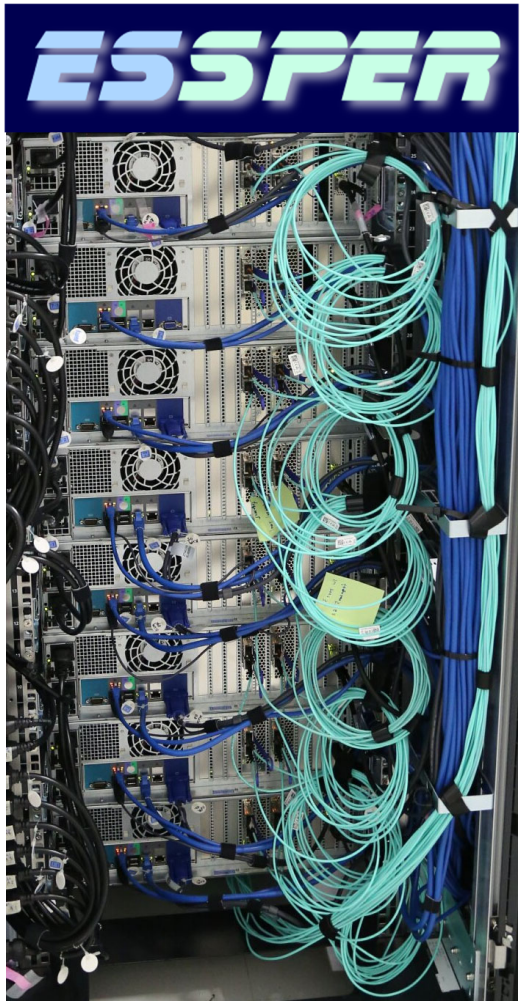
- Introduction
- Reconfigurable HPC with FPGAs and its challenges
- **ESSPER :** Proof-of-Concept FPGA Cluster System
- Summary



AFU Shell



Intel FPGA PAC D5005



PoC FPGA Cluster System

Introduce Myself : Kentaro Sano

Hiring researchers:
R-CCS2015 or R-CCS2022

RIKEN Center for Computational Science

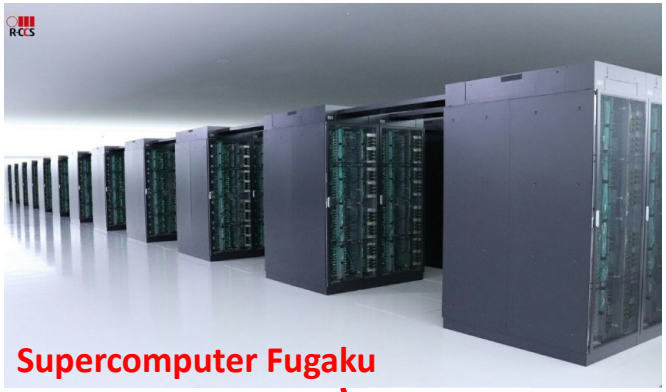
- ✓ Develop and operate Supercomputer Fugaku
- ✓ Facilitate leading edge infrastructures for research based on supercomputers
- ✓ Conduct cutting-edge research on HPC

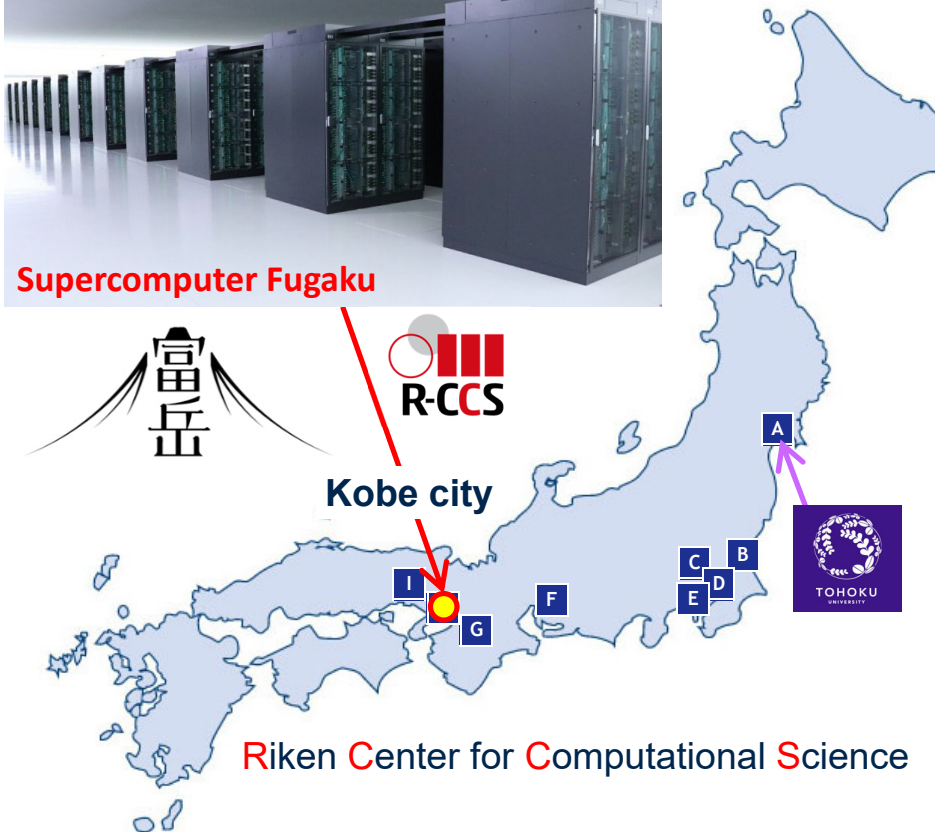
Leader, Processor Research Team

- ✓ Exploration of future HPC architectures
- ✓ Advanced use of present HPC systems

Joint Laboratory at Tohoku University

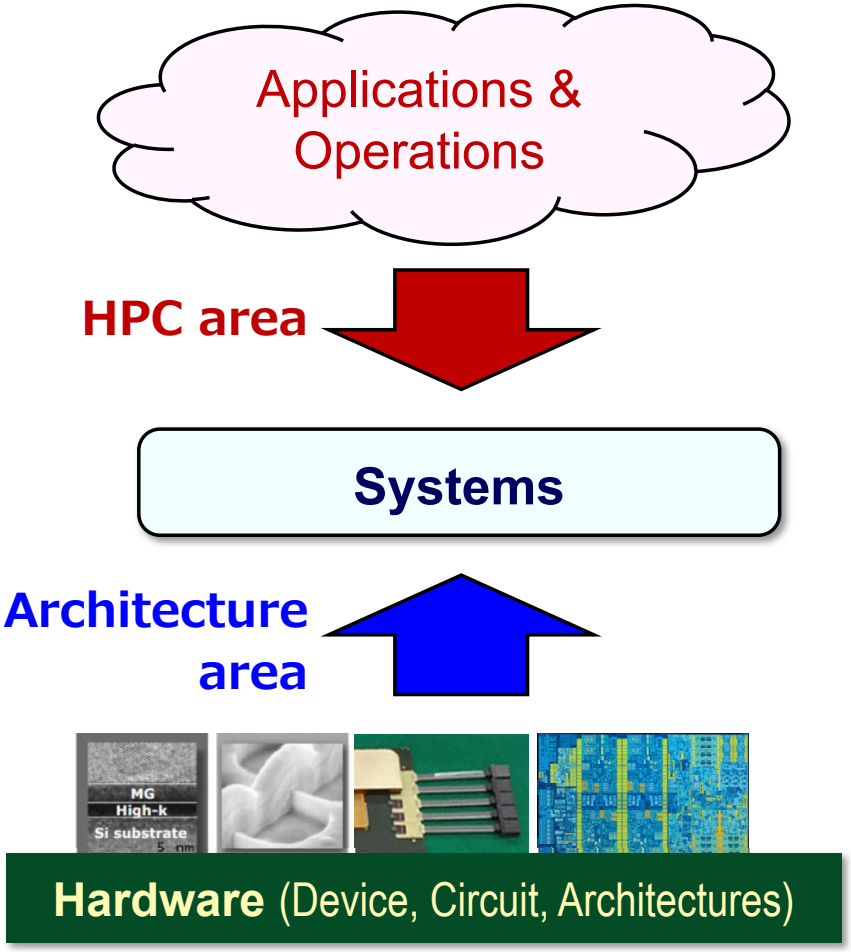
- ✓ Visiting Professor
  "Advanced Computing Systems Lab"





I'd like to Invent a New Computer!

Explore new architectures and systems.



Goal and Roadmap of the Team

Establish HPC architectures suitable for Post-Moore Era

2020 — 2021 — 2022 — 2023 — 2024 — 2025 — 2026 — 2027

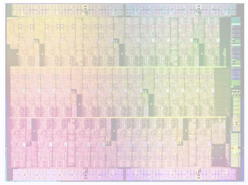
Advancement of Fugaku

This talk

- ✓ **Functional extension with FPGAs and eco-system**
- ✓ System software and apps of task-flow computing

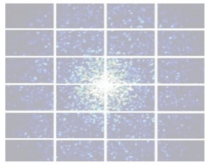
Exploration of HPC Architectures

- ✓ Novel processors based on CGRA / Data-flow architectures
- ✓ New organization of computing nodes with CPU and novel processors



Near-sensor / Near-storage Processing

- ✓ FPGA-based processing for X-ray imaging detector

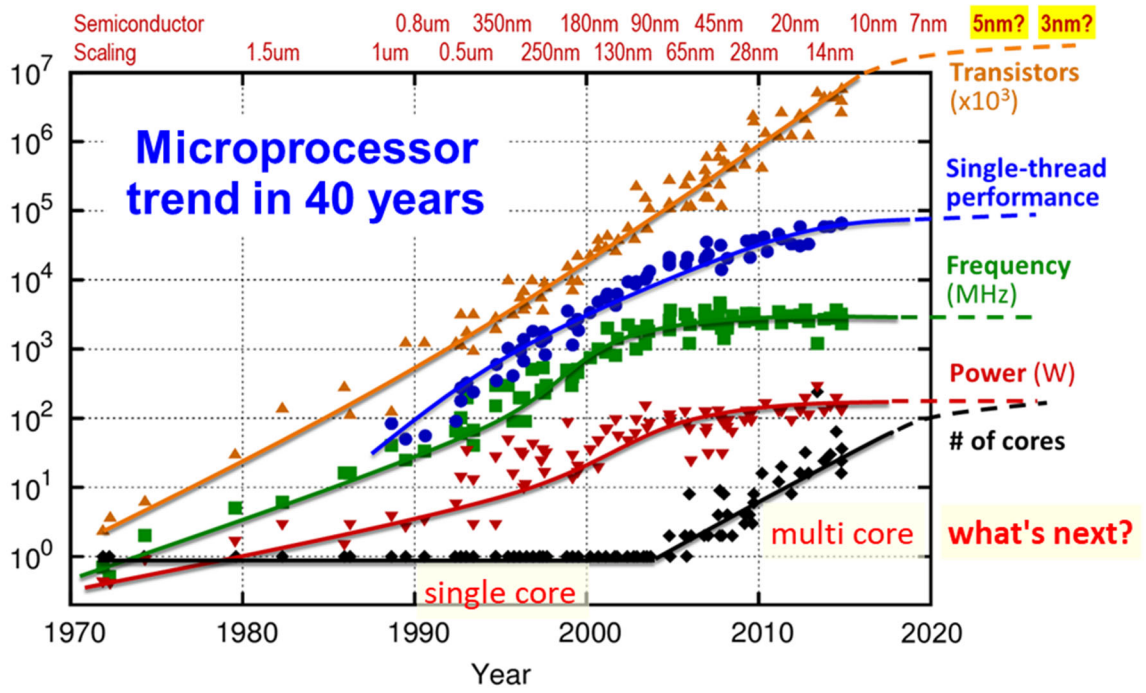


Exploration of Novel Computing Principle and Model

- ✓ Neuromorphic computing and its applications to CPS

Motivation of Reconfigurable HPC with FPGAs

Introduction



Present mainstream : many-core

(perf) = (# cores) x (freq) x (utilization)

Can many-core continue to scale?

- ✓ **# of cores ?** <-- end of Moore's law
- ✓ **frequency ?** <-- end of Dennard scaling
- ✓ **utilization ?** <-- Neumann architecture and inefficient data movement
- ✓ **Performance per power ?** --> **difficult to further improve**

Seems difficult to scale performance of Many-core. But is this REAL?

Difficult to Scale! ... Especially, Data movement

Core performance doesn't increase.

- ✓ Limited frequency, limited parallelism
- ✓ Extra hardware required to boost IPC

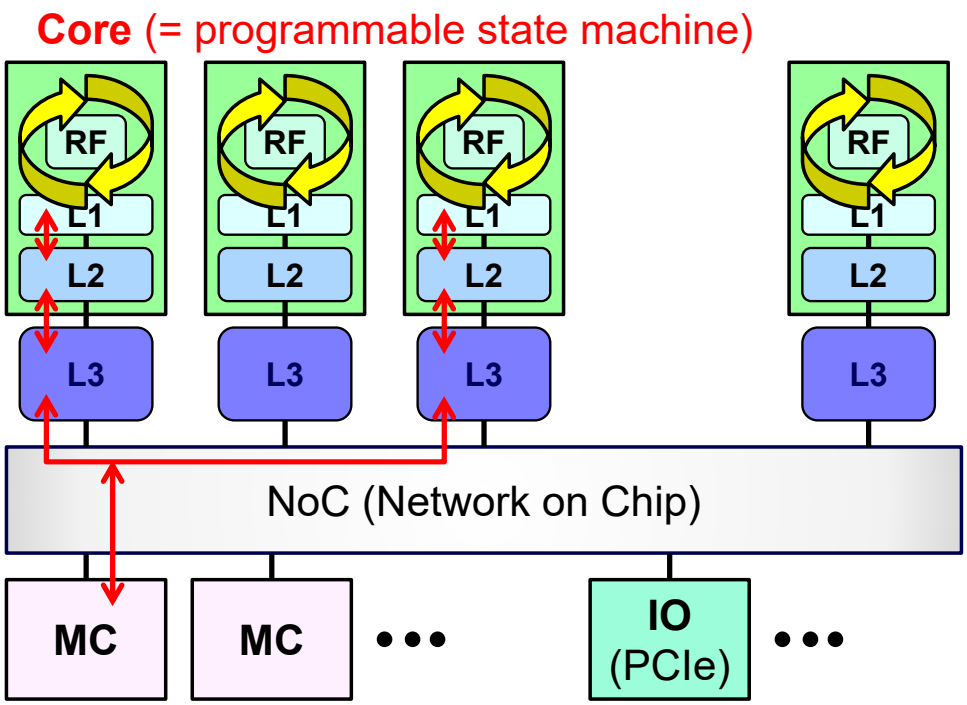
Moving data via memory doesn't scale.

- ✓ NoC and cache can be a bottleneck of data supply from external memories.
- ✓ Inter-core data-movement is inefficient.
  > latency in writing/reading cache memories

NoC itself and shared LLC doesn't scale.

- ✓ Scalable NoC is challenging.
- ✓ $-coherence protocol gives higher pressure.

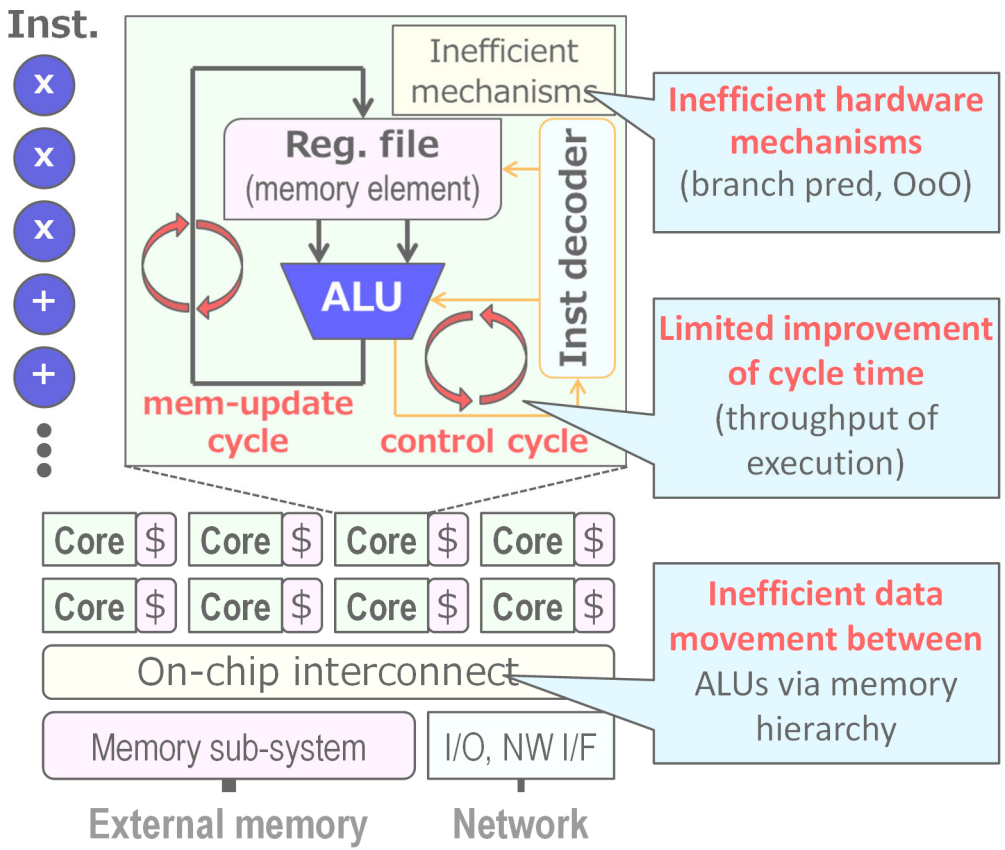
No more improvemewnt in perf/power.



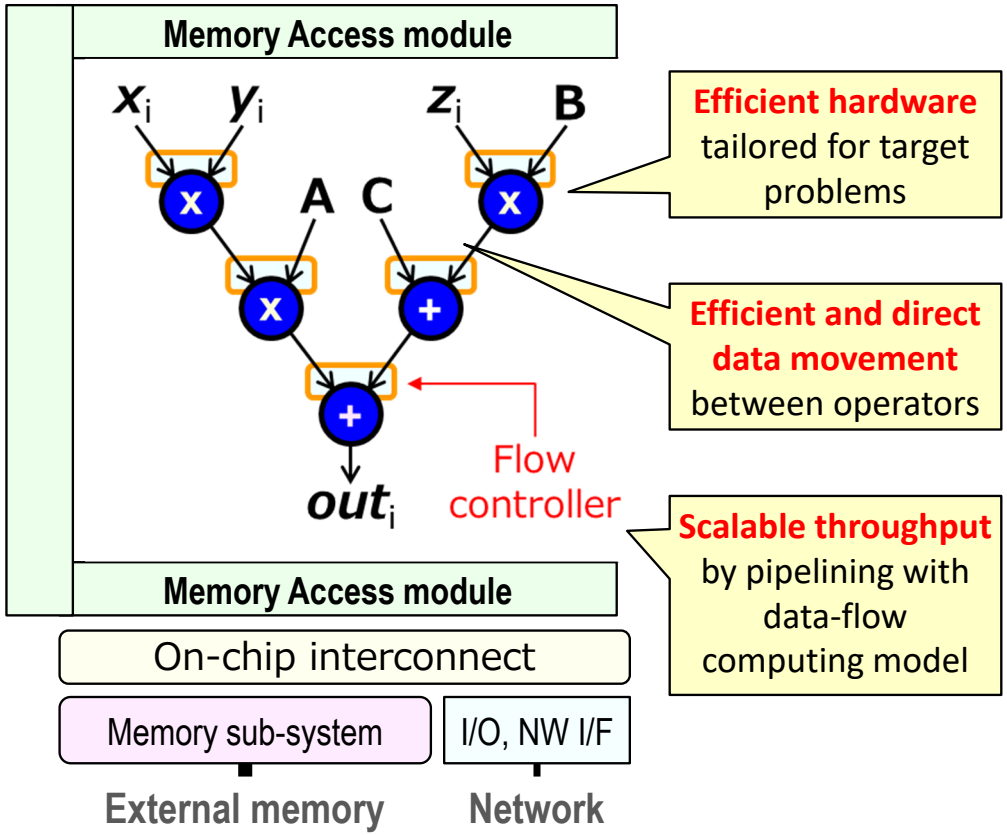
How can we make it scalable?

Solution : Costom Data-Flow Computing

Many-core processor

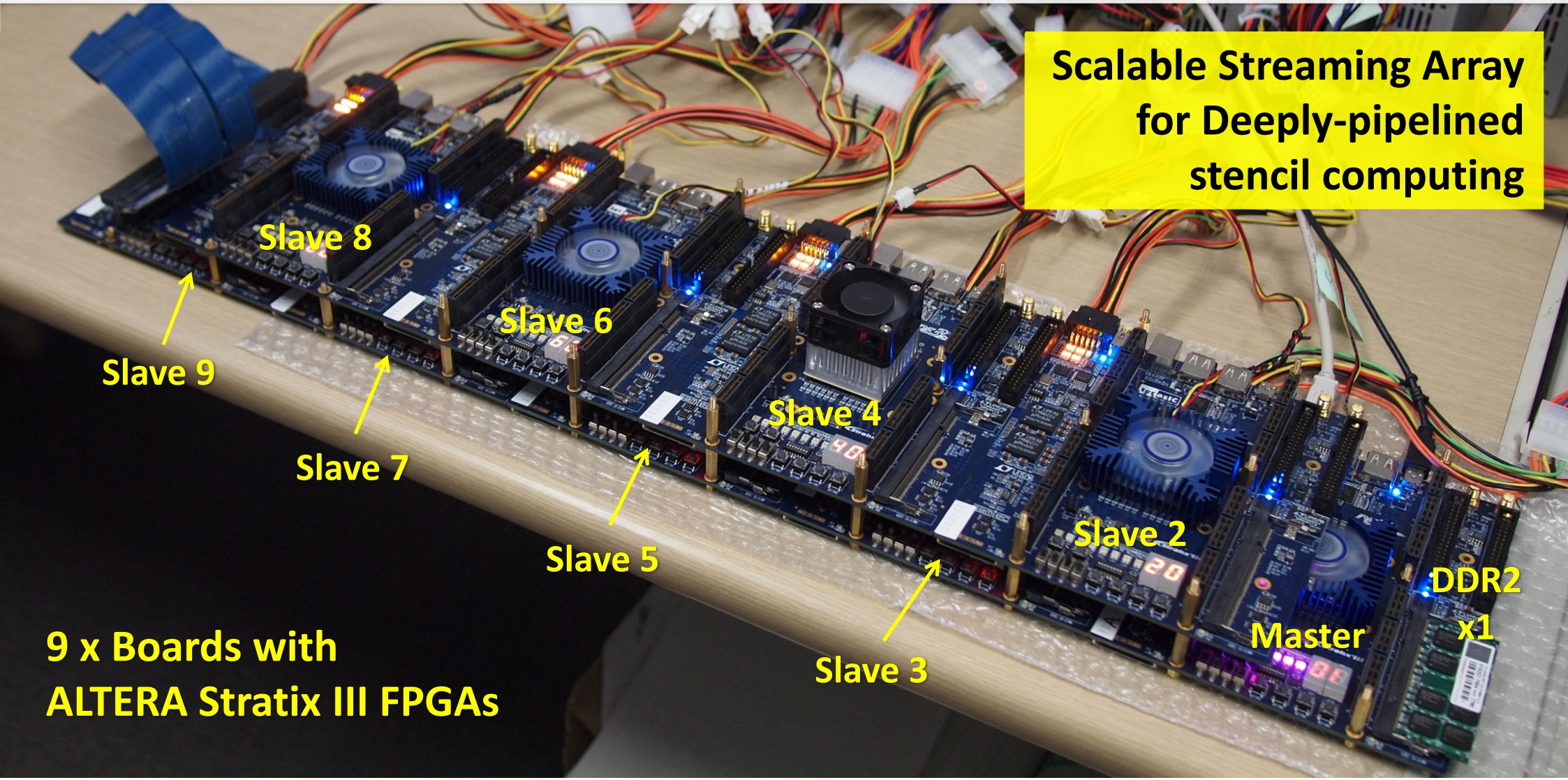


Custom data-flow computing



FPGAs Allow us DIY for HPC!

[KSano, FCCM2011]

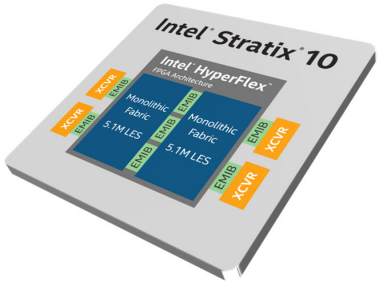


FPGA's High Potentials for HPC

The state-of-the-art FPGA

- ✓ High-performance operation
- ✓ High-bandwidth external memories
- ✓ Ultra high-bandwidth on-chip memories
- ✓ Fast inter-device communication

Intel Stratix 10 FPGA (14nm)



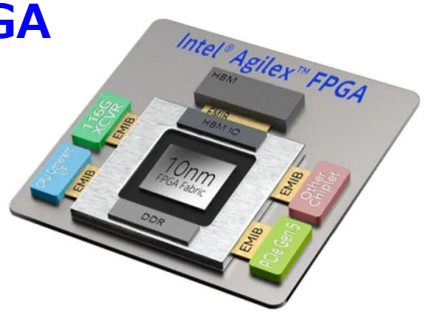
- ➢ 5760 floating-point DSPs
- ➢ comparative to CPU, GPU (DDR4, HBM2)
- ➢ aggregate ~1000 TB/s
- ➢ multiple tx / rx of 100 Gbps

Use cases in data-center, cloud, or HPC systems

- ✓ *Microsoft Catapult, AWS EC2 F1, Alibaba Cloud, Tencent Cloud, Huawei Cloud*
- ✓ *Tsukub U Cygnus, Paderborn U Noctua*

Intel Agilex FPGA

More advanced next-generation 10nm FPGA



Promising not only for computing, but also for data movement

Challenges for Reconfigurable Computing with FPGAs

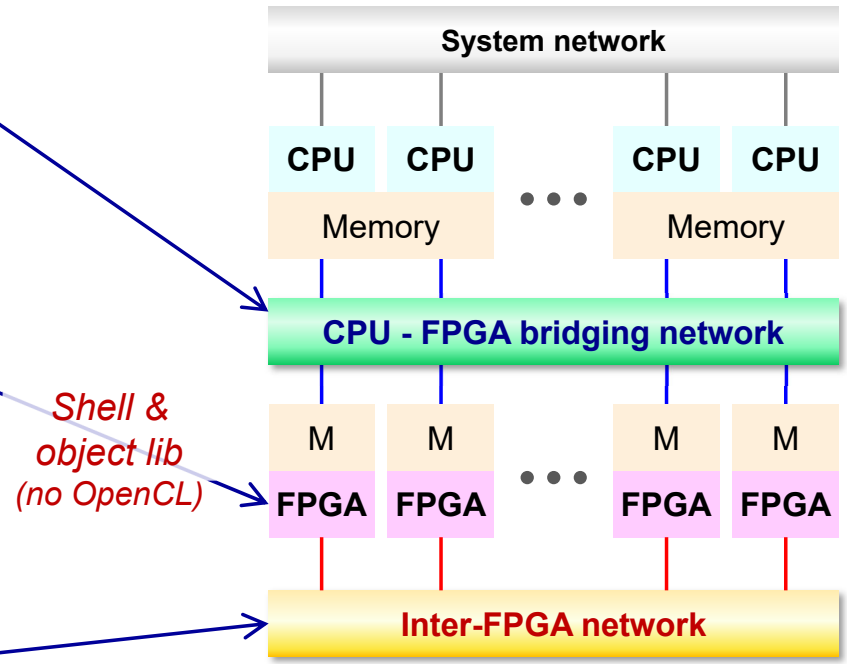
What are Missing for FPGA-based Systems?

FPGAs are not a main stream, because we are missing:

Interoperability and flexibility in operation
- ✓ To allow FPGAs to be installed in various systems
- ✓ Flexible resource utilization in a large system

Platform for sufficient customizability
- ✓ To program hardware algorithms as we want
- ✓ At acceptable productivity

Techniques for performance scalability
- ✓ Dedicated network for multiple FPGAs
- ✓ Appropriate parallel computing models

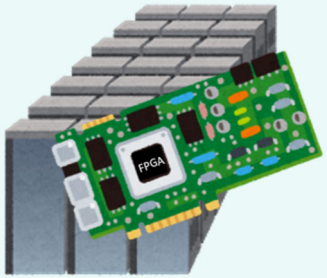


Our approach

Requirements for FPGA-based HPC Systems

Req. 1 Interoperability w/ various HPC systems

- ✓ Able to easily extend existing systems with FPGAs
- ✓ Can we extend Supercomputer Fugaku?

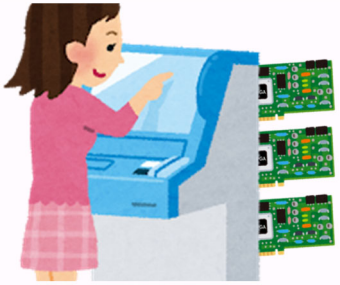


Req.2 Flexibility for using FPGAs in a system

- ✓ Allow any CPUs to flexibly utilize FPGAs in a system
- ✓ Appropriate for a machine shared with multiple users
- ✓ High utilization of FPGAs

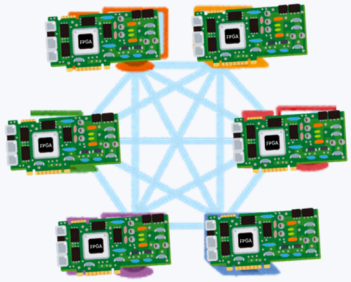
Req.3 Platform for sufficient customizability

- ✓ Able to implement various hardware (algorithms) on FPGA
- ✓ Give a high productivity by providing common SoC and its software abstraction



Req.4 Techniques for performance scalability

- ✓ Allow low-latency and high-throughput communication among FPGAs
- ✓ Allow users to easily try multi-FPGA applications



Approaches for Proof of Concept

Req. 1 Interoperability w/ various HPC systems

- ✓ Able to easily extend existing systems with FPGAs
- ✓ Can we extend Supercomputer Fugaku?

Req.2 Flexibility for using FPGAs in a system

to flexibly
stem
achine
e users
ion of FPGAs

Software-bridged FPGA driver

FPGA resource manager

Req.3 Platform for sufficient customizability

- ✓ Able

inte

Req.4 Techniques for performance scalability

- ✓ All

try

FPGA Shell (SoC)

HLS-based programming flow, & FPGA object lib

Inter-FPGA network

Virtual circuit switching over Ethernet

ESSPER

Elastic and Scalable System for High-Performance Re-configurable Computing

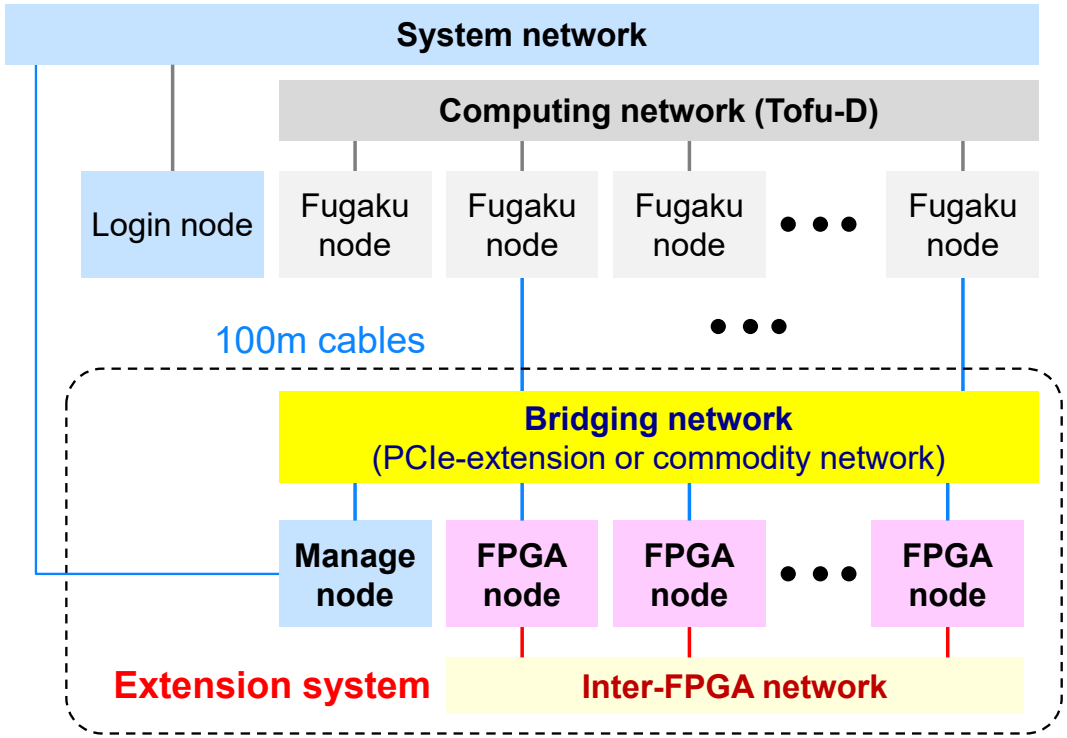
ESSPER : Proof-of-Concept FPGA Cluster System

ESSPER
Elastic and Scalable System for High-Performance Re-configurable Computing

Architecture of ESSPER

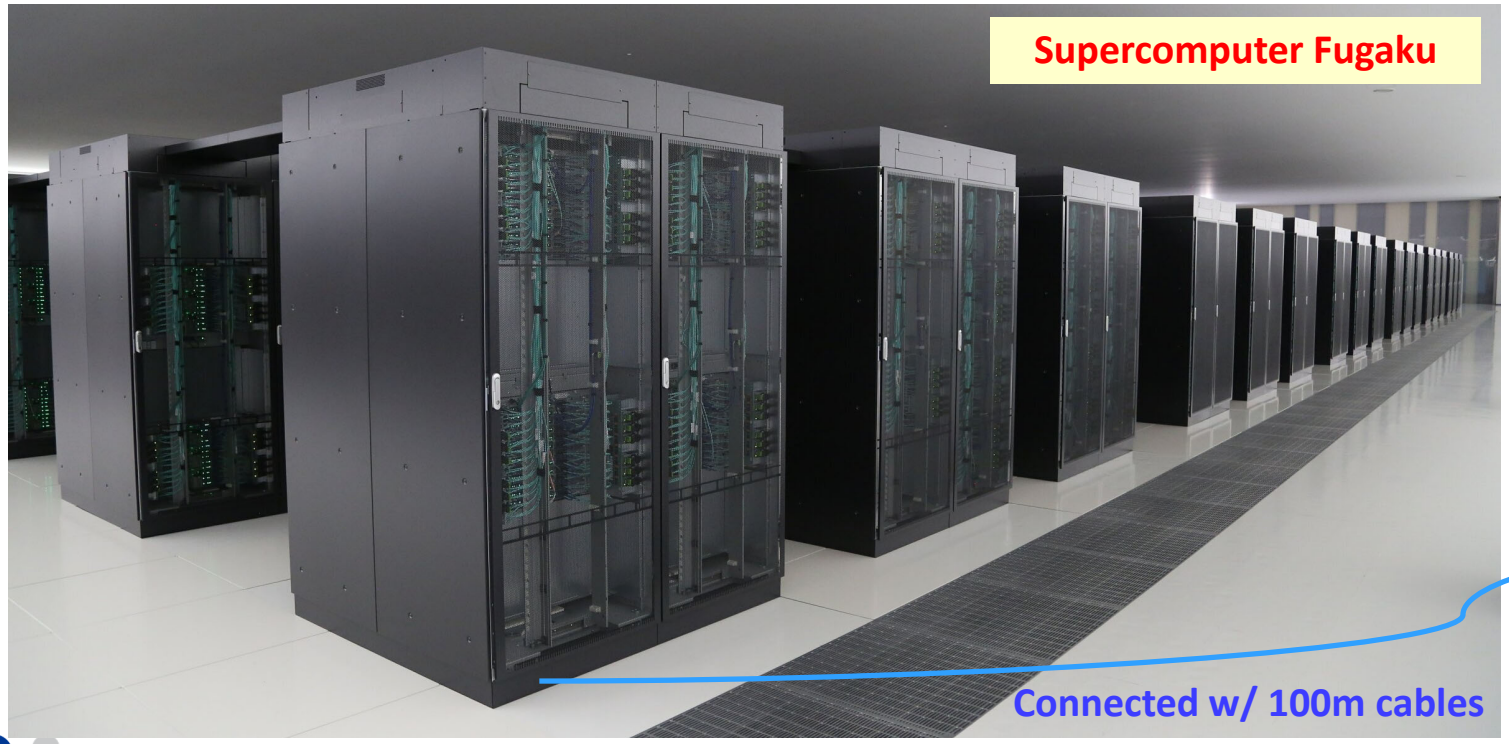
Project to investigate functional extension of Fugaku

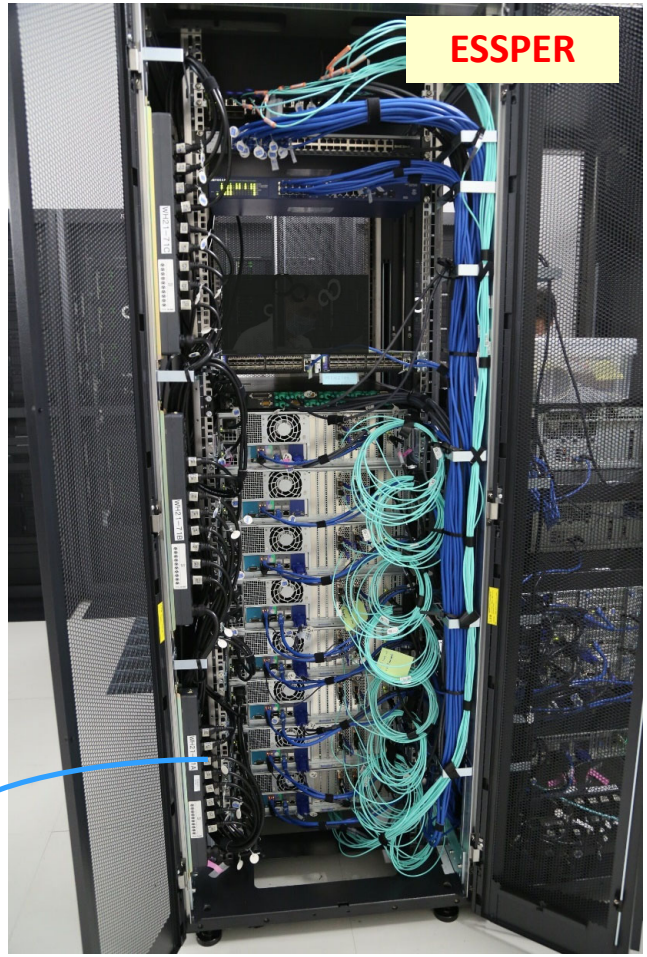
- FPGA Cluster
- CPU-FPGA bridging high-bandwidth network
- Other service nodes



Elastic and Scalable System for High-PErformance Reconfigurable Computing

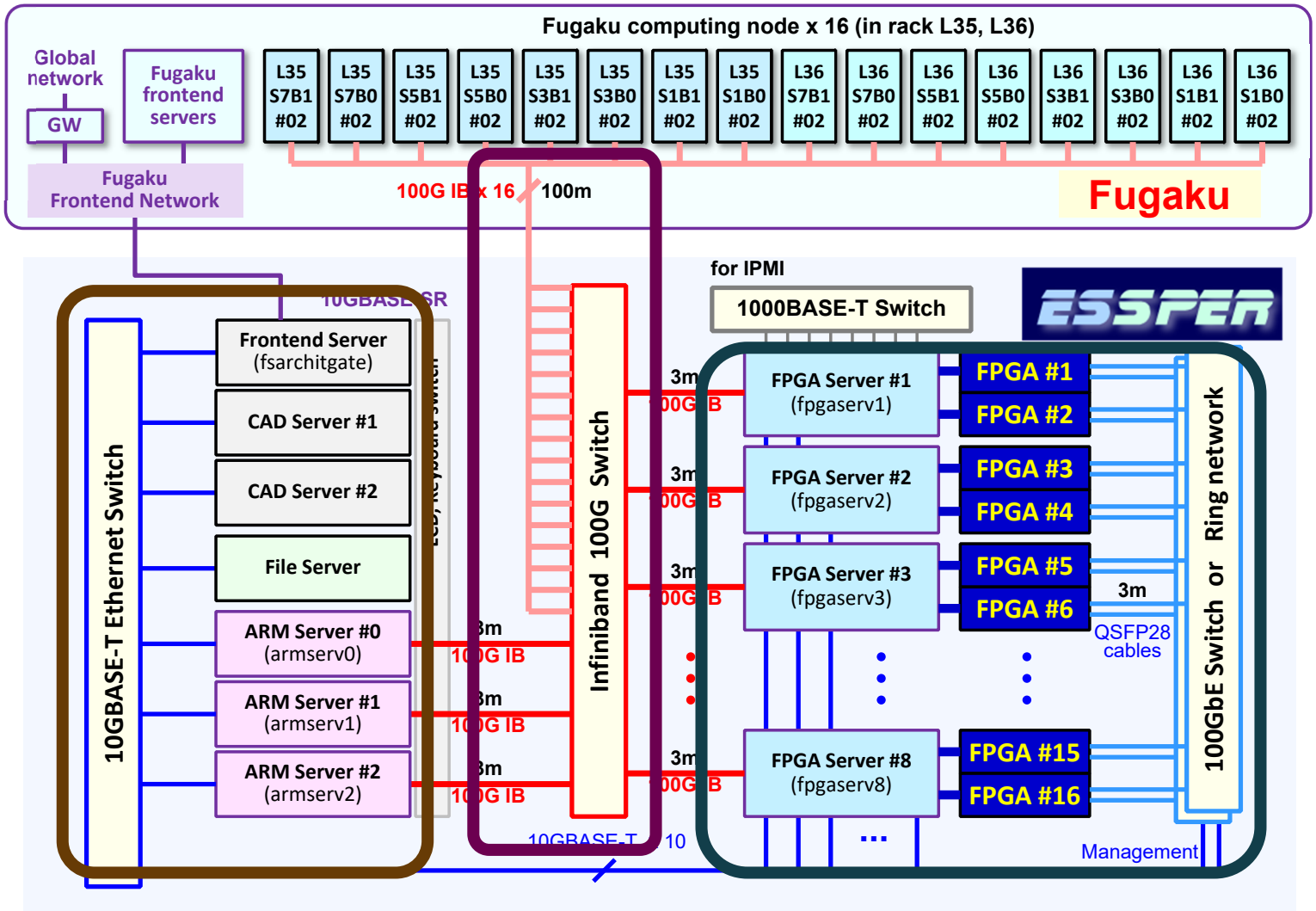
Experimental Prototype
to extend existing HPC machines with FPGAs





System Organization

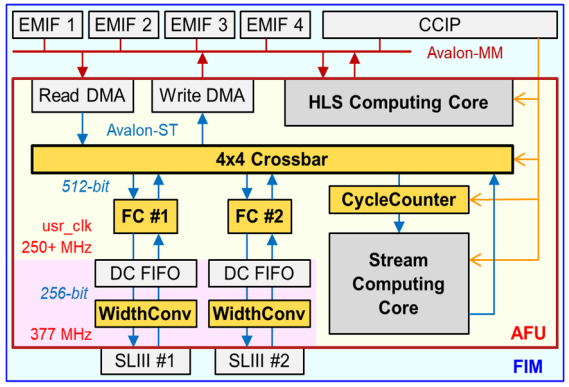
- FPGA Cluster
- CPU-FPGA bridging network
- Other servers

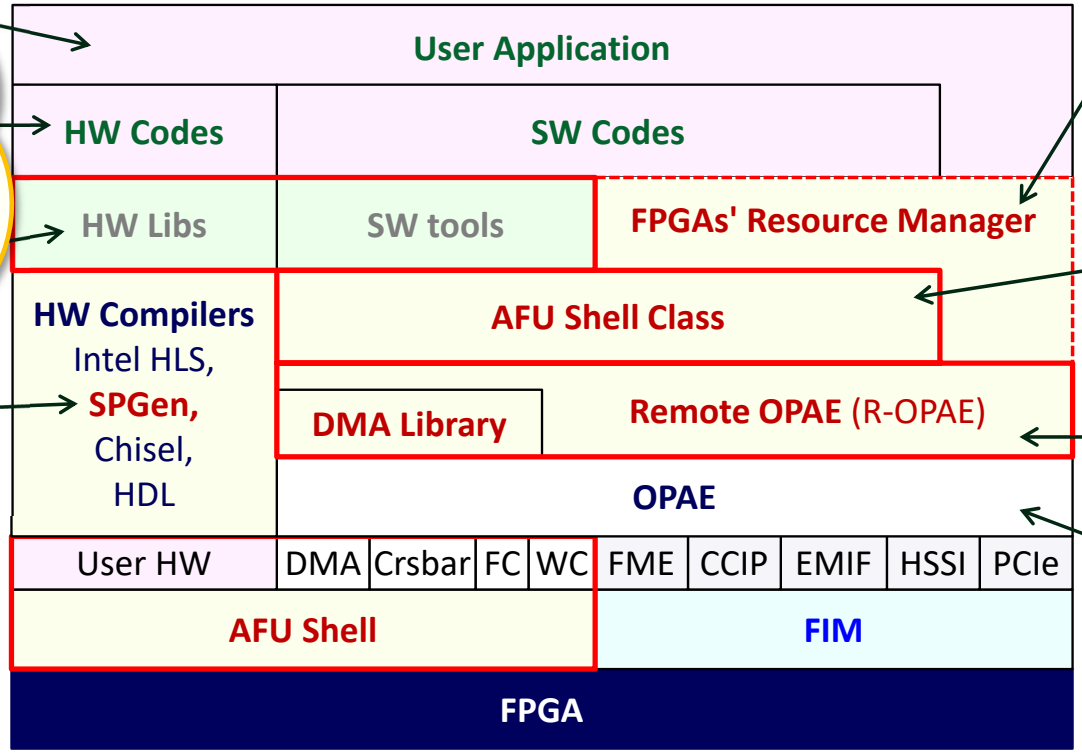


System Stack of ESSPER

Joint researches are expected:

- Applications
- Libraries for HW and SW
- Tools / system software
- Parallelization techniques with multi FPGAs





Resource manager

- Search and allocate resources of multiple FPGAs
- FPGA network management / control

AFU Shell class

- Object of AFU shell
- Abstraction of HW

Remote OPAE

- Software bridge using Infiniband Verbs

OPAE

- Low-level driver

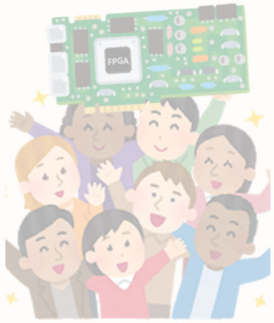
Proof of Concept

Req. 1 Interoperability w/ various HPC systems

- ✓ Able to easily extend existing systems with FPGAs
- ✓ Can we extend Supercomputer Fugaku?

Req.2 Flexibility for using FPGAs in a system

to flexibly
stem
achine
e users
ion of FPGAs



Software-bridged FPGA driver

FPGA resource manager

Req.3 Platform for sufficient customizability

- ✓ Able

inte

Req.4 Techniques for performance scalability

- ✓ All

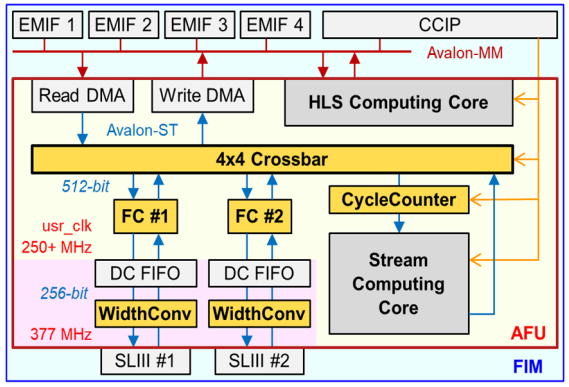
FPGA Shell (SoC)

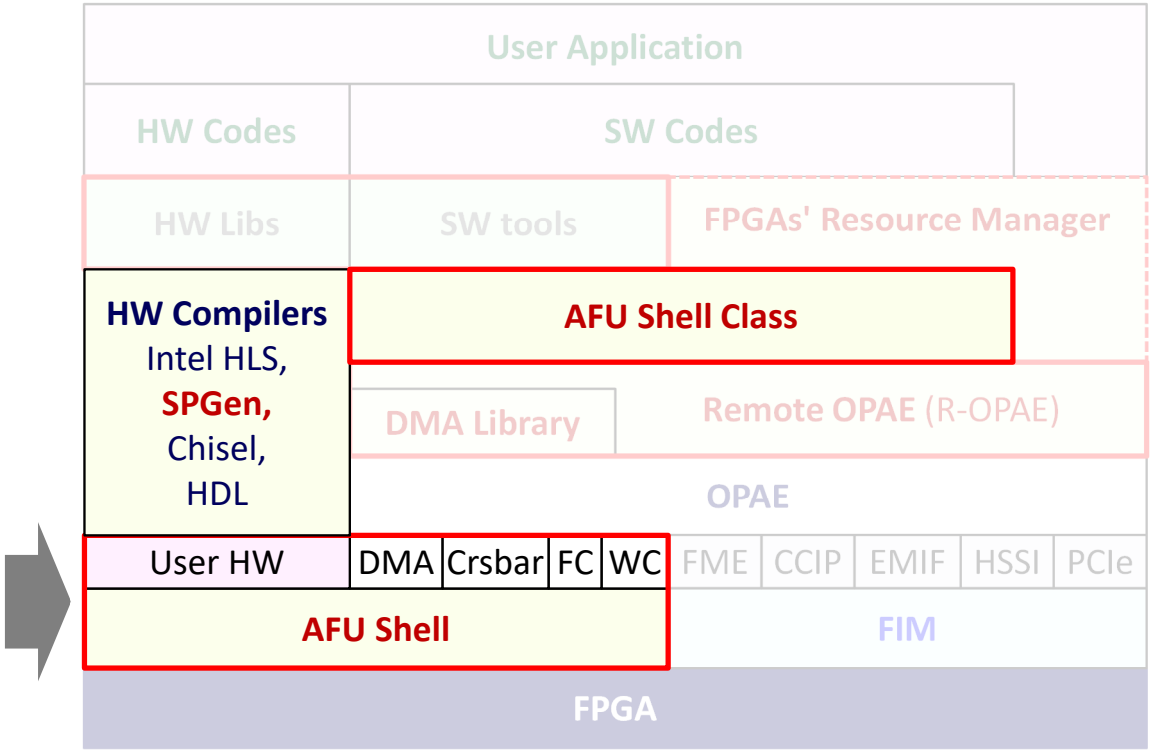
HLS-based programming flow, & FPGA object lib

Inter-FPGA network

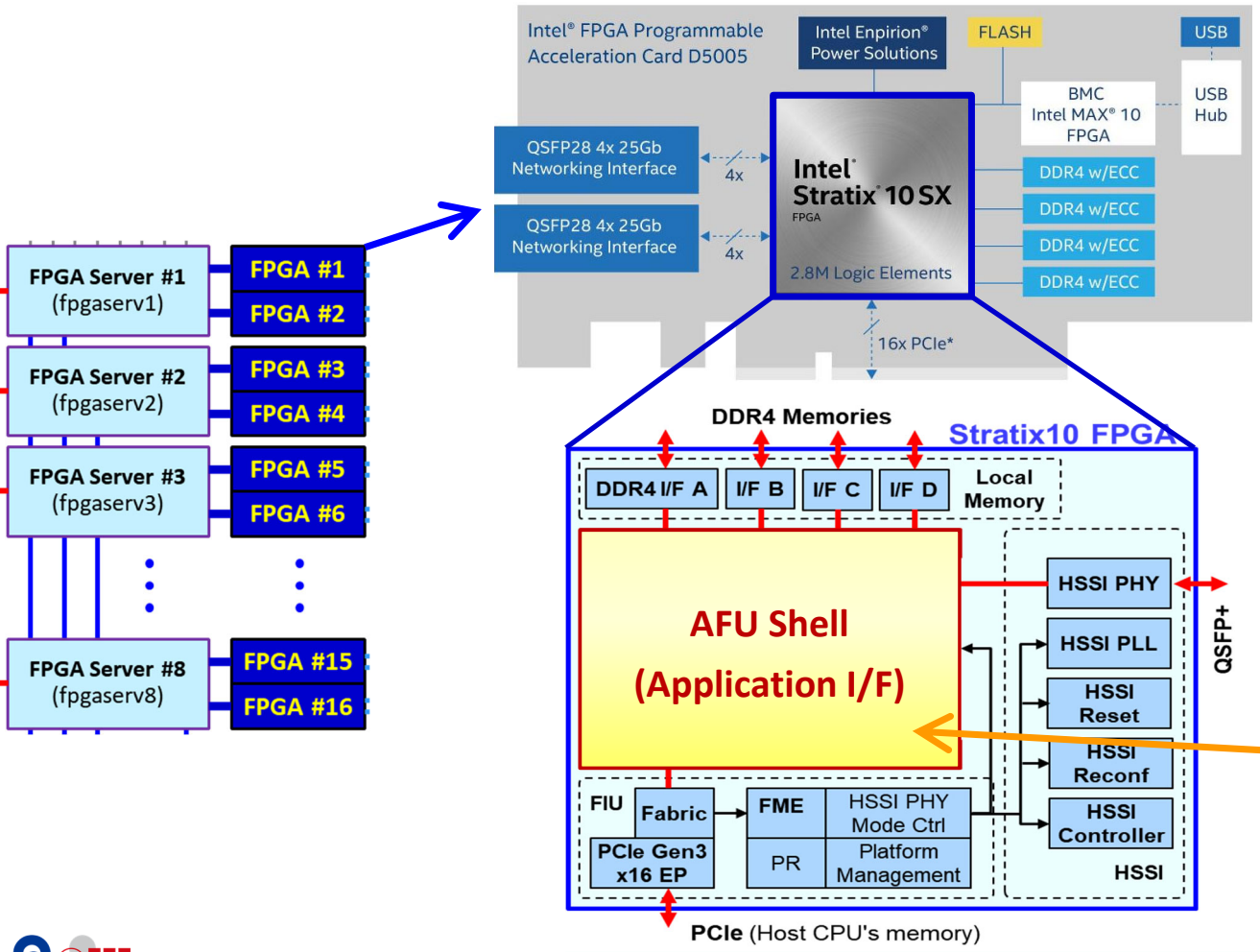
Virtual circuit switching over Ethernet

System Stack of ESSPER





Design of FPGA System-on-Chip



Intel FPGA PAC D5005

- ✓ *Intel Stratix 10 FPGA (14nm)*
- ✓ *2753K LEs, 229 Mb BRAMs*
- ✓ *5760 FP DSPs (7TF @ 600MHz)*
- ✓ 8GB DDR4 x 4ch
- ✓ PCIe Gen3 x16
- ✓ 2x QSFP28 (100Gb/s)

FIM (FPGA Interface Manager)

- ✓ Fixed region including I/F

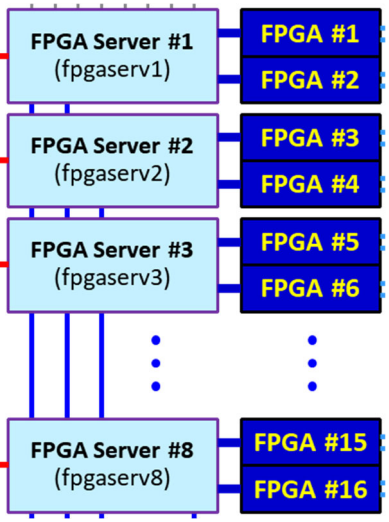
AFU (Acceleration Function Unit)

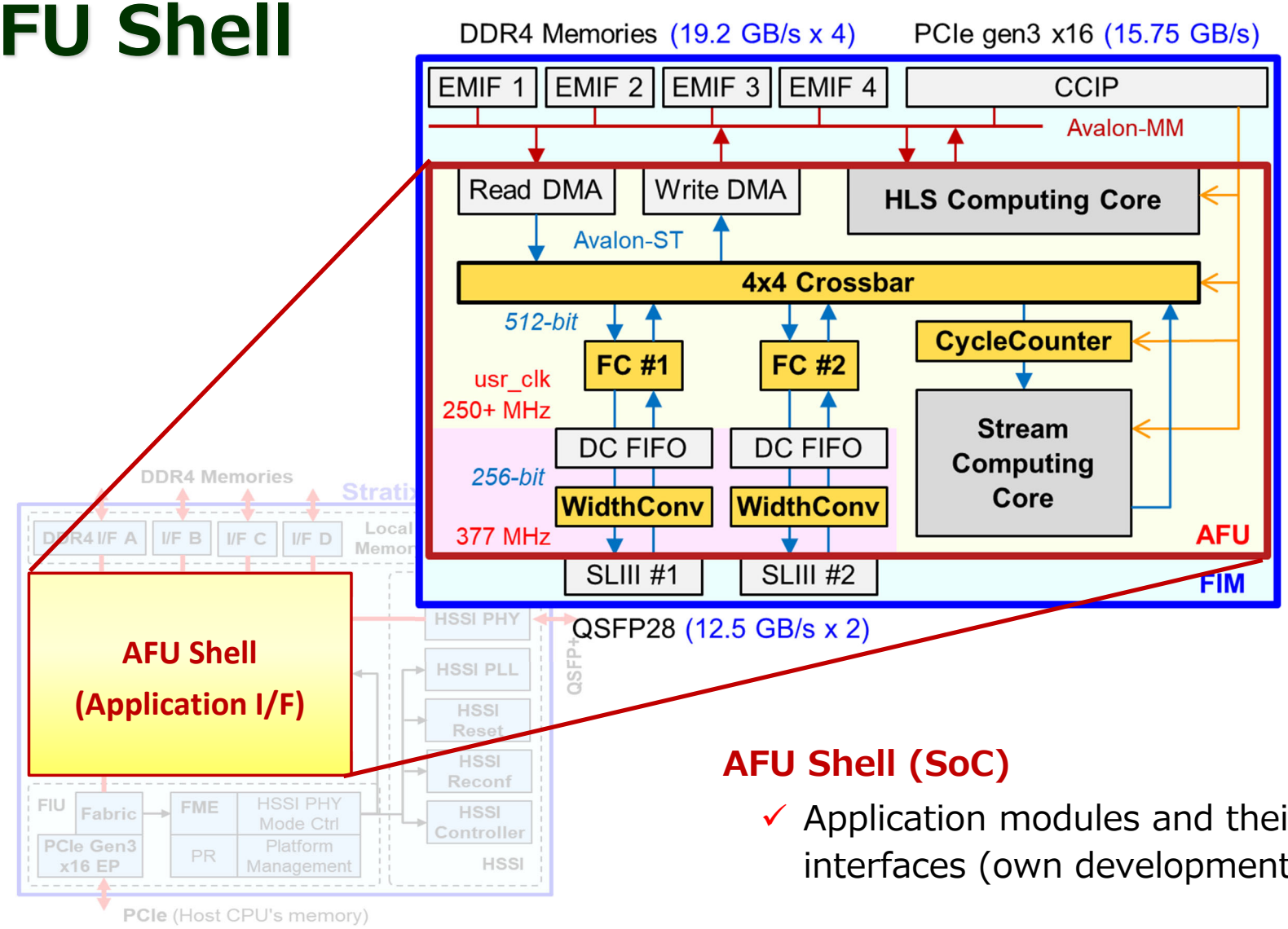
- ✓ Reconfigurable region

AFU Shell (SoC)

- ✓ Application modules and their interfaces (own development)

Design of AFU Shell





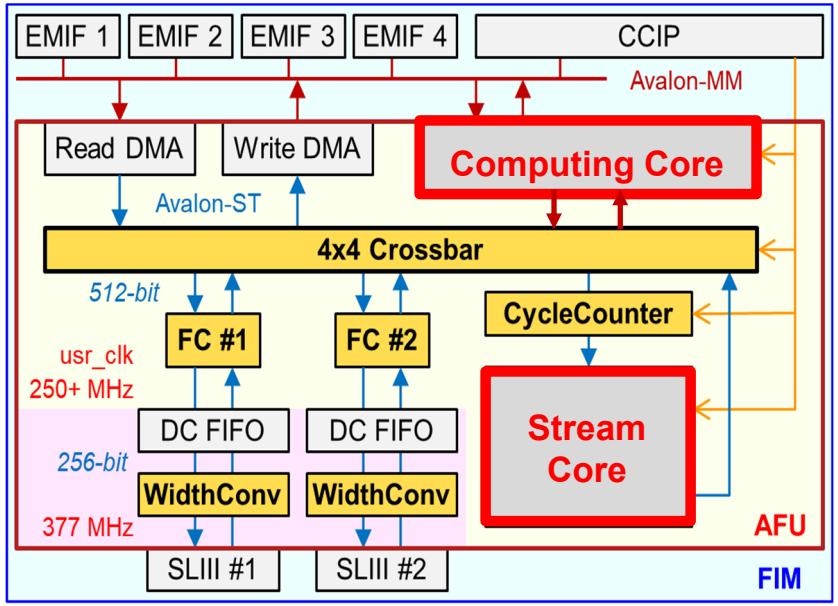
AFU Shell (SoC)

- ✓ Application modules and their interfaces (own development)

Programming for Computation

Implement you comp. core and embed it!

- ✓ **Computing core** — Connected to DDR4 memories. read and write data by itself.
- ✓ **Stream core** — Connected to crossbar. compute with data stream.



How to program cores

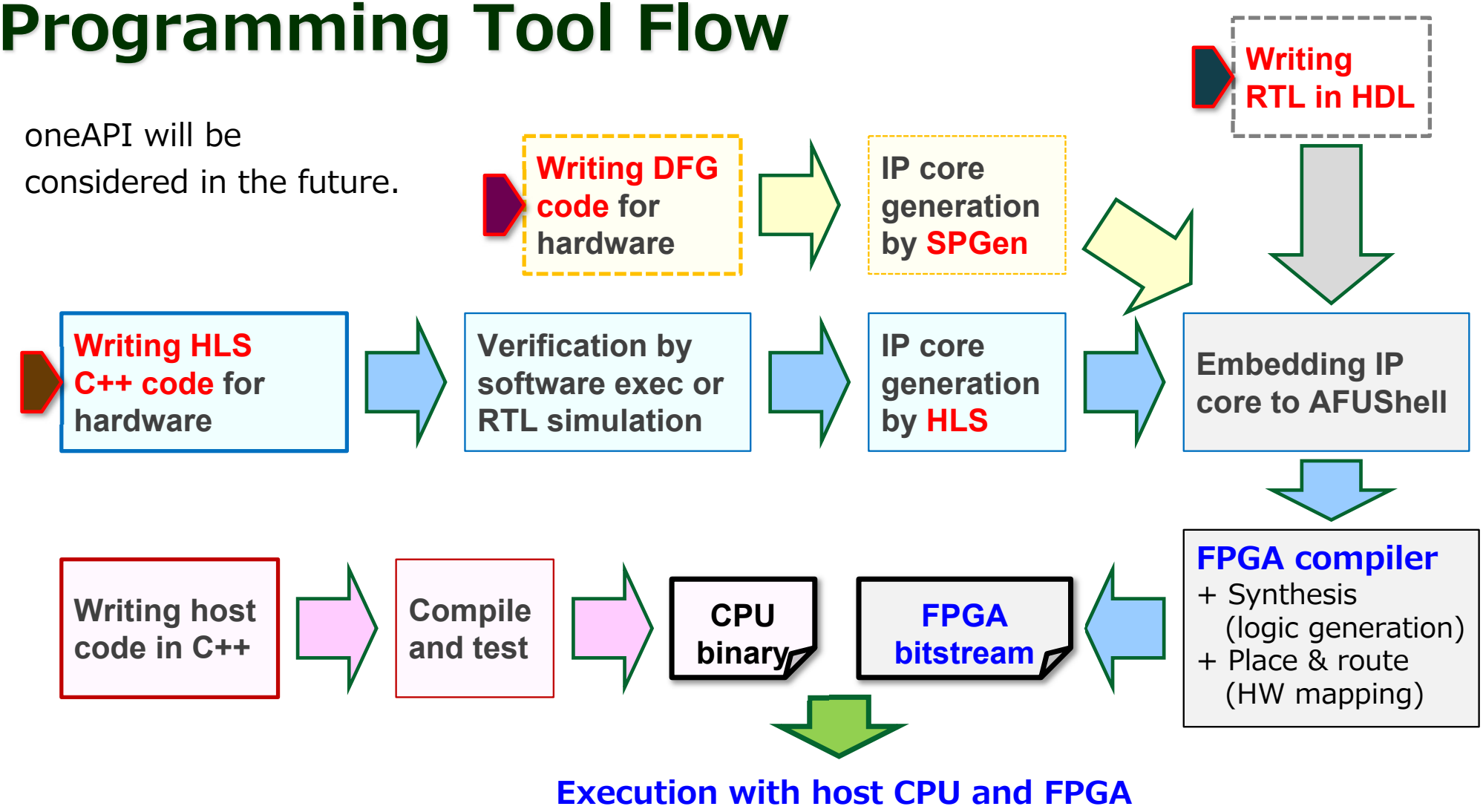
- ✓ **Software-oriented** **HLS** **Describe algorithms in C/C++** (Intel HLS)
- ✓ **Hardware oriented** **HDL** **Describe hardware structure & FSM** (Verilog-HDL, VHDL, Chisel, etc.)
- ✓ **Others** **DSL** **Domain-specific langs for HW generation** (Stream processor generator : SPGen)

Low-level, but more flexible than OpenCL and its BSP. Mem IF and network are customizable.

HLS: High-level synthesis, **Chisel**: Scala-based language for RTL, **SPGen** : Stream processor generator

Programming Tool Flow

oneAPI will be considered in the future.



Host Code using "afushell_class"

HW details are abstracted.

- ✓ Addresses of modules
- ✓ Interface of service functions
- ✓ **Low-level control** still possible

App code is written simply.

- ✓ Instantiate **AFUShell object**
- ✓ Open object
- ✓ Use modules / services
  - ➢ *Crossbar*
  - ➢ *Hardware cycle-counter*
  - ➢ *DMA (Host-FPGA, FPGA-FPGA)*
  - ➢ *Computing core*
- ✓ Close object

```
int very_simple_example(void)
{
  uint32_t allCycles, validCycles, csr;
  uint64_t bytes = 1024*1024*64;
  char *begin_ptr = (char *)malloc(sizeof(char)*bytes);

  afush_class afush("AFUSH0", "AFUSH0:");   <- Instantiate object

  if (!afush.open(0))   <- Open device
    {
      cout << "+ " << afush.name << " was not opened. Abort\n";
      return 0;
    }

  afush.set_crossbar(CROSSBAR_RdmaSl3a_Sl3b2CompWdma, cout);   <- Set Crossbar
  afush.read_crossbar(cout);

  // Blocking DMA transfers
  afush.dmaTransfer((uint64_t)begin_ptr, 0x800000000, bytes, HOST_TO_FPGA);

  afush.reset_ccounter(cout);   <- Use cycle-counter
  afush.dmaTransfer(0x00000000, 0x200000000, bytes, FPGA_TO_FPGA);   DMA Transfer
  afush.read_ccounter(allCycles, validCycles, csr, cout);

  afush.dmaTransfer(0x00000000, (uint64_t)begin_ptr, bytes, FPGA_TO_HOST);

  // Read and write a csr of your module
  cout << "== " << afush.mod[afush::ENTIRE_SPACE] << "\n"; // See memory map of "
  uint32_t val1 = 0x1234ABCD, val2;                        // "int" is NG.
  afush.mod[afush::ENTIRE_SPACE].writeMMIO32(0x00000340, val1); // crossbar write
  afush.mod[afush::ENTIRE_SPACE].readMMIO32 (0x00000340, val2); // crossbar read

  afush.close(cout);   <- Close device
  free(begin_ptr);

  return 1;
}
```

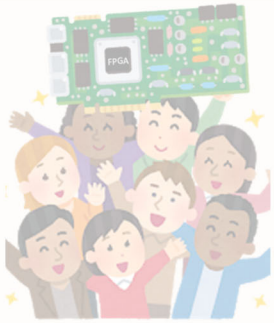
Proof of Concept

Req. 1 Interoperability w/ various HPC systems

- ✓ Able to easily extend existing systems with FPGAs
- ✓ Can we extend Supercomputer Fugaku?

Req.2 Flexibility for using FPGAs in a system

to flexibly
stem
achine
e users
ion of FPGAs



Software-bridged FPGA driver

FPGA resource manager

Req.3 Platform for sufficient customizability

✓ Able

inte

FPGA Shell (SoC)

HLS-based programming flow, & FPGA object lib

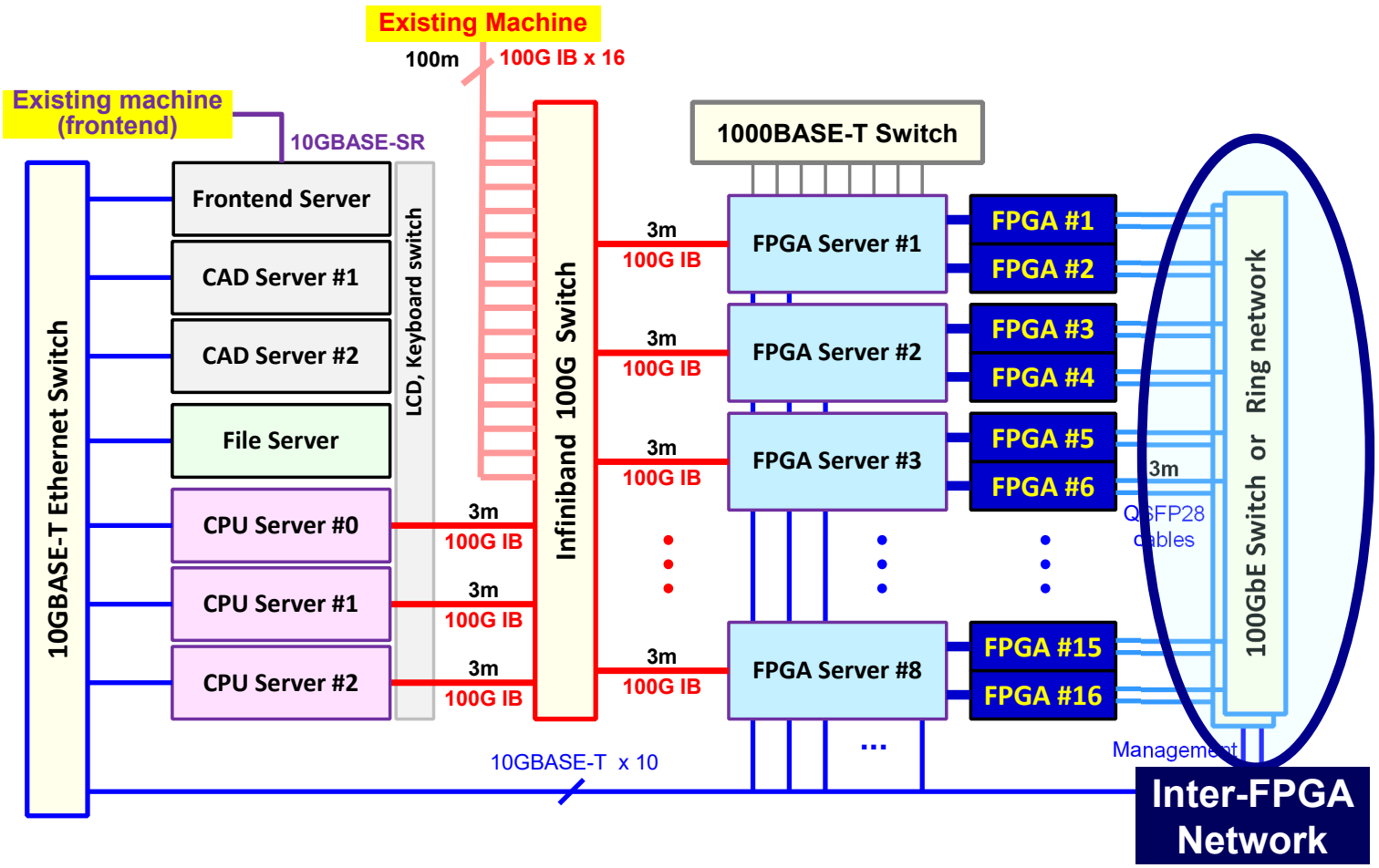
Req.4 Techniques for performance scalability

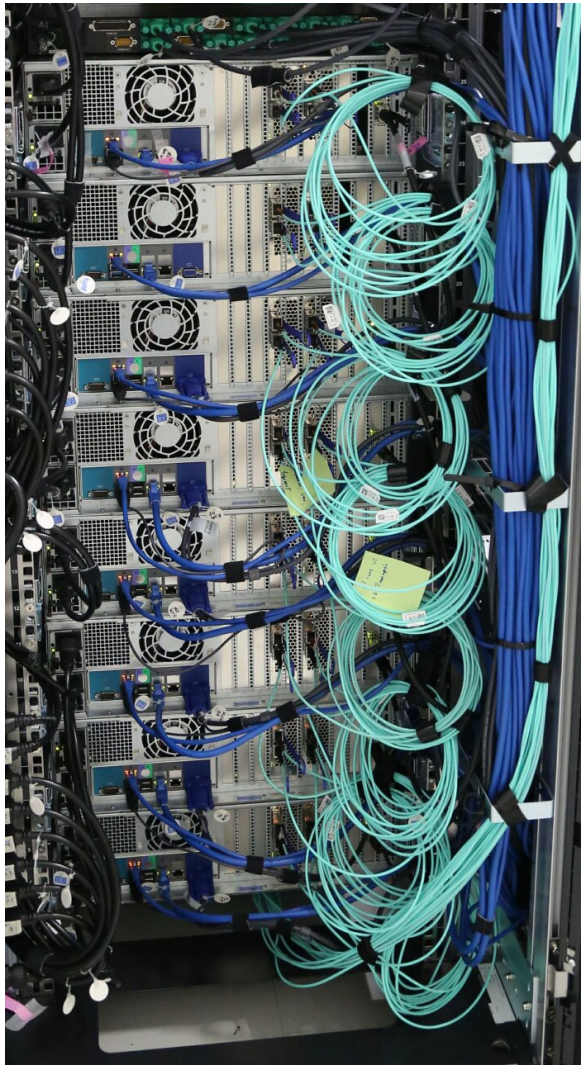
✓ All

Inter-FPGA network

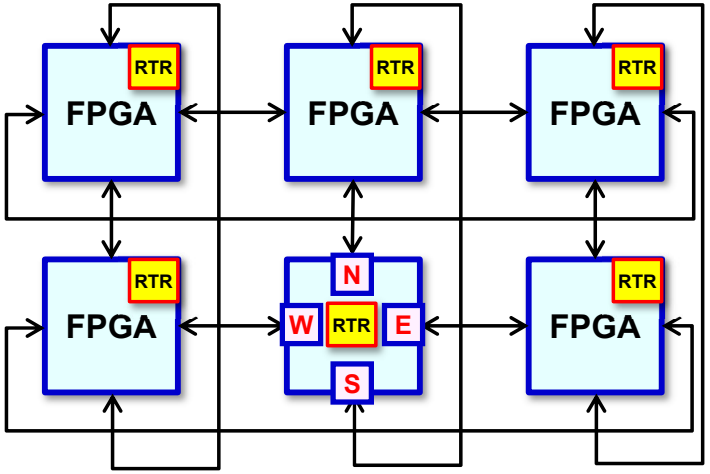
Virtual circuit switching over Ethernet

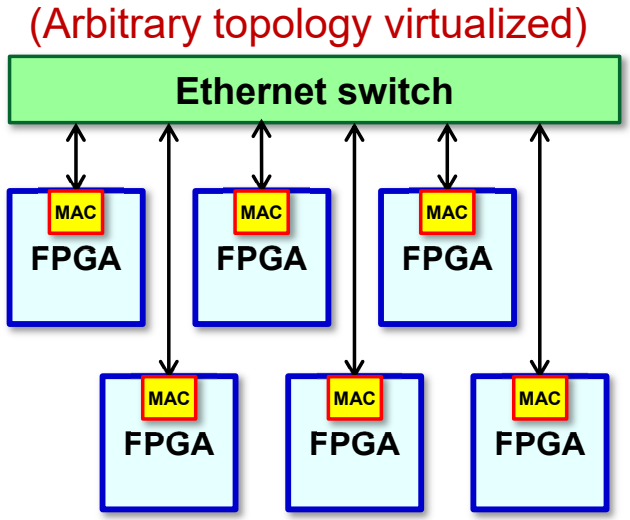
Inter-FPGA Networks





Two Types of Candidate Networks





Direct network (p2p)

Pros) Smaller overhead (lower/fixed latency), easy to use

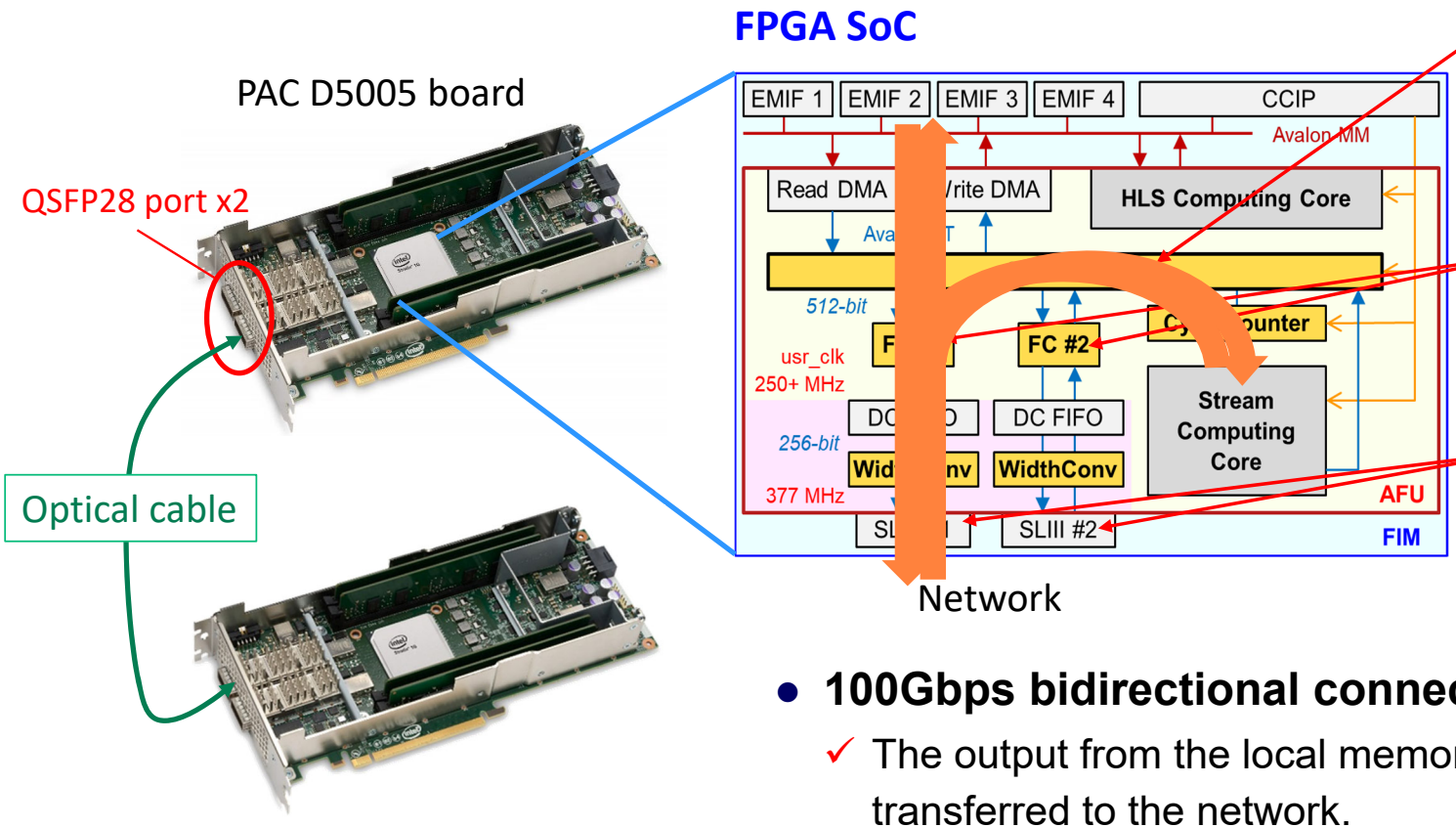
Cons) Inflexibility of resource allocation, more consumption of HW resources, difficulty to catch up

Indirect network (100G Ethernet)

Pros) Flexibility of resource allocation, easy adoption of cutting-edge tech

Cons) Overhead of ethernet frames (higher and variable latency), difficulty in flow-control and use, cost of expensive switches

Direct Connection Network (DCN)



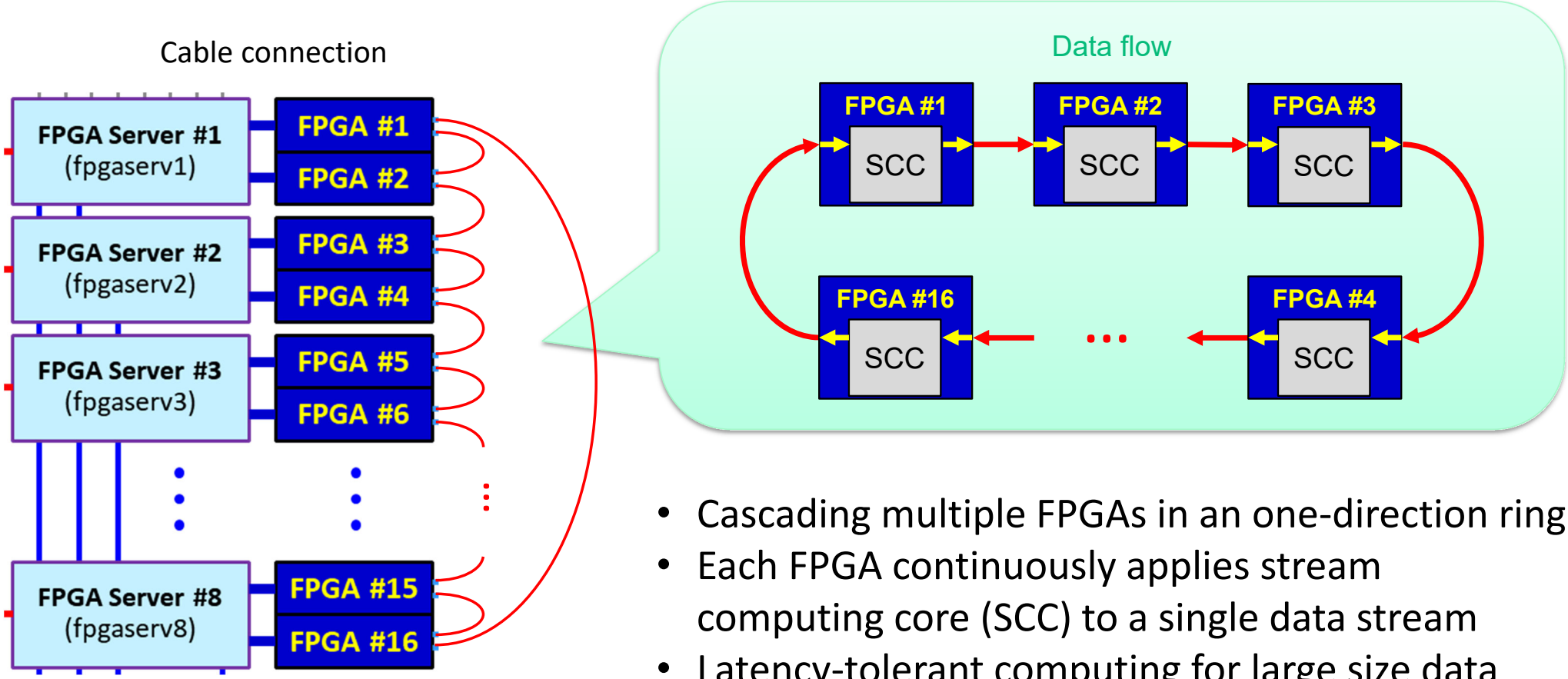
Control data routing among local memory, computing core, and network

Providing back pressure signal to prevent data lost in the link

100Gbps serial transceiver IP

- **100Gbps bidirectional connection x 2 on each board**
  - ✓ The output from the local memory and compute core can be directly transferred to the network.
  - ✓ Received data can also be input to the compute core or written to memory.

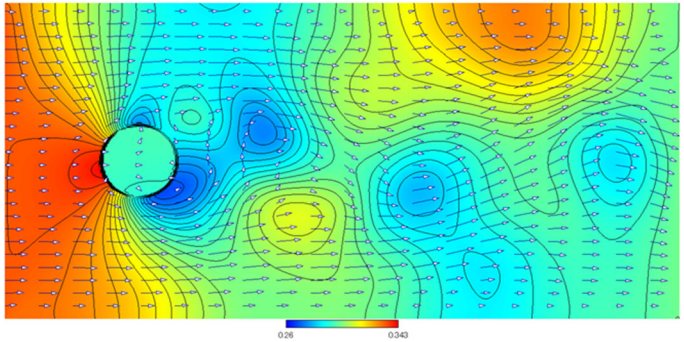
Stream Computing with a Ring Network

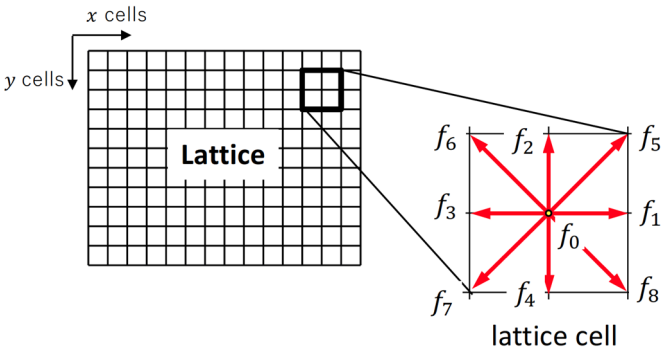


- Cascading multiple FPGAs in an one-direction ring
- Each FPGA continuously applies stream computing core (SCC) to a single data stream
- Latency-tolerant computing for large size data

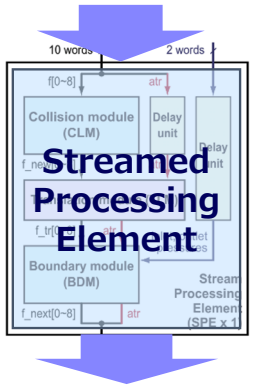
Example of Stream Computing

2D Lattice Boltzmann Method (LBM)





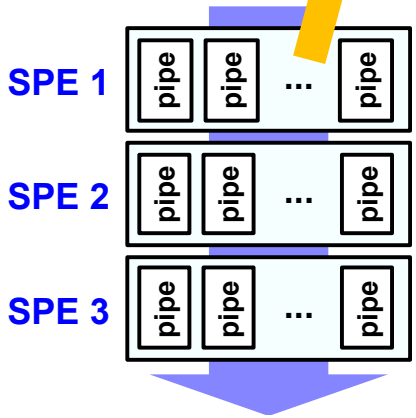
input streams (6.4 GB/s)

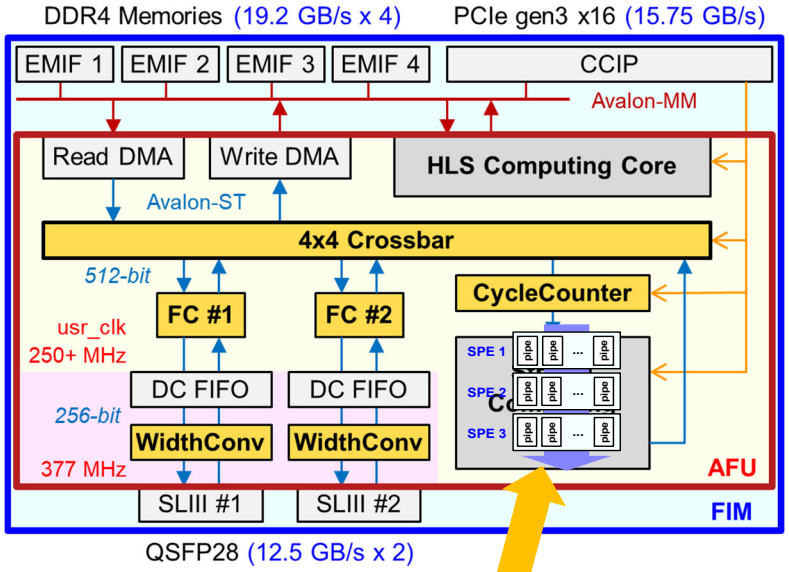


output streams

2D LBM SPE

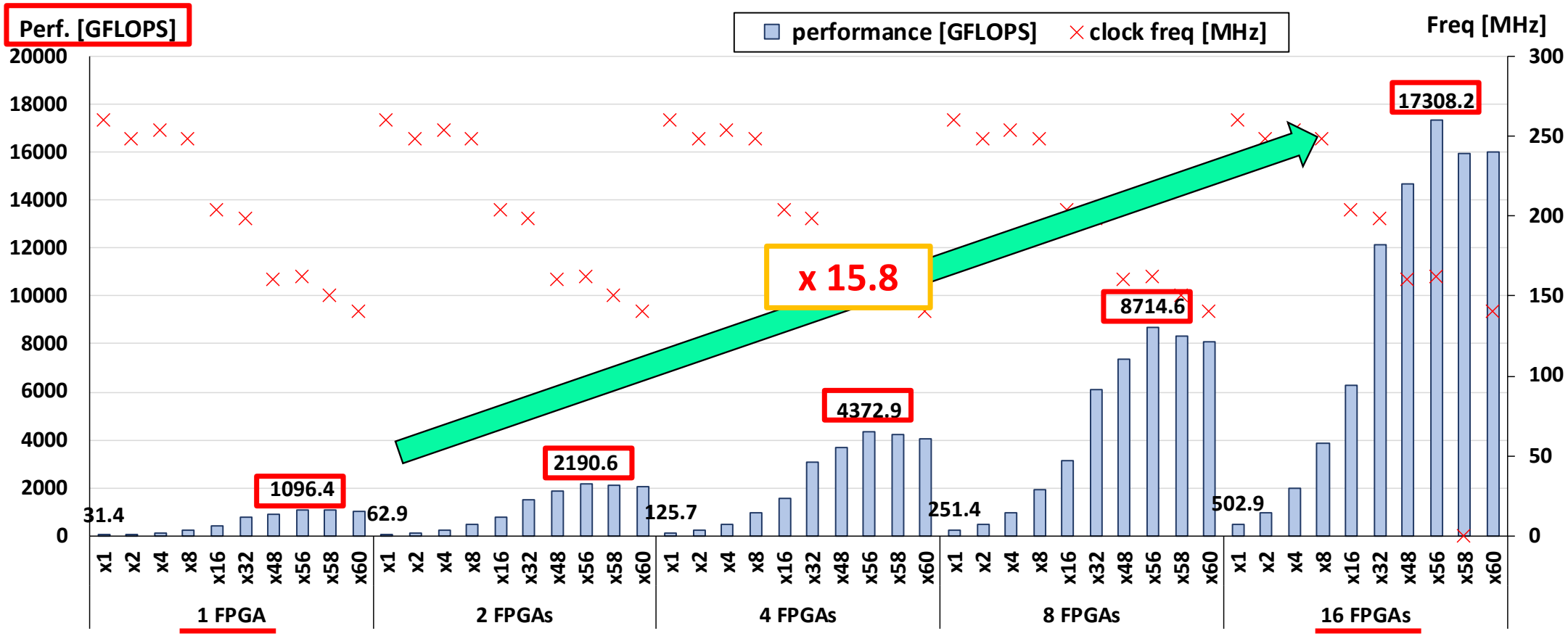
Cascading SPEs





Performance of 2D LBM with 100Gbps Ring NW

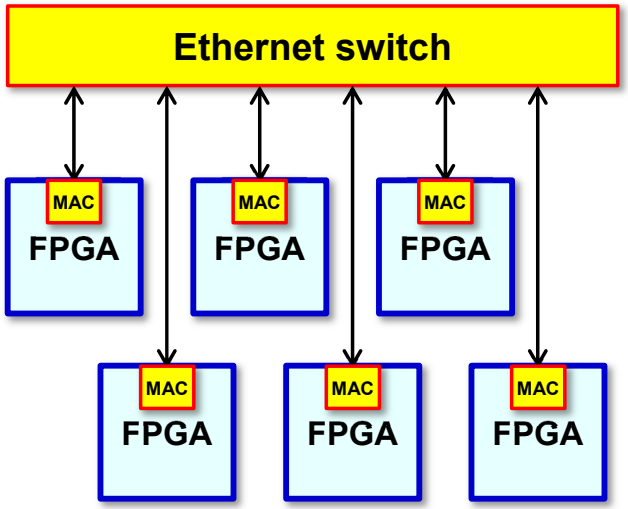
Computational performance (FLOPS) when processing about 2GB data

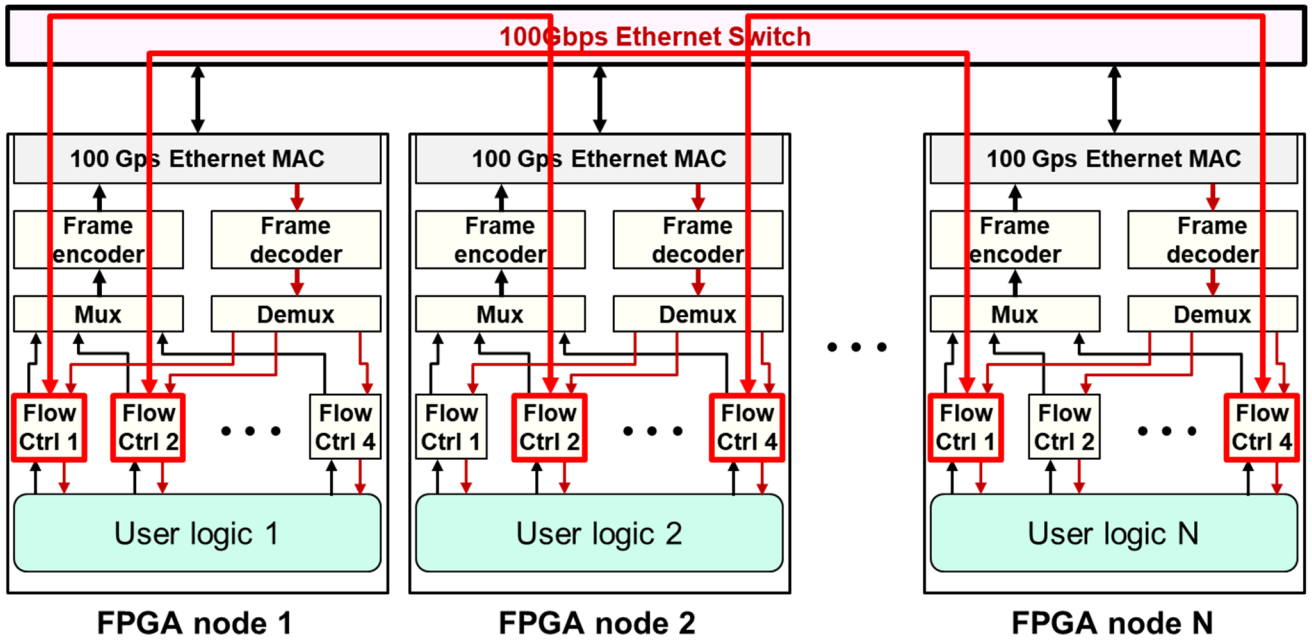


Virtual Circuit Switching Network (VCSN)

Arbitrary topology with virtual links between FPGAs over Ethernet

✓ User logic can simply send and receive data streams through virtual links.





100G Ethernet switches

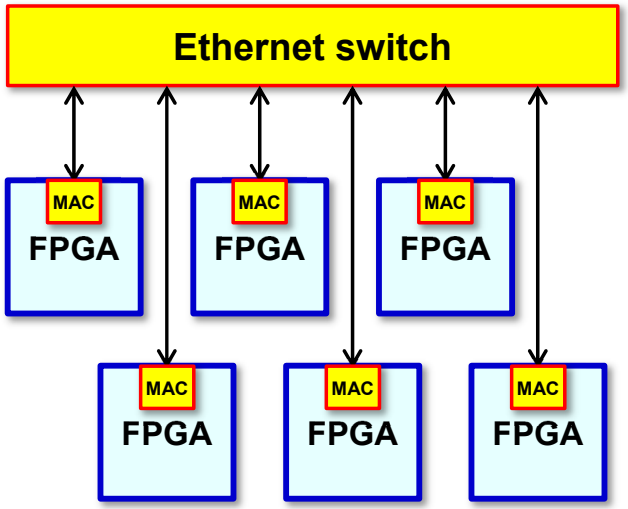
Pros) Flexibility, cutting-edge technology

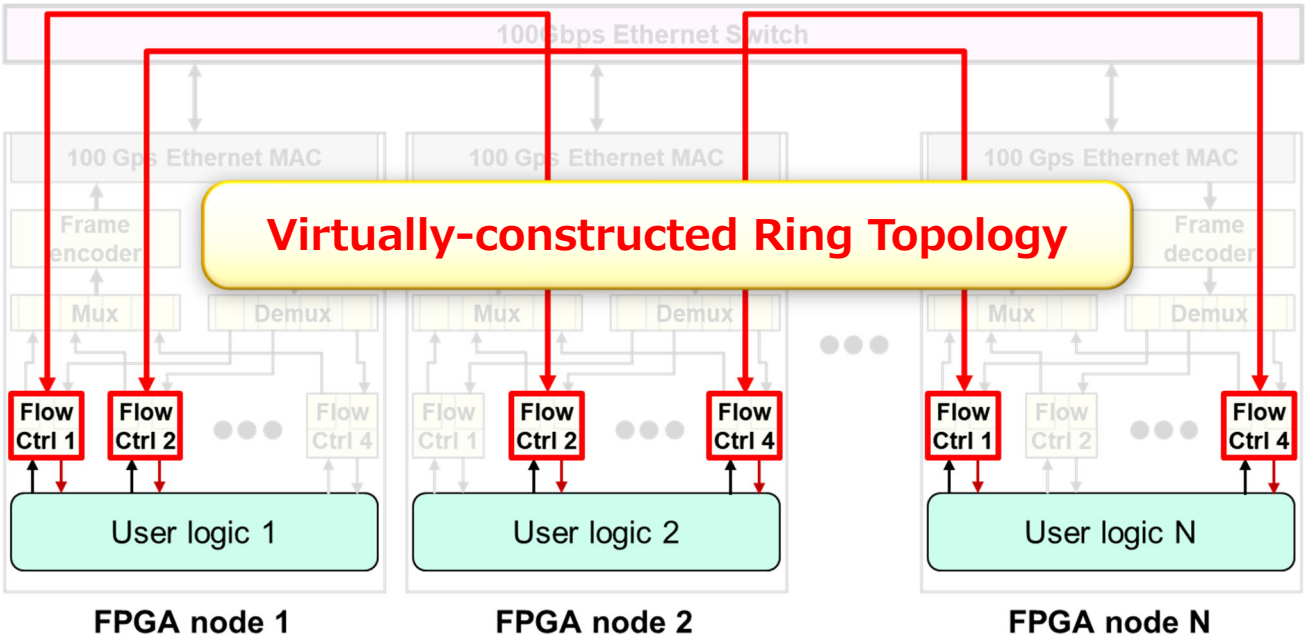
Cons) Overhead of ethernet frames, higher and variable latency, difficulty in flow-control and use

Virtual Circuit Switching Network (VCSN)

Arbitrary topology with virtual links between FPGAs over Ethernet

✓ User logic can simply send and receive data streams through virtual links.



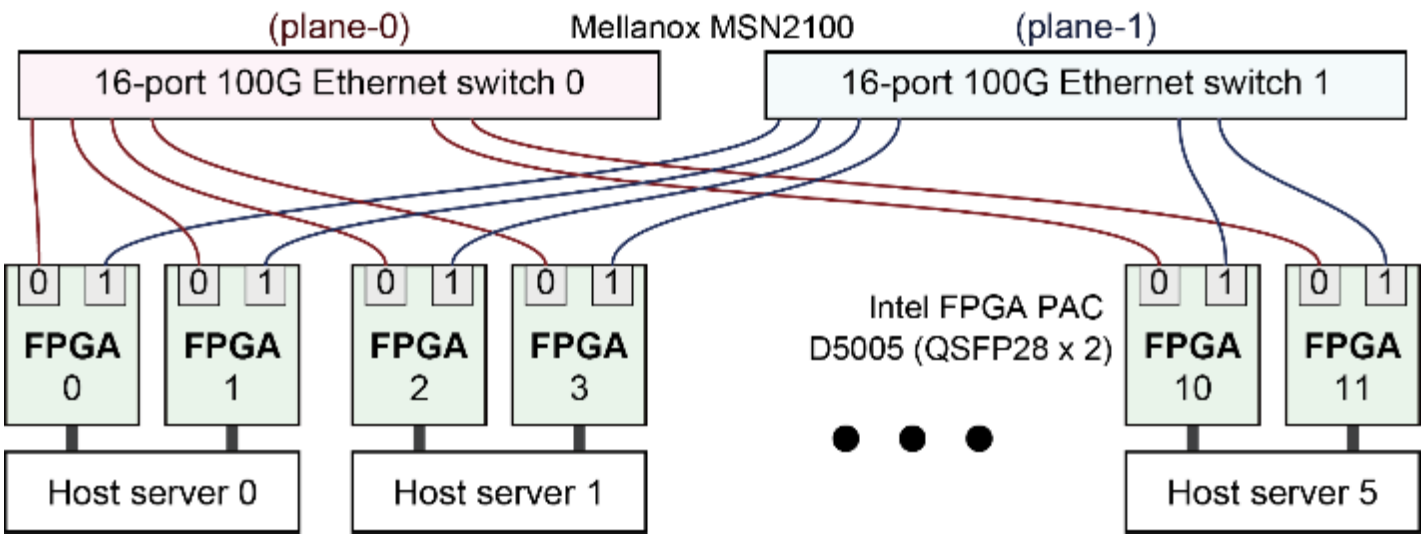


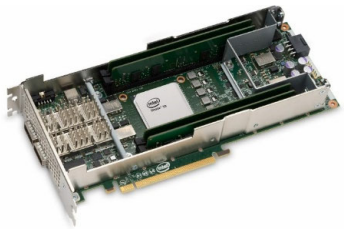
100G Ethernet switches

Pros) Flexibility, cutting-edge technology

Cons) Overhead of ethernet frames, higher and variable latency, difficulty in flow-control and use

Implementation with 100Gbps Ethernet SWs





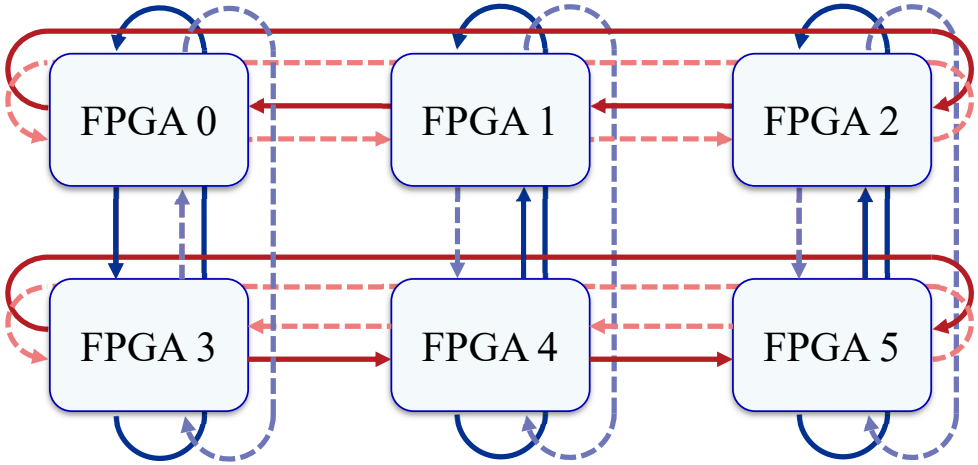
Intel PAC D5005 FPGA cluster with VCSN

- FPGA's two ports connected to a different switch (Dual Plane)
- Two 16-port 100G Ethernet switches and optical cables

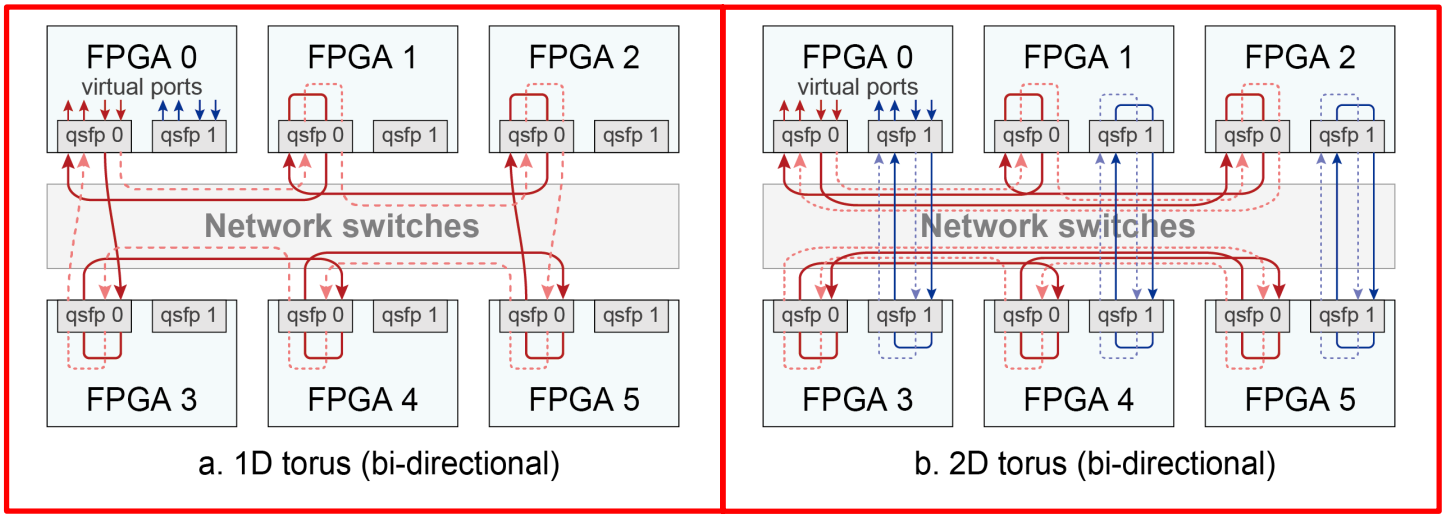
VCSN Configurations

6-FPGA networks

- 1-D torus
- 2-D torus (2x3)



Virtual topology of bi-dir 2D Torus



Preliminary Evaluation : Throughput

DCN v.s. VCSN

- ✓ DCN: Direct Connection Network
- ✓ VCSN: Virtual Circuit Switching Network

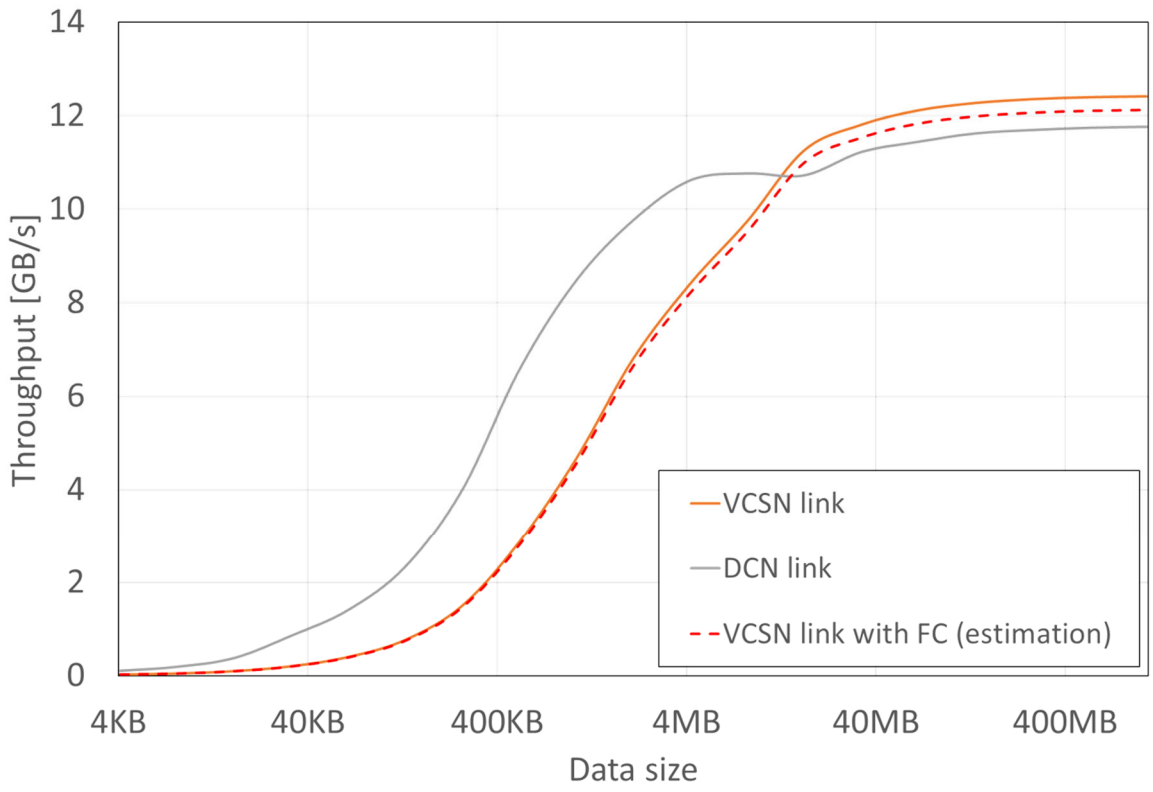
VCSN rises slowly due to higher latency.

- ✓ P2P latency of VCSN 851 ns
- ✓ P2P latency of DCN 490 ns

VCSN has higher Max throughput.

- ✓ Jumbo frame of Ethernet is more efficient.

Anyway, it works!



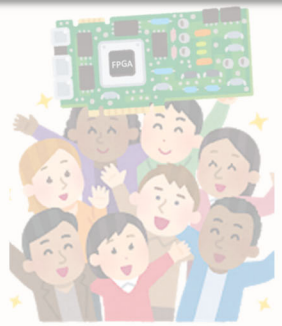
Proof of Concept

Req. 1 Interoperability w/ various HPC systems

- ✓ Able to easily extend existing systems with FPGAs
- ✓ Can we extend Supercomputer Fugaku?

Req.2 Flexibility for using FPGAs in a system

to flexibly
stem
achine
e users
ion of FPGAs



Software-bridged FPGA driver

FPGA resource manager

Req.3 Platform for sufficient customizability

- ✓ Able

inte

FPGA Shell (SoC)

HLS-based programming flow, & FPGA object lib

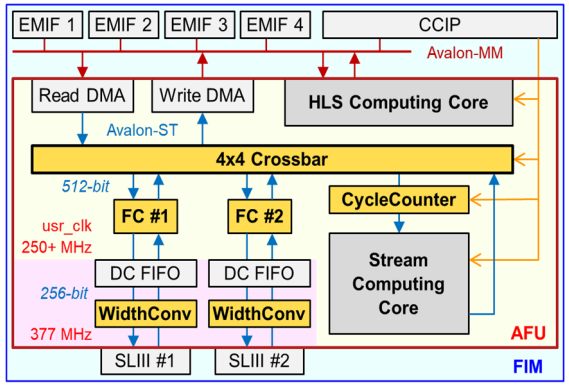
Req.4 Techniques for performance scalability

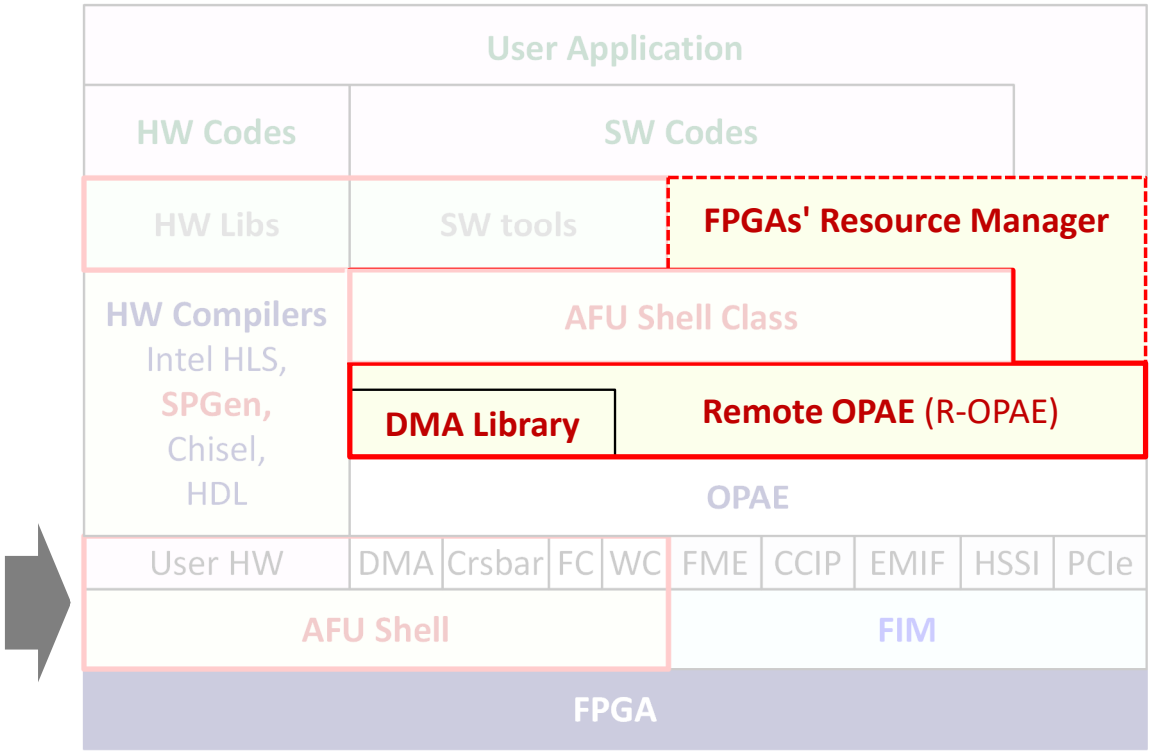
- ✓ All

Inter-FPGA network

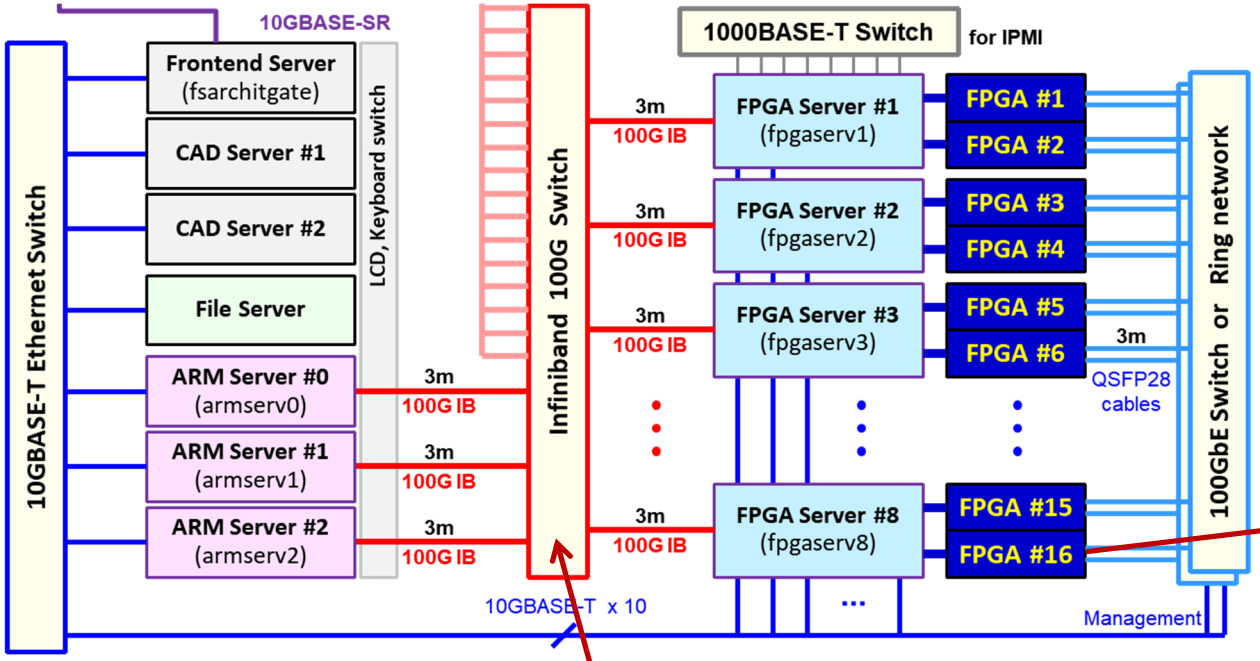
Virtual circuit switching over Ethernet

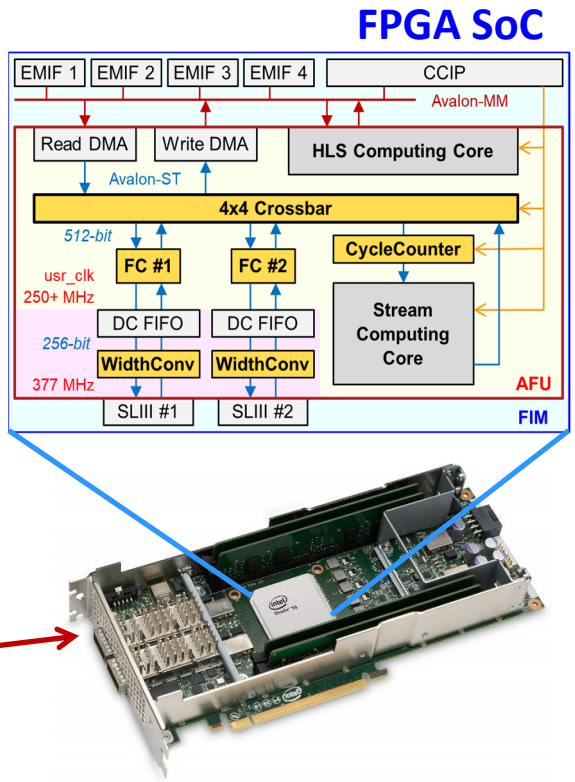
System Stack of ESSPER





Software-bridged Driver to Utilize Remote FPGAs





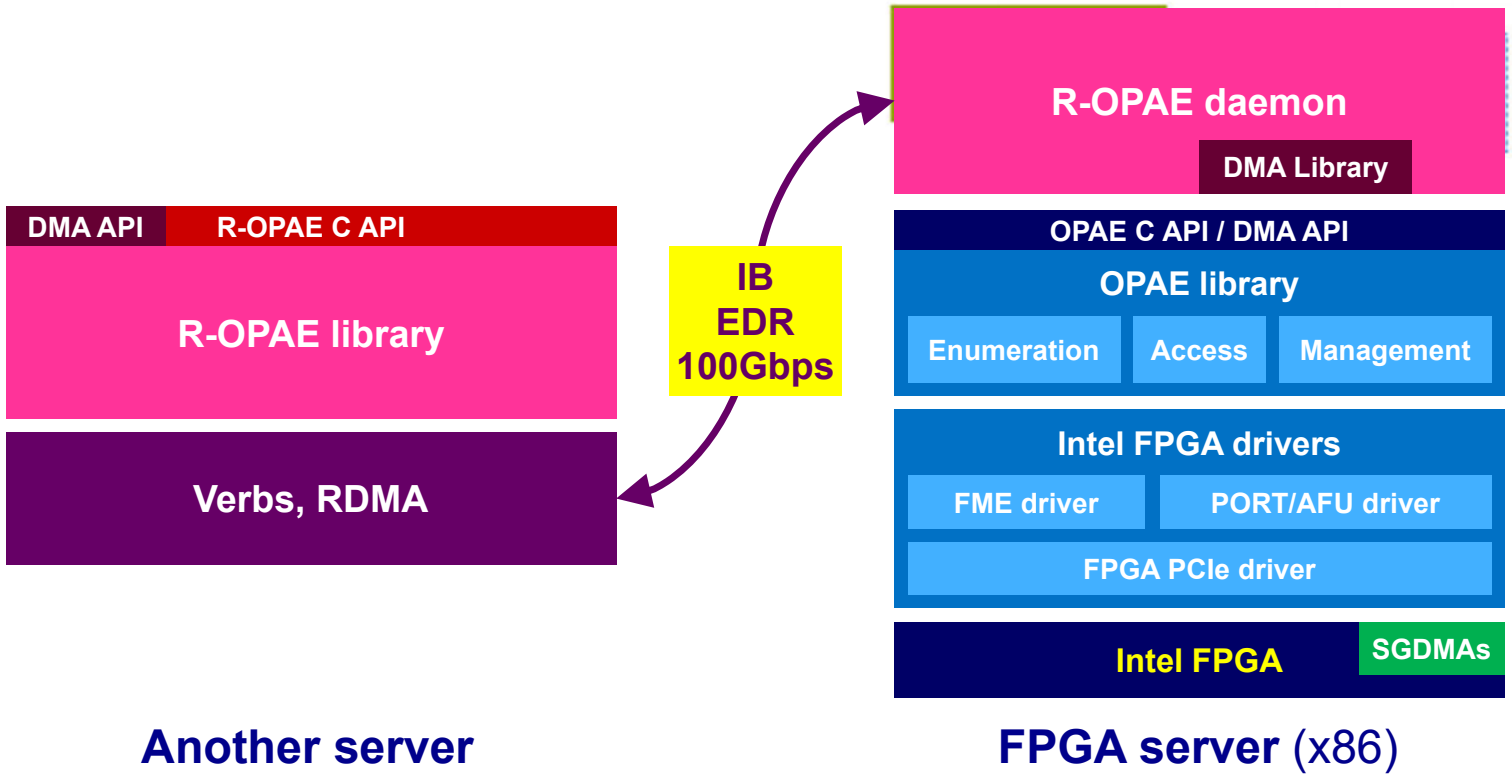
CPU - FPGA network

- 100G Infiniband
- Software-bridged driver (R-OPAE)

Remote-OPAE (for remote FPGA Access)

Software bridge for FPGAs over Infiniband

- ✓ **OPAE**: Open Programmable Acceleration Engine (PCIe FPGA driver)
- ✓ 99% of OPAE APIs are supported.
- ✓ We can use any FPGAs in a system via IB as if they were locally installed.



R-OPAE as Software-based Resource Disaggregation

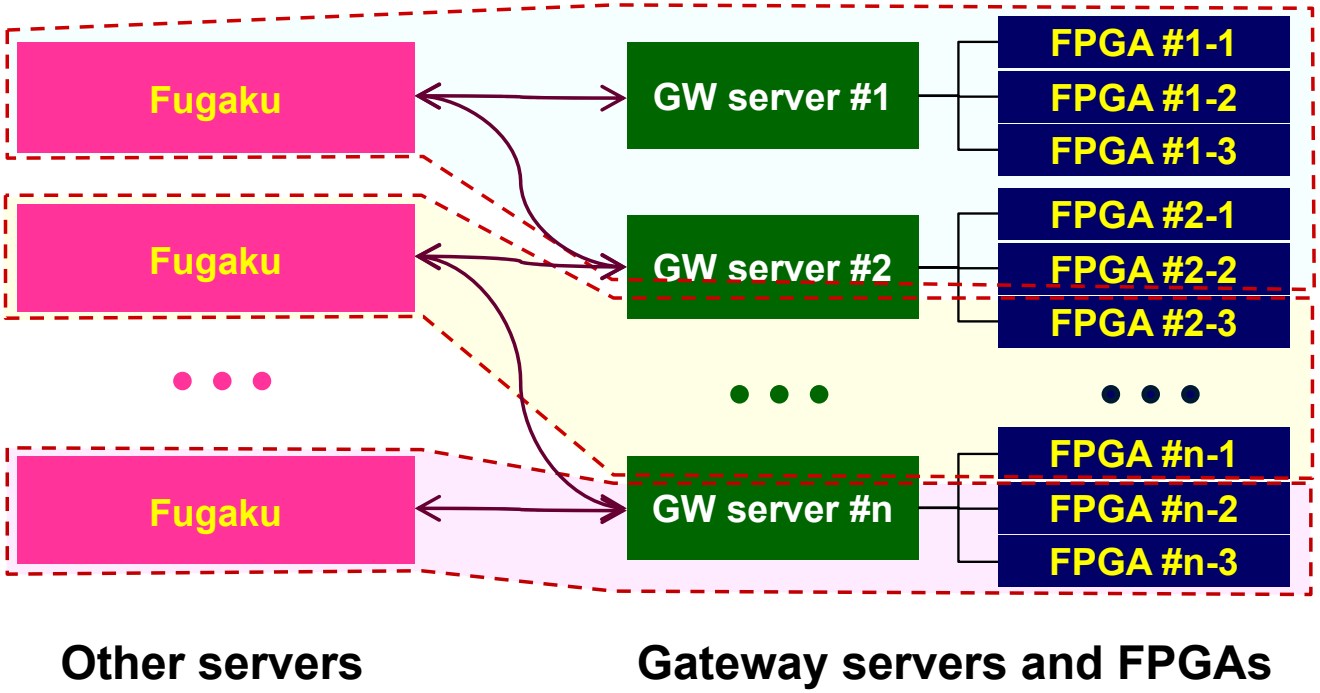
Transparent access to remote FPGAs

Flexible utilization:

- ✓ Can use any available FPGA resources

Inter-operability and extensibility:

- ✓ Vendor/ISA-independent
- ✓ Operable with various architectures such as Fugaku (ARM)

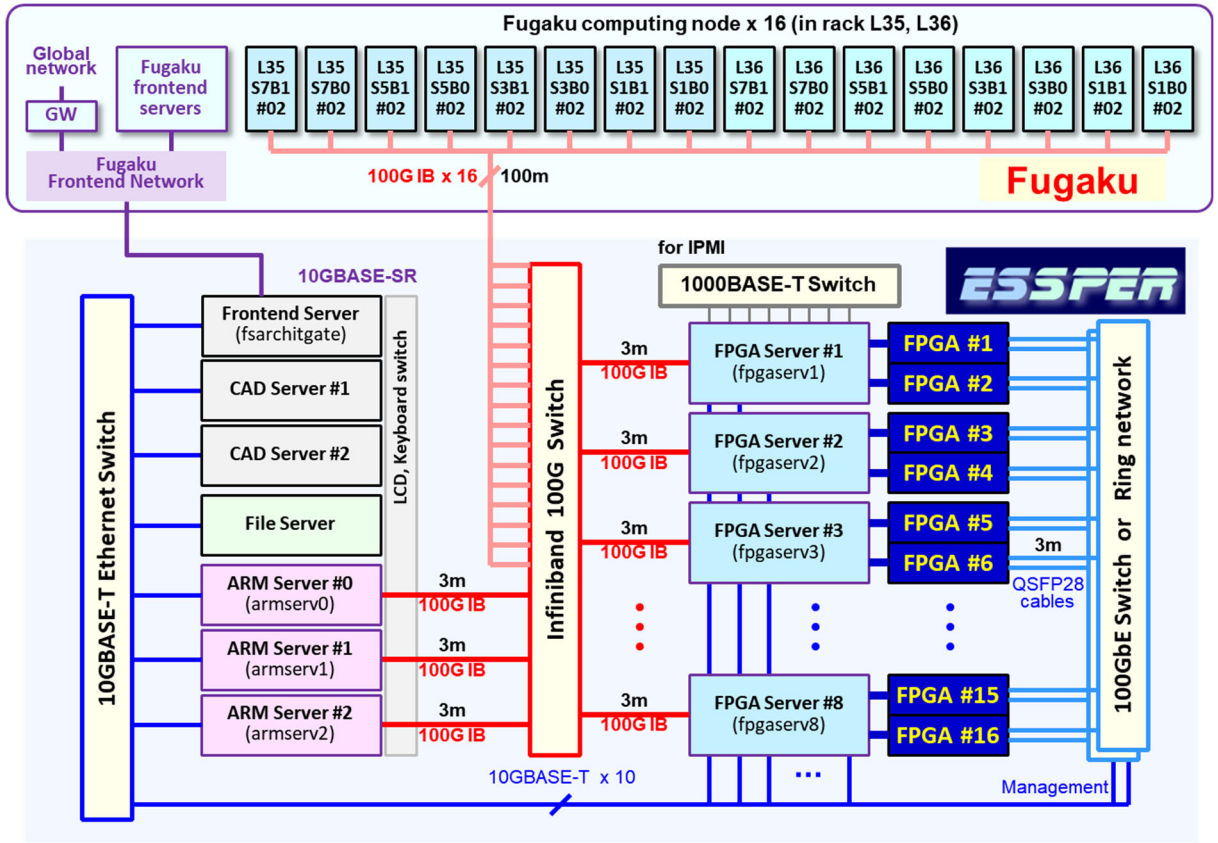


Implementation and Verification

Implementation of the System

Implemented & verified

- ✓ FPGA cluster and system stack
- ✓ Connected to Fugaku
- ✓ Under operation for research

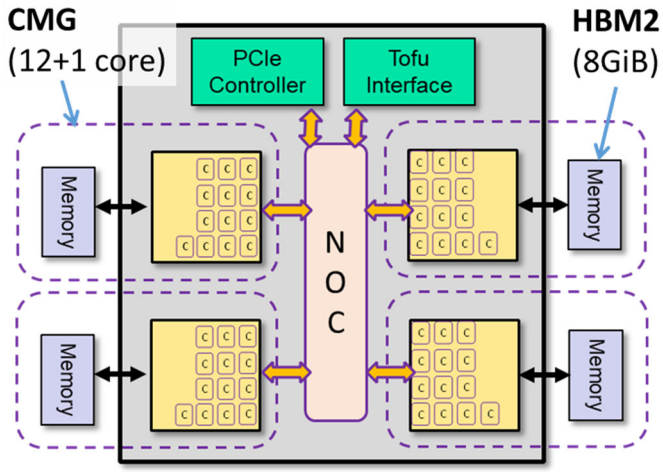


Verification and Performance Evaluation

Used machines

- ✓ FPGA server　Intel Xeon Gold5122 (3.6GHz, 4 cores) x2
- ✓ ARM server　Cavium ThunderX2 (2.0GHz, 28 cores) x2
- ✓ Fugaku node　Fujitsu A64FX (2.0GHz, 48+4 cores) x 1

R-OPAE & afushell class can successfully be compiled and linked in the environment of x86, ARM, Fugaku Linux.



Fugaku A64FX　(48+ core, NUMA)

Operations with OPAE and R-OPAE

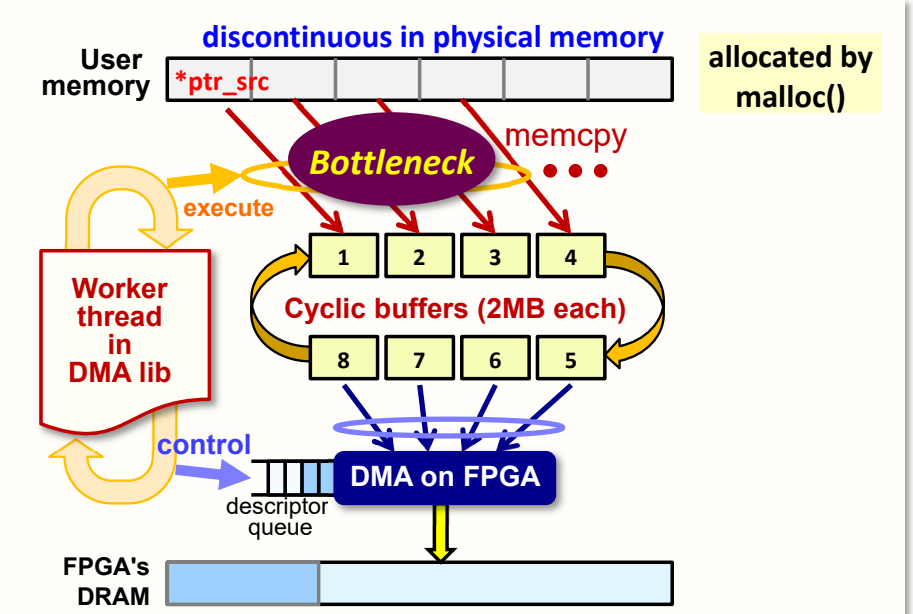
- ✓ To execute applications with FPGA
  - ➢ Partial reconfiguration of AFU (2+ sec)
  - ➢ Data transfer
  - ➢ Control

Procedures to execute application with FPGA

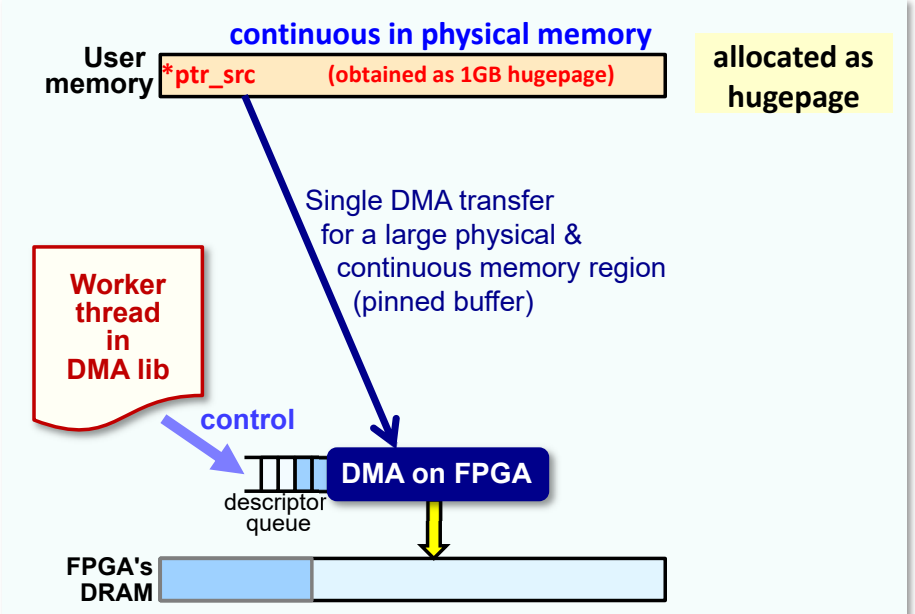
- ✓ Enumerate available FPGA resources
- ✓ Program a bit stream of AFUShell to FPGA
- ✓ Open AFU Shell device on FPGA
- ✓ **Upload data to DDR4 memory of FPGA**
- ✓ Run the computing core on FPGA
- ✓ **Download data from DDR4 memory of FPGA**
- ✓ Close AFU Shell device on FPGA

Local Data Transfer between CPU and FPGA

- **Two types of DMA transfer libraries were implemented.**



a) Host-to-FPGA DMA transfer using 2MB kernel buffers with memcpy()



b) Host-to-FPGA DMA transfer using 1GB pinned memory without memcpy()

Local Data Transfer by OPAE

PCIe Gen3 x16	16 GB/s
DDR4-2400	19.2 GB/s
100Gbps IB EDR	12.5 GB/s

Old DMA Lib

Pinned buffer

Pinned achieves 78% of PCIe throughput.

Pinned should be used for transfer-bound apps.

Different implementation of memcpy

Memcpy achieves 1/2～2/3 of the Pinned version.

a) Host-to-FPGA DMA transfer by OPAE with various memcpy implementation

b) FPGA-to-Host DMA transfer by OPAE with various memcpy implementation

Data Transfer by OPAE or R-OPAE

PCIe Gen3 x16	16 GB/s
DDR4-2400	19.2 GB/s
100Gbps IB EDR	12.5 GB/s

OPAE, Pinned

The throughput by R-OPAE over IB is equivalent to local transfer by memcpy.

OPAE, sse2 cpy

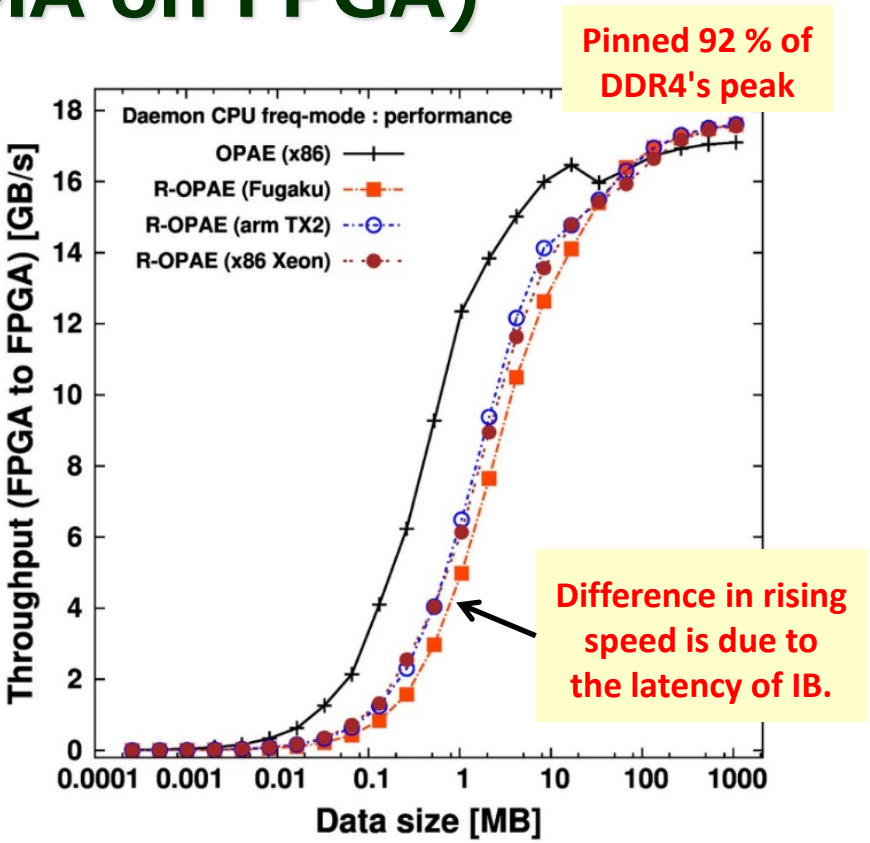
R-OPAE over 100G IB

Difference in rising speed is due to the latency of IB.

a) Host-to-FPGA DMA transfer by OPAE or R-OPAE

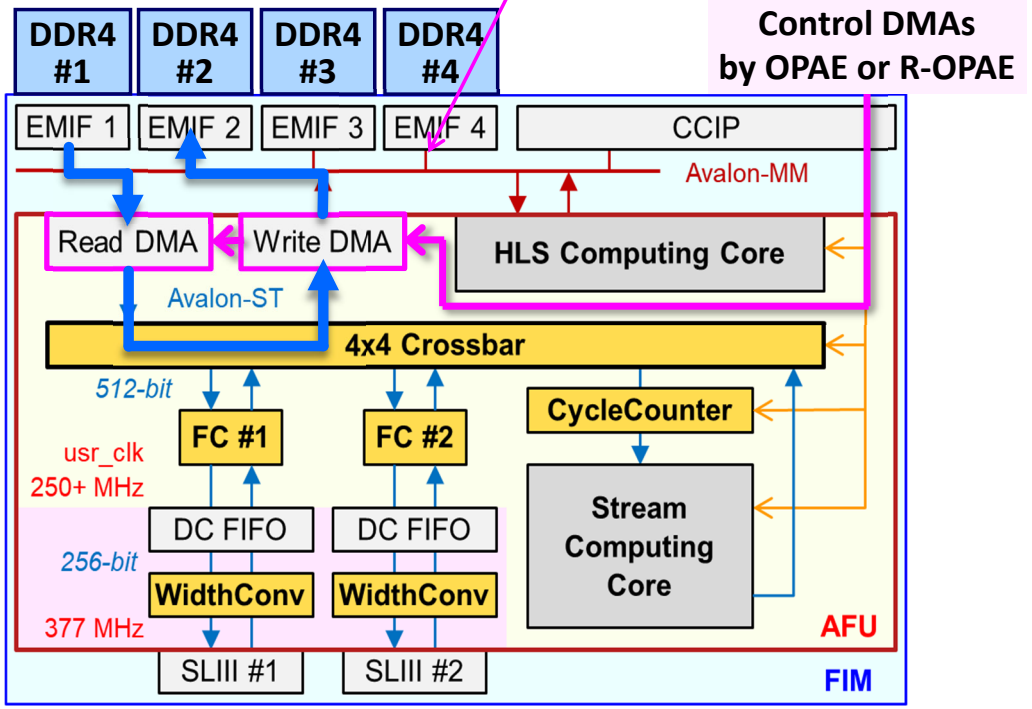
b) FPGA-to-Host DMA transfer by OPAE or R-OPAE

Control Overhead by R-OPAE (DMA on FPGA)



FPGA-to-FPGA DMA transfer controlled by OPAE or R-OPAE

PCIe Gen3 x16	16 GB/s
DDR4-2400	19.2 GB/s
100Gbps IB EDR	12.5 GB/s

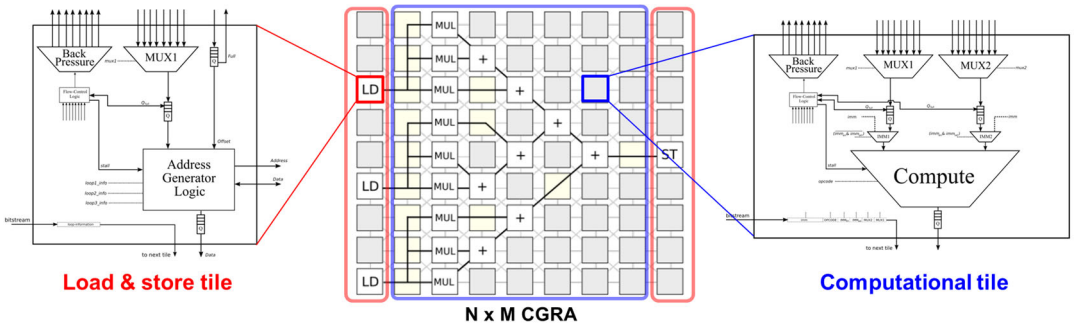


Hardware behavior of AFU Shell for FPGA-to-FPGA DMA transfer

Applications,
Joint Research Projects

Ongoing (Joint) Research Projects

- **Hardware**
  - ✓ Processor Team **CGRA**
  - ✓ Kumamoto Univ **AI Engine** (ReNA)
- **System Software**
  - ✓ RIKEN RPC for FPGAs
  - ✓ Tohoku Univ neoSYCL (on Fugaku)
- **Applications**
  - ✓ Univ of Tokyo Bayesian network analysis
  - ✓ Meiji Univ **3D FFT** (presented later)
  - ✓ Processor Team Fluid simulation
  - ✓ Nagasaki Univ Convex method
  - ✓ Hiroshima City U Breadth First Search for a graph
  - ✓ Processor Team Hardwired MNIST
  - ✓ JAIST Sound rendering



Riken CGRA (coarse-grained reconfigurable array)

AI Engine, ReNA

Summary

Goal Explore new system architectures for reconfigurable HPC

This project **ESSPER**: Elastic and Scalable System for High-Performance Reconfigurable Computing

PoC : ***Interoperability and flexibility, Platform for customizability, and scalablity***

Research subjects

- ✓ More applications with multi-FPGAs, Indirect network on Ether, Resource management & Task scheduling for Fugaku

Waiting for Collaboration with You!

Hiring researchers:
R-CCS2015 or R-CCS2022