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

RIKEN R-CCS PROCT

ESSPER: FPGA Cluster for Researches on Reconfigurable HPC with Supercomputer Fugaku

Kentaro Sano

RIKEN Center for Computational Science (R-CCS)

July 7, 2022

Introduce Myself : Kentaro Sano

Hiring researchers:
R-CCS2105 or
R-CCS2022

RIKEN Center for Computational Science

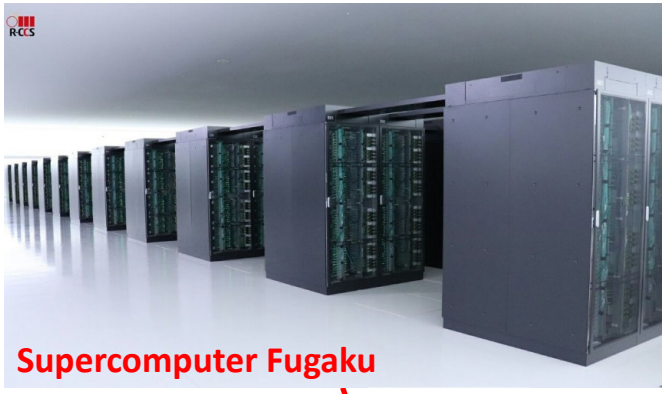
- ✓ Develop and operate Supercomputer Fugaku
- ✓ Facilitate leading edge infrastructures for research based on supercomputers
- ✓ Conduct cutting-edge research on HPC

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"



Goal and Roadmap of Processor Research Team

Establish HPC architectures suitable for Post-Moore Era

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

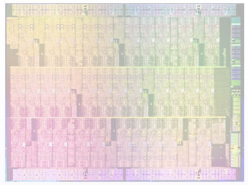
Advancement of Fugaku

This talk

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

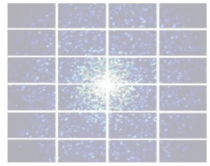
Exploration of New HPC Architectures

- ✓ Novel accelerators based on data-flow model (CGRA)
- ✓ System architectures



Near-sensor / Near-storage Processing

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

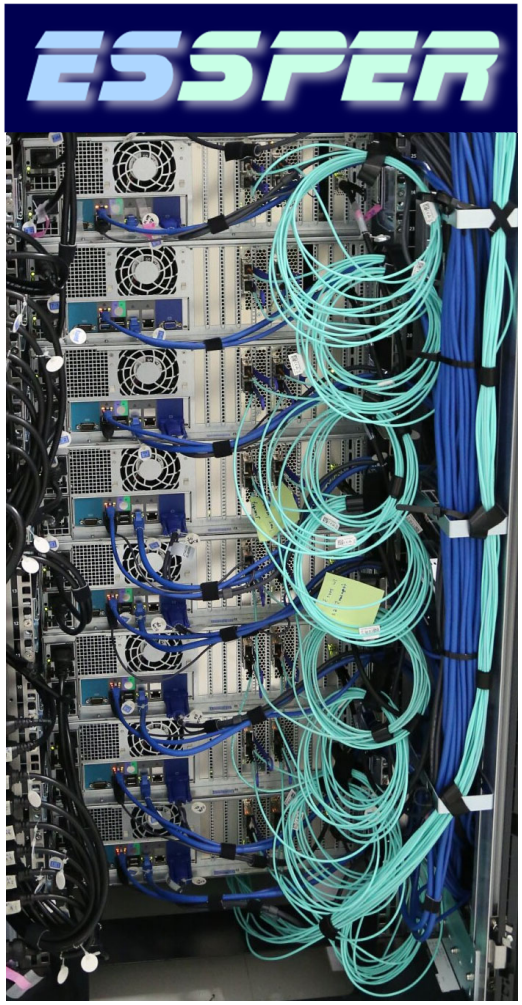


Exploration for Novel Computing Principle

- ✓ Specialized hardware design for quantum error correction

Outline

- Introduction
- **ESSPER :** Proof-of-Concept FPGA Cluster System
- How to Program?
- Inter-FPGA Network
- Summary

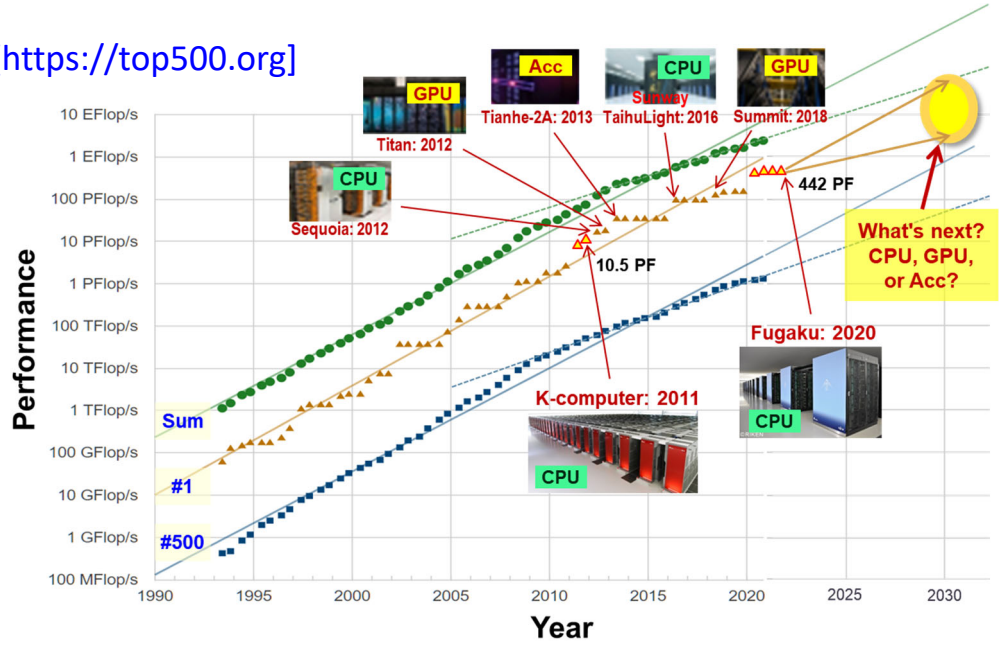


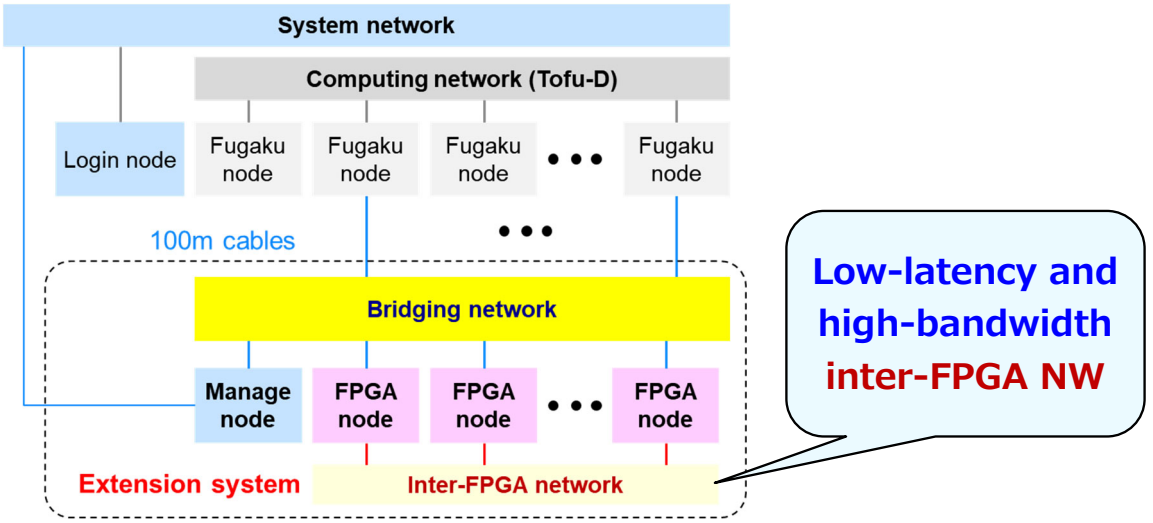
ESSPER: FPGA Cluster System

Introduction

- **Accelerators for higher power-efficiency**
  - ✓ System power is the most critical issue.
  - ✓ Standard CPUs are not sufficient.
    Accelerators (Accs) for higher performance per power
- **Reconfigurable computing with FPGAs**
  - ✓ GPUs are popular as gen-purpose Accs.
  - ✓ More specialized, higher efficiency.
    But we also need flexibility. **FPGAs!**
- **Prototype FPGA Cluster "ESSPER"**
  - ✓ Proof-of-concept system to evaluate functional extension of Fugaku.
  - ✓ Challenges:
    What architecture?
    How to program?

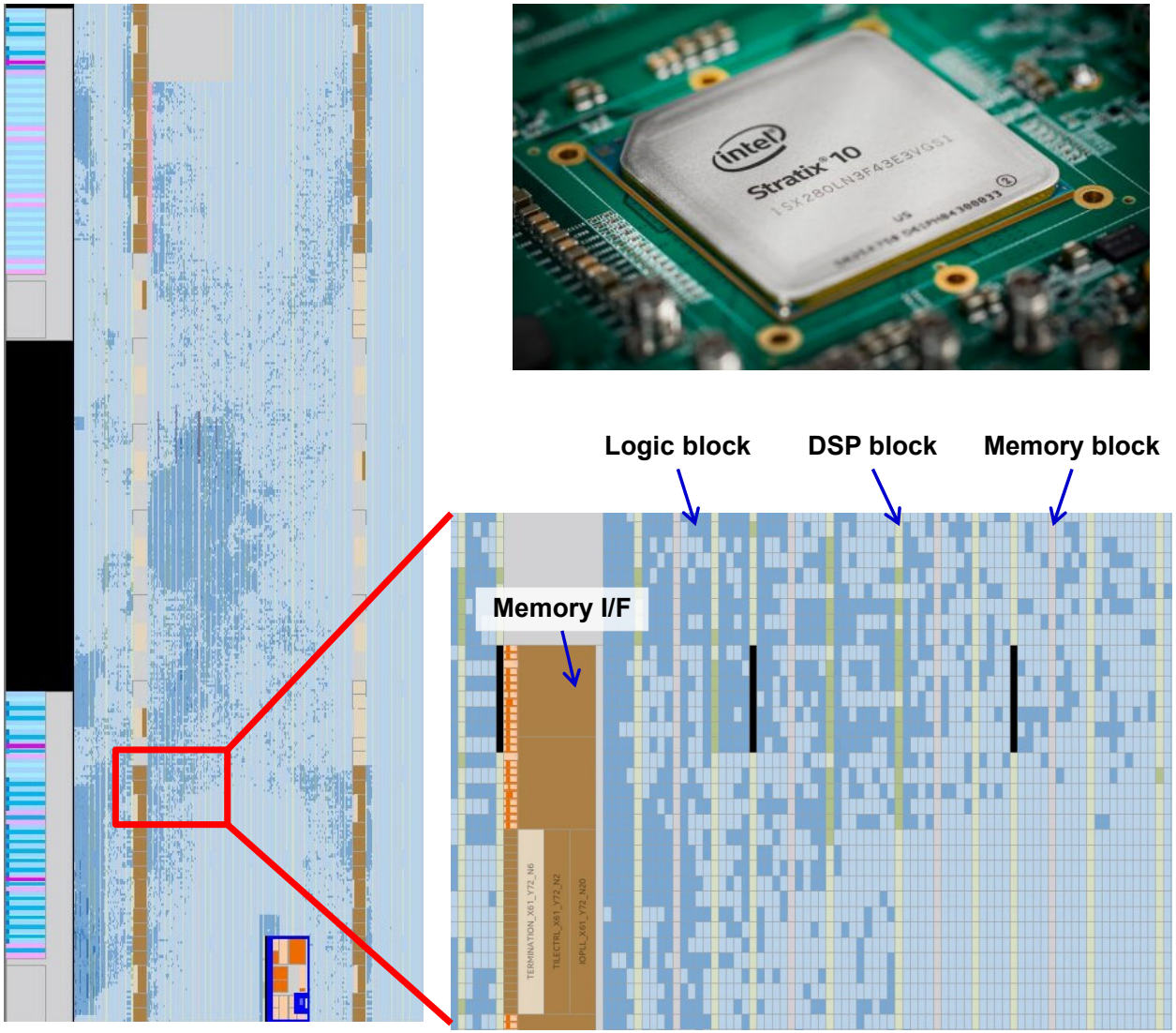
[https://top500.org]



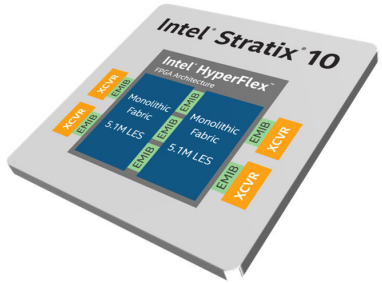


What's FPGA?

- **Reconfigurable device**
  - ✓ Allow you to construct any circuits
- **Array of hardware "blocks"**
  - ✓ Logic blocks
  - ✓ Memory blocks
  - ✓ DSP blocks (integer, floating-point)
  - ✓ I/O blocks (mem I/F, PCIe, HSSI)
  - ✓ SW blocks and wires
  - ✓ Clock trees, etc.
- **Recent FPGA**
  - ✓ Sufficiently big for system-on-chip
  - ✓ Specialized hardware for HPC



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)

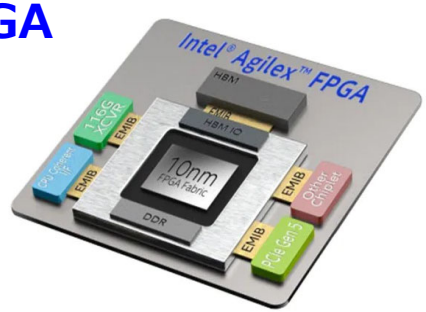
- ➢ 5760 floating-point DSPs
- ➢ comparative to CPU, GPU (DDR4, HBM2)
- ➢ aggregate ~1000 TB/s
- ➢ multiple tx / rx of 100 Gbps

Use cases in data-centers, cloud, or HPC

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

Intel Agilex FPGA

More advanced next-generation 7nm FPGA in a few years



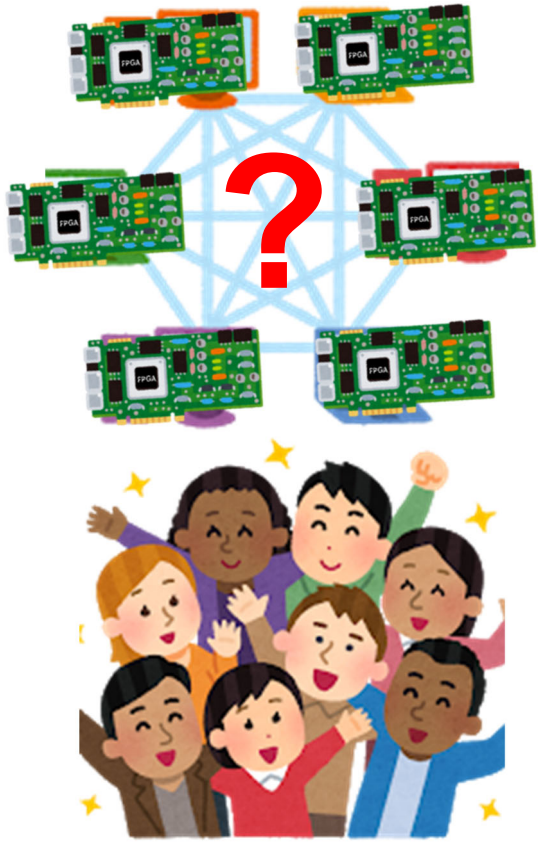
Promising not only for computing, but also for data movement

Motivation and Objective

- **Challenges of FPGA based HPC**
  - ✓ How to program with sufficient customizability
  - ✓ How to scale performance with multiple FPGAs
  - ✓ How to use FPGAs w/ existing systems (***interoperability***)
- **Objective**

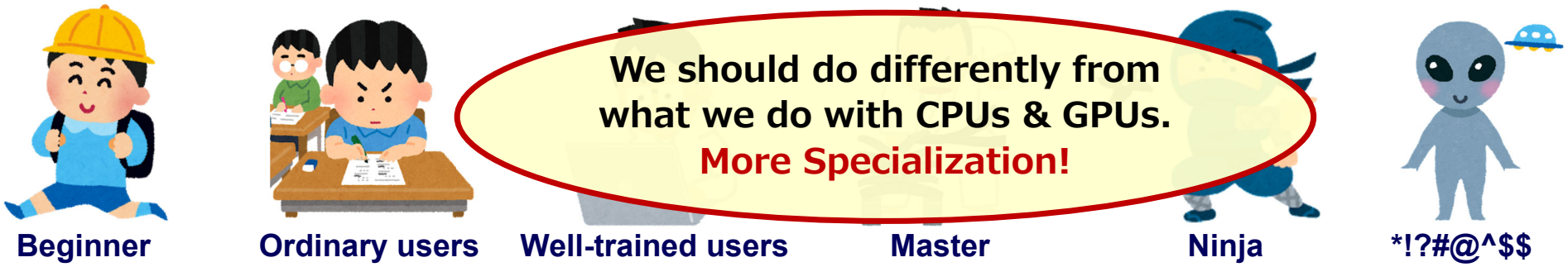
To find appropriate architecture & programming environment of FPGA cluster for HPC

- ✓ Design and implement a system stack with hardware and software
- ✓ Prototype FPGA Cluster for Proof-of-Concept
- ✓ Study application use cases



What is Appropriate for Programming FPGAs?

Gradual transition of different levels



Easy ——— Mastery ———→ Hard

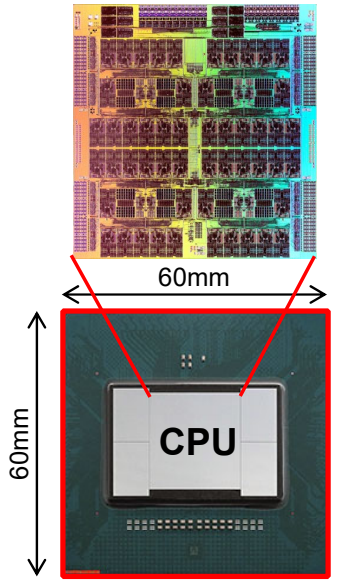
Unnecessary ——— Knowledge of hardware / architecture ———→ Must

Few ——— Opportunities for acceleration / customization ———→ A lot

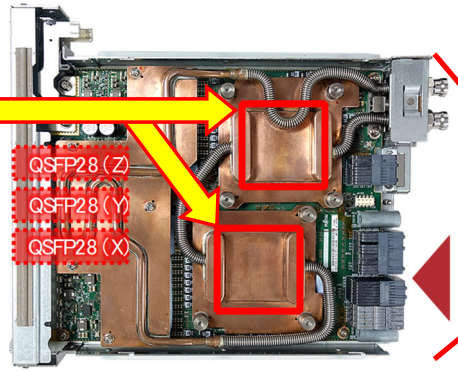
ESSPER : FPGA Cluster Prototype

System Configuration of Fugaku

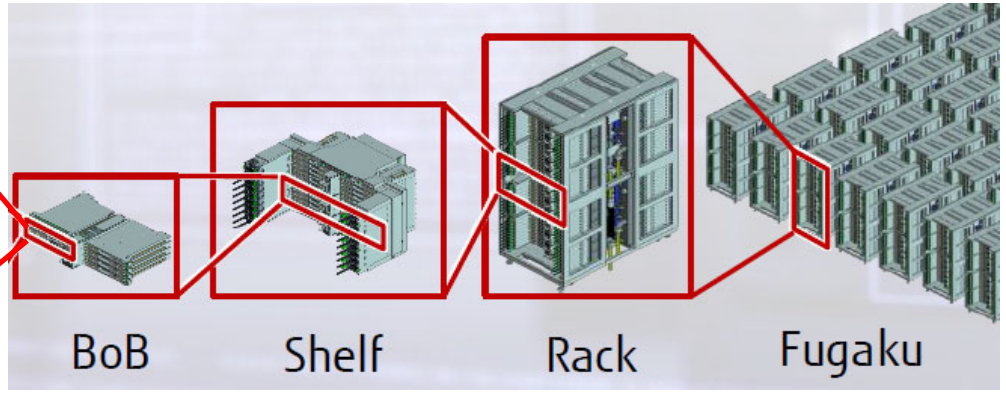
Modern Supercomputers are based on Many-core CPUs (& GPUs).



48+ cores / 1 node
2.7+ TF



CPU-Memory Unit (CMU) 2 nodes
5.4+ TF



16 nodes
43+ TF

48 nodes
129+ TF

384 nodes
1+ PF

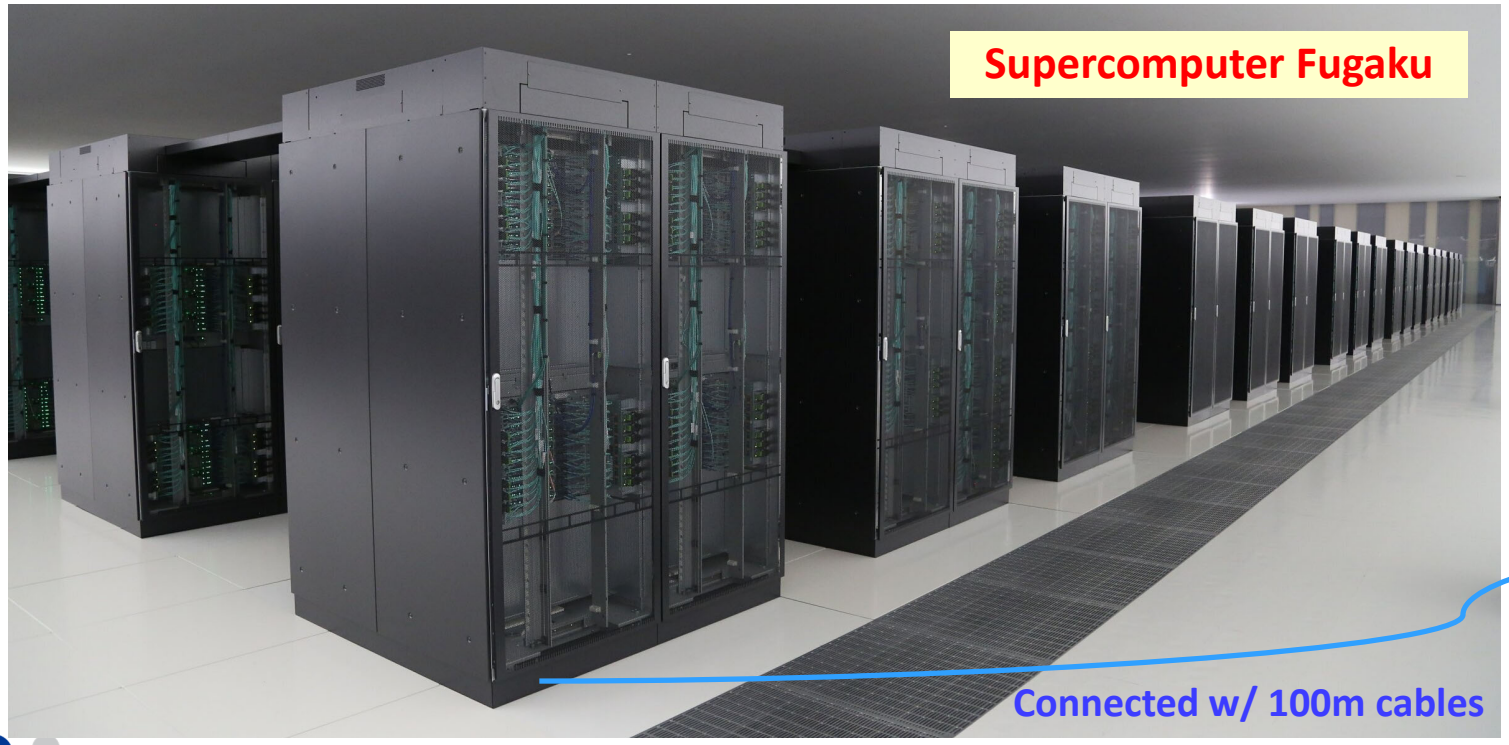
158,976 nodes
537 PF @ FP64
(414 racks)

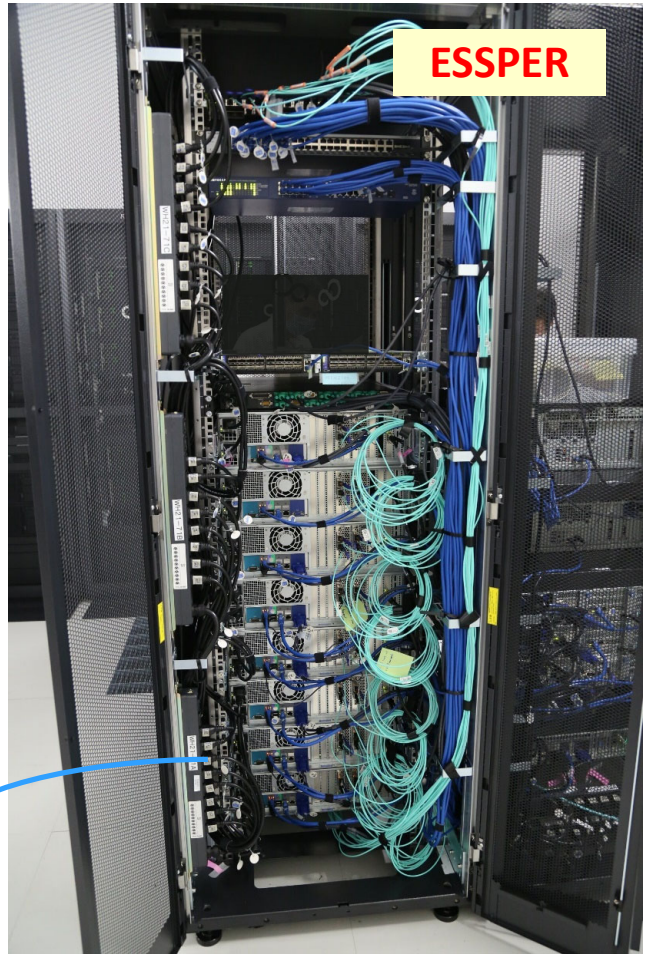
Photos & figs by Fujitsu

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

Elastic and Scalable System for High-PErformance Reconfigurable Computing

Experimental prototype
for research on functional extension with FPGAs





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

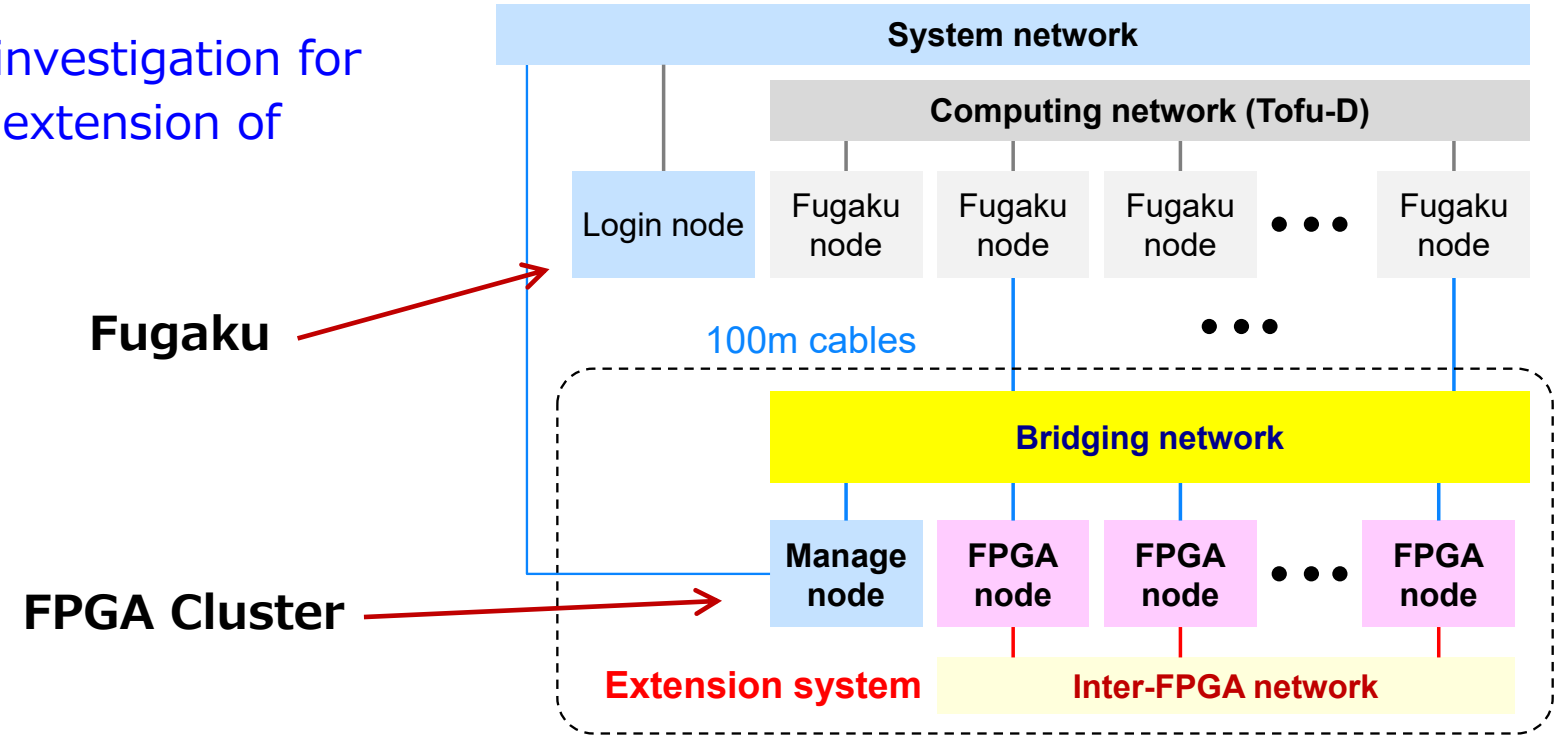
Elastic and Scalable System for High-PErformance Reconfigurable Computing

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

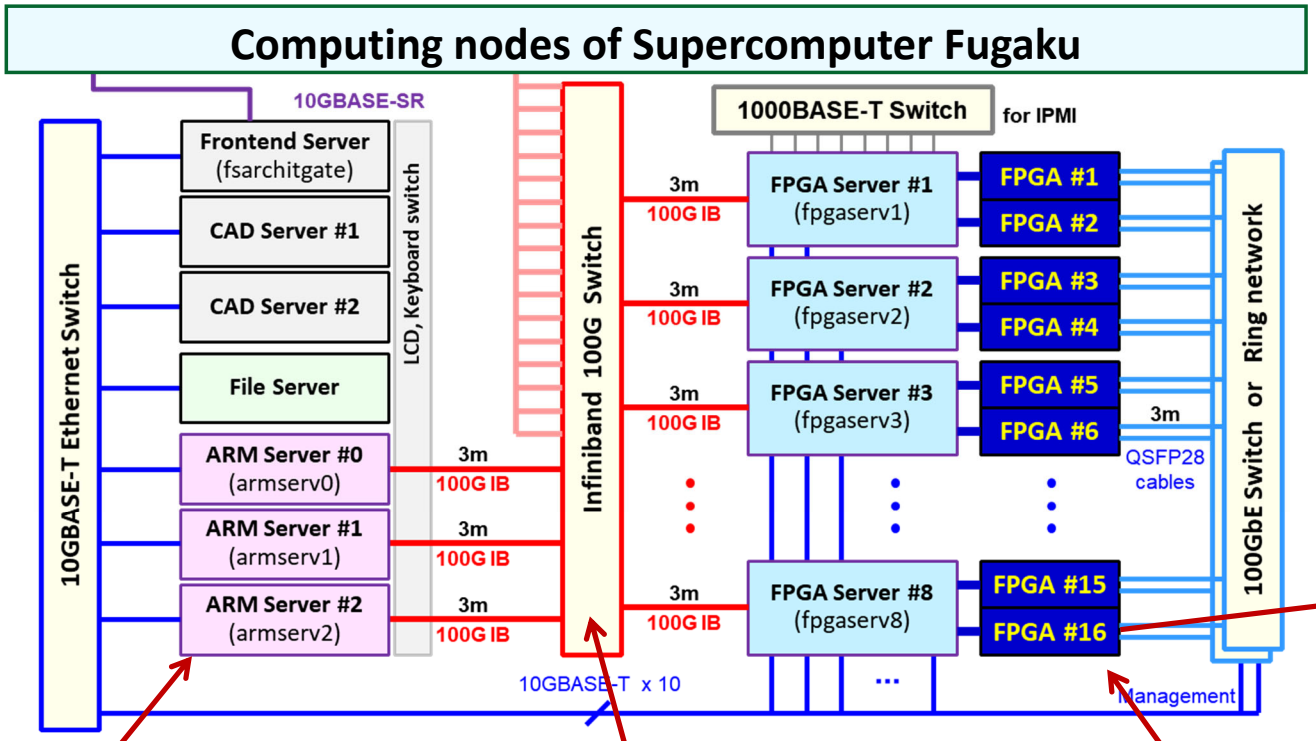
Architecture of ESSPER

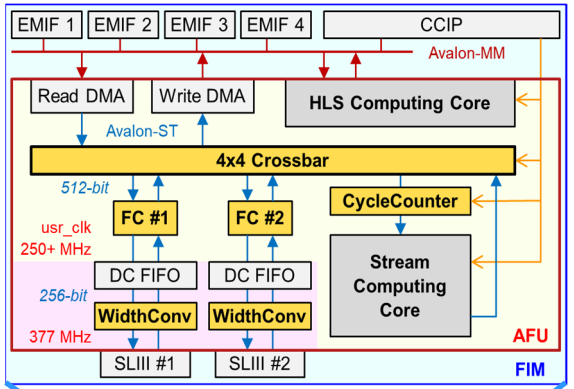
Goal

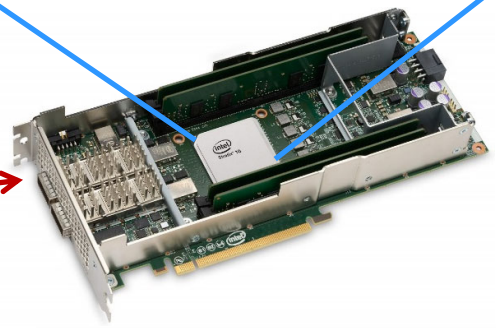
- ✓ Technical investigation for functional extension of Fugaku.



Hardware Organization of ESSPER







Various servers

- CAD servers
- Storage server
- ARM servers

CPU - FPGA network

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

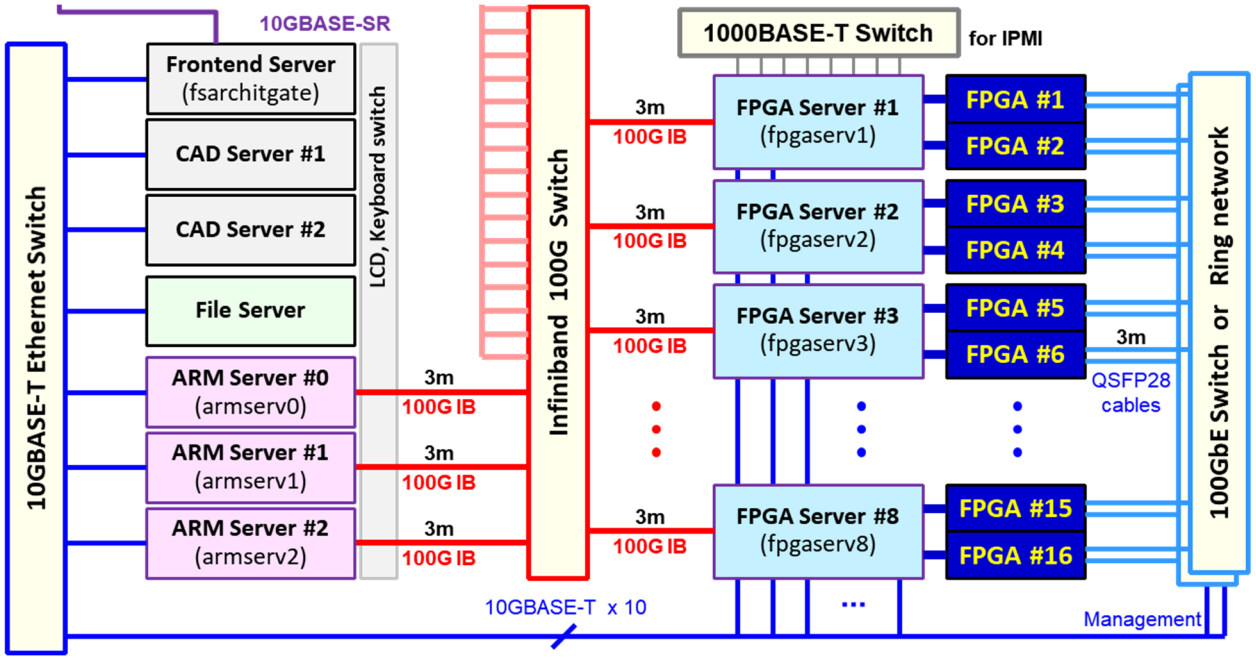
FPGA cluster

- FPGA host servers (x86)
- FPGA boards
- Inter-FPGA network

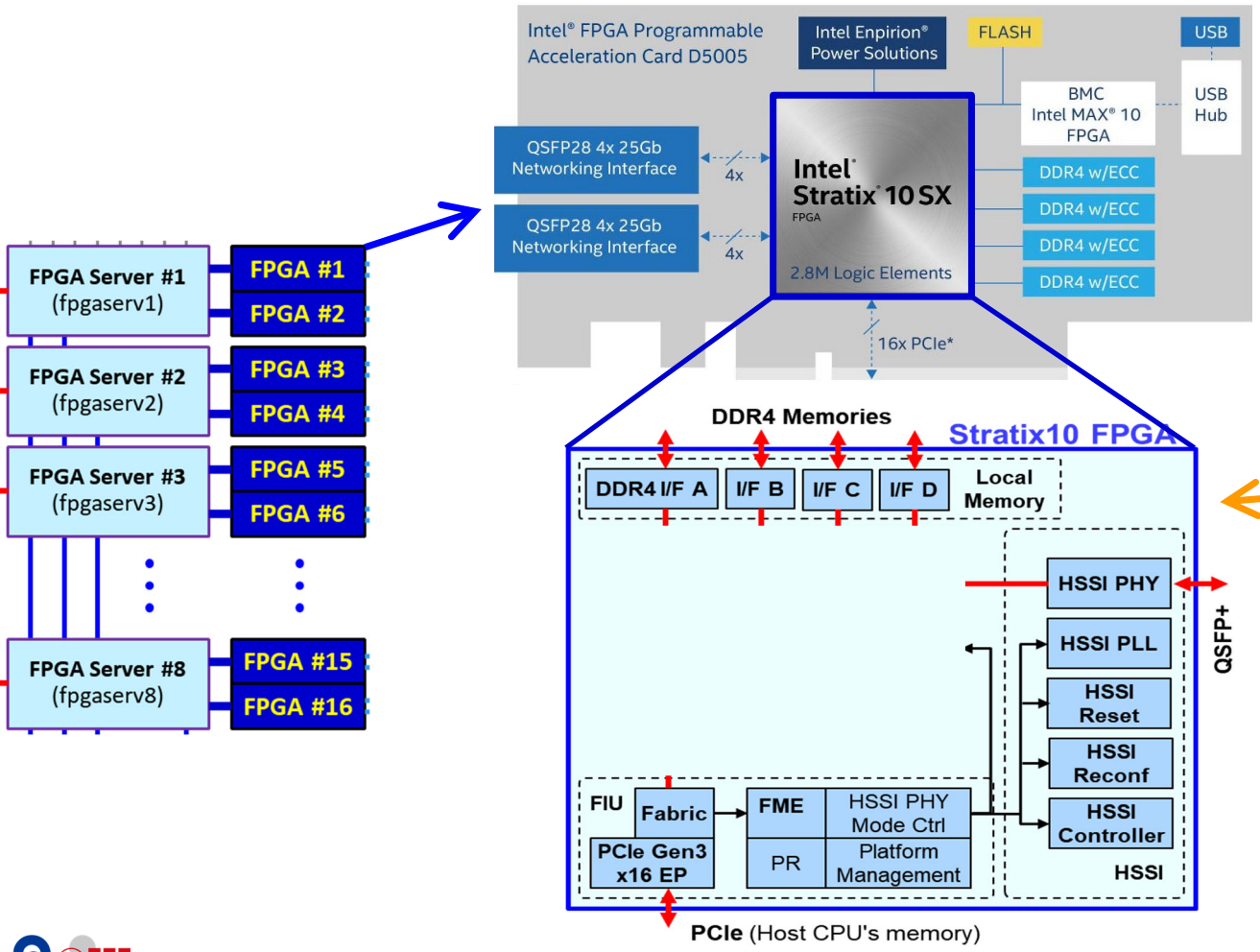
FPGA SoC

- AFU Shell design
- FPGA object class as HAL
- Programming by HLS, DSL

Hardware Organization of ESSPER



FPGA System-on-Chip



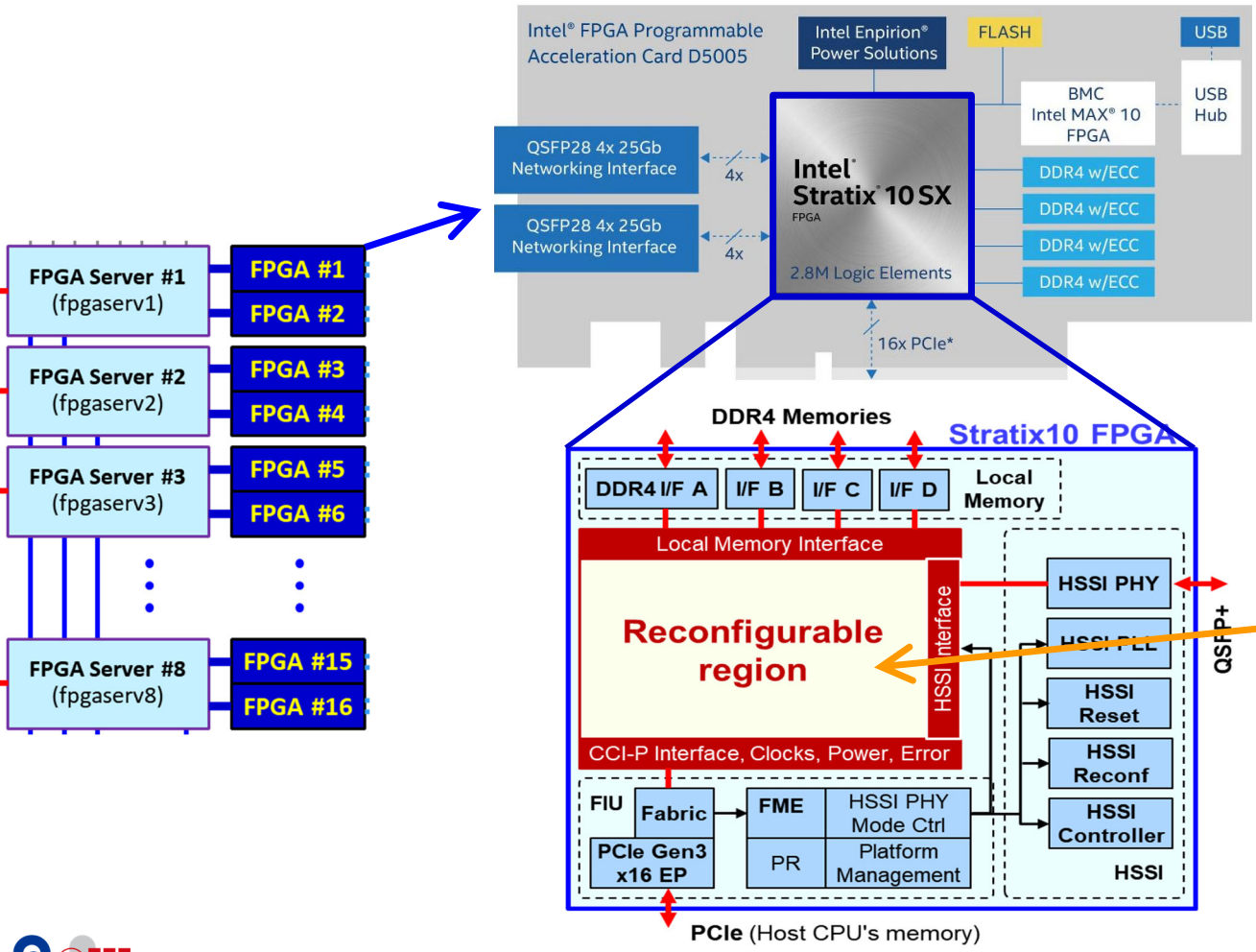
Intel FPGA PAC D5005

- ✓ *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

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)

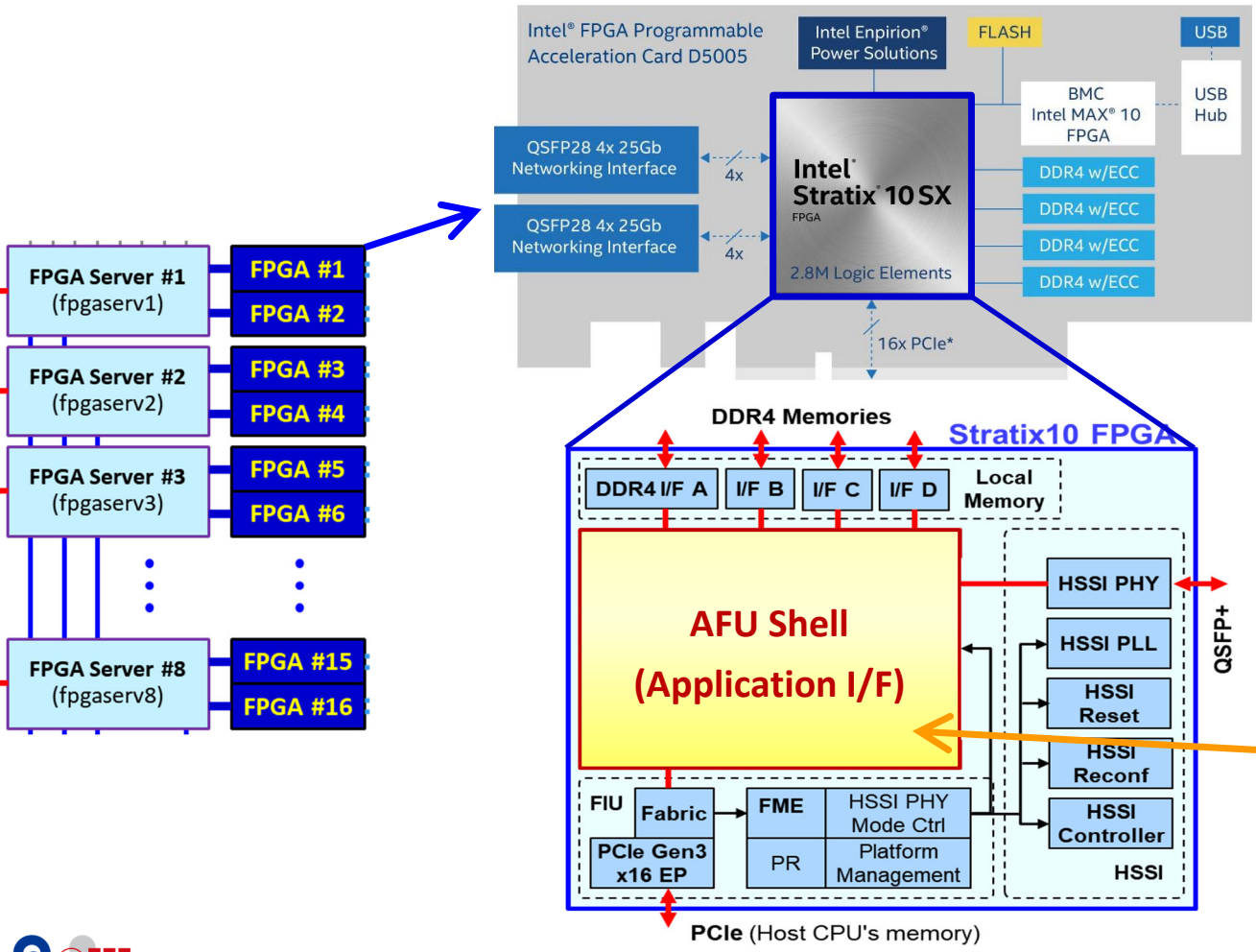
FIM (FPGA Interface Manager)

- ✓ Fixed region including I/F

AFU (Acceleration Function Unit)

- ✓ Reconfigurable region

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

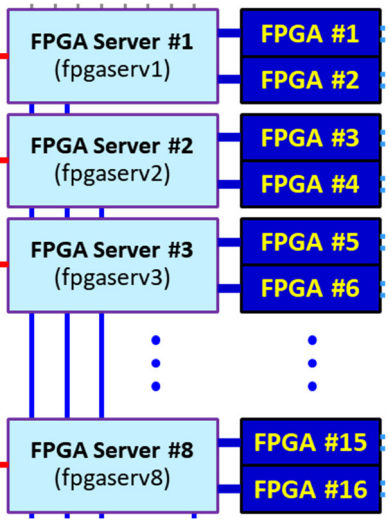
AFU (Acceleration Function Unit)

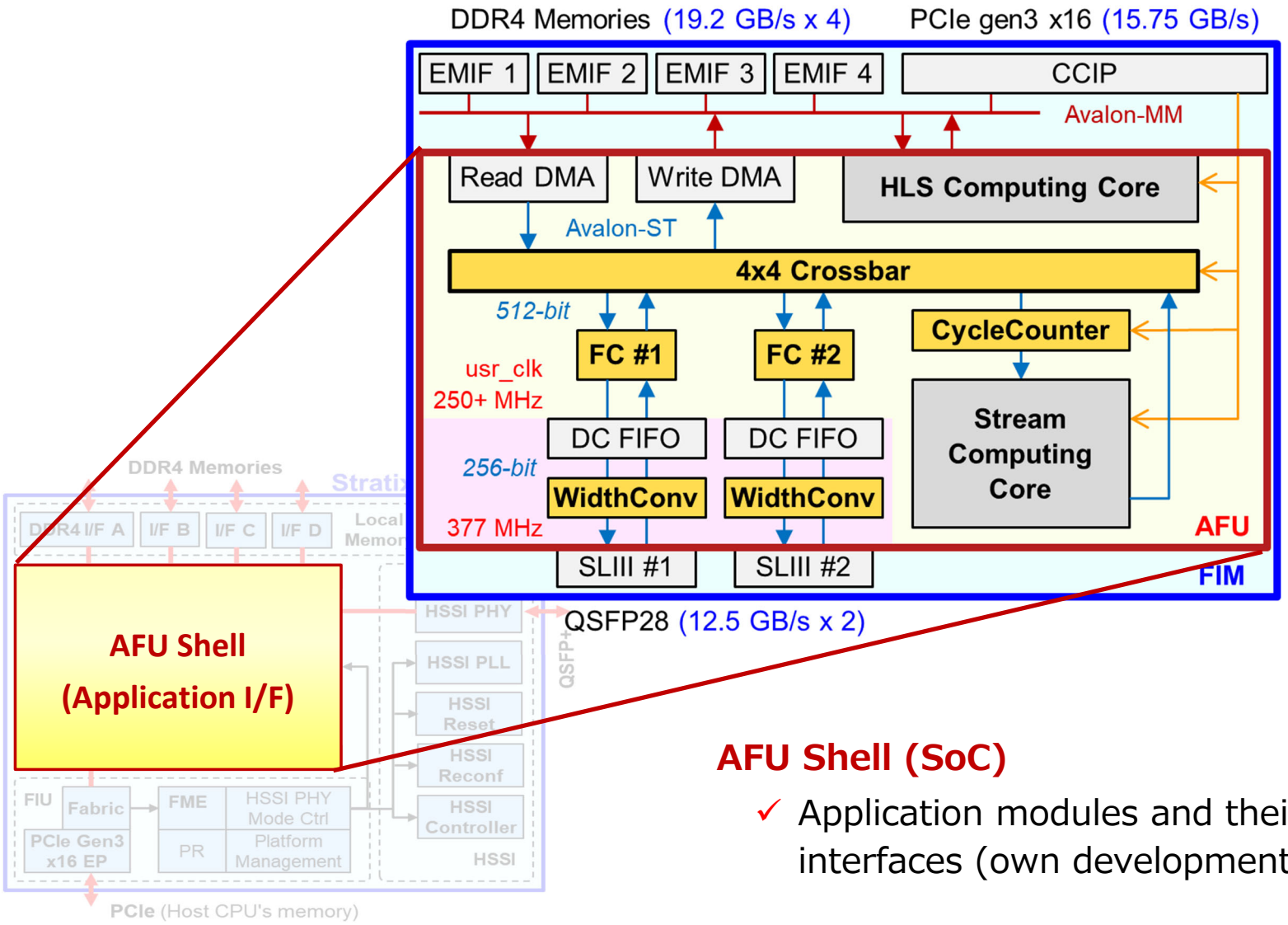
- ✓ Reconfigurable region

AFU Shell (SoC)

- ✓ Application modules and their interfaces (own development)

AFUShell

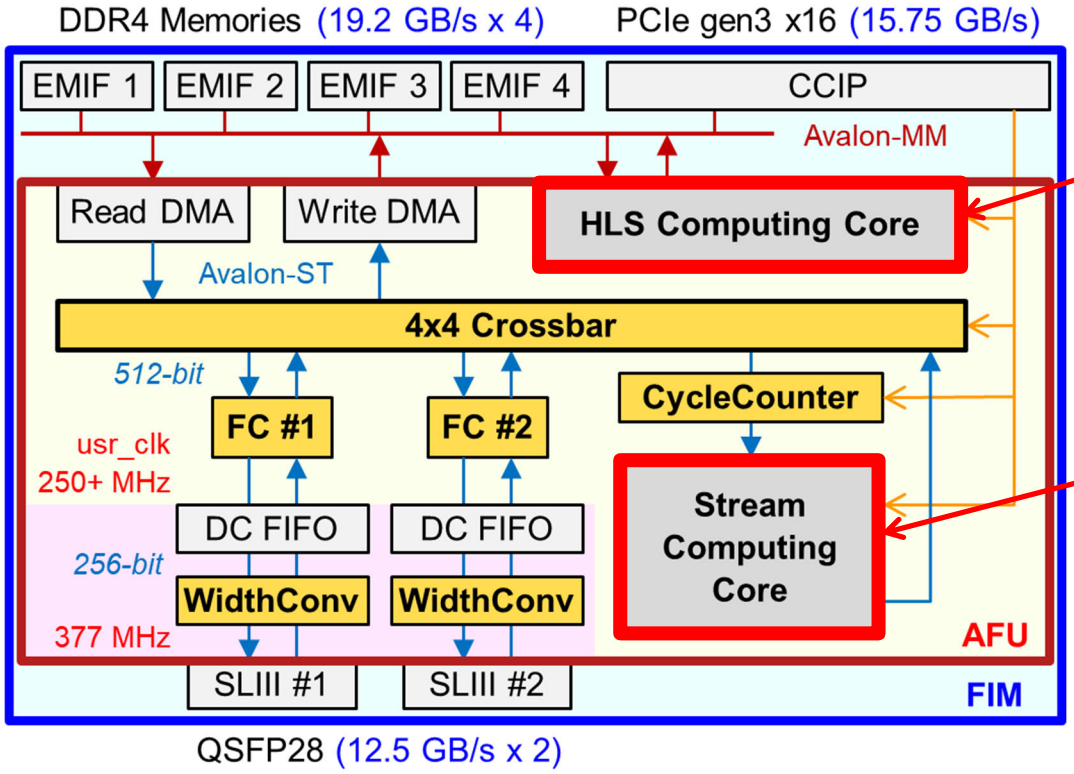




AFU Shell (SoC)

- ✓ Application modules and their interfaces (own development)

Embed your own Computing Core in AFU Shell



Memory-mapped Computing Core

- Connected to DDR4 memory controllers
- Autonomously read & write DDR4 to compute
- Computation with complex data structures

Stream Computing Core

- Connected to crossbar for data streams
- Passively compute with given data stream
- Pipelined computation at high throughput

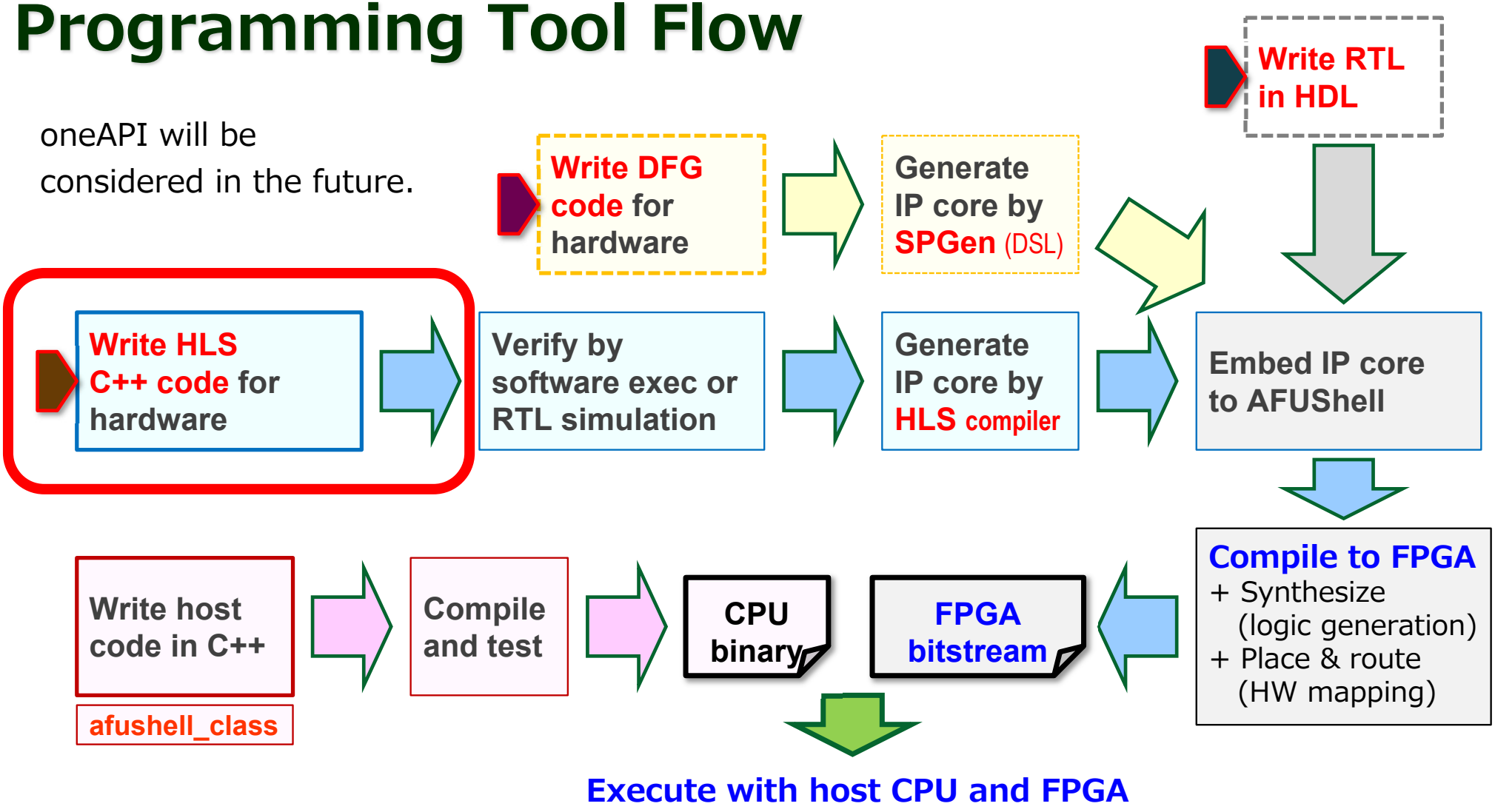
Cores are programed by using

- HLS (high-level synthesis) with C++,
- HDL (hardware description language), or
- Others (DSL, chisel, etc.)

How to Program

Programming Tool Flow

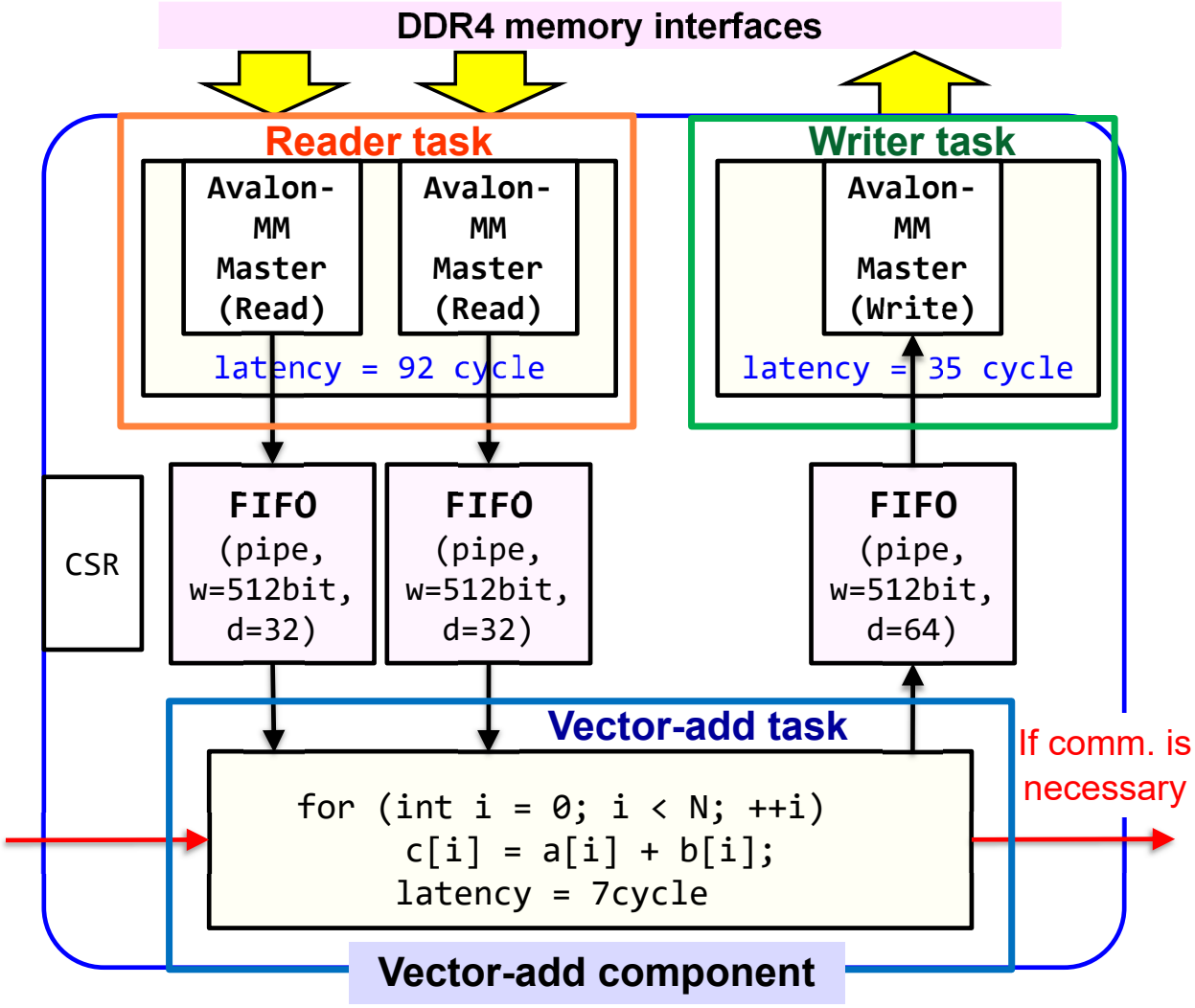
oneAPI will be considered in the future.



Programming Example with High-Level Synthesis (HLS)

Example: Vector-add module

- **Concurrently-working hardware modules**
  - ✓ Reader task module
  - ✓ Writer task module
  - ✓ Vector-add task module
- **Top module**
  - ✓ Control-status registers
- **Program host software separately**



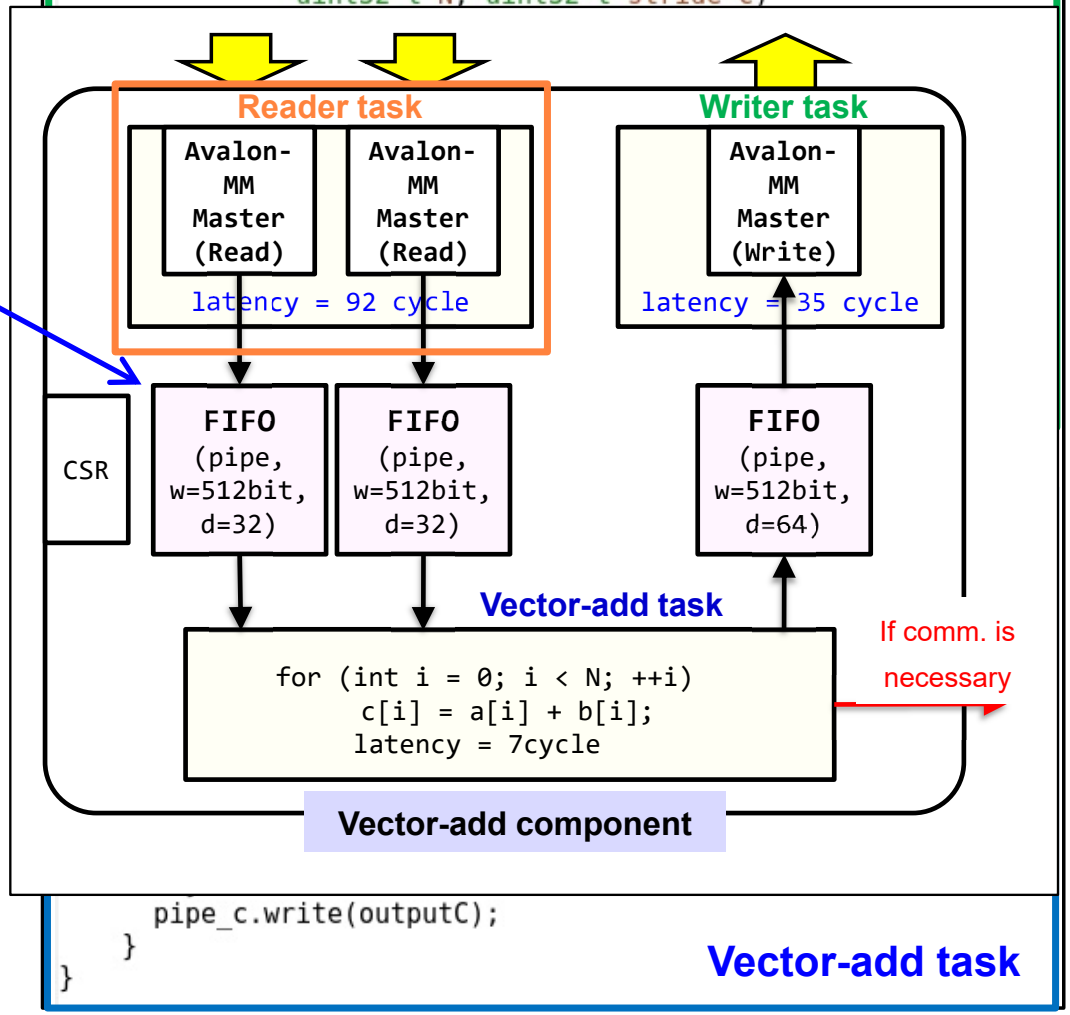
HLS Code for Vector-add

```
struct pipe_width_float
{
  float data[16];
};

// ihc::pipe
ihc::pipe<input_pipe_id<1>, pipe_width_float, pipe_depth> pipe_a;
ihc::pipe<input_pipe_id<2>, pipe_width_float, pipe_depth> pipe_b;
ihc::pipe<input_pipe_id<3>, pipe_width_float, pipe_depth> pipe_c;

// Avalon-MMでDDRからデータを読んできて、FIFOに入れるtask
void mm_reader(avmm_master_port<float, 1> *a,
               avmm_master_port<float, 2> *b,
               uint32_t N, uint32_t stride_a, uint32_t stride_b)
{
#pragma unroll
  for (uint32_t i = 0; i < N; i += pipe_width)
    {
      pipe_width_float inputA, inputB;
#pragma unroll
      for (uint32_t j = 0; j < pipe_width; j++)
        {
          uint32_t index = (i + j);
          inputA.data[j] = (*a)[index * stride_a];
          inputB.data[j] = (*b)[index * stride_b];
        }
      pipe_a.write(inputA); // 16個データが来るまで待つことになる
      pipe_b.write(inputB);
    }
}
```

FIFOs

Reader task

```
// FIFOからデータを取り出して、Avalon-MMでDDRに書き込むtask
void mm_writer(avmm_master_port<float, 3> *c,
               uint32_t N, uint32_t stride_c)
```



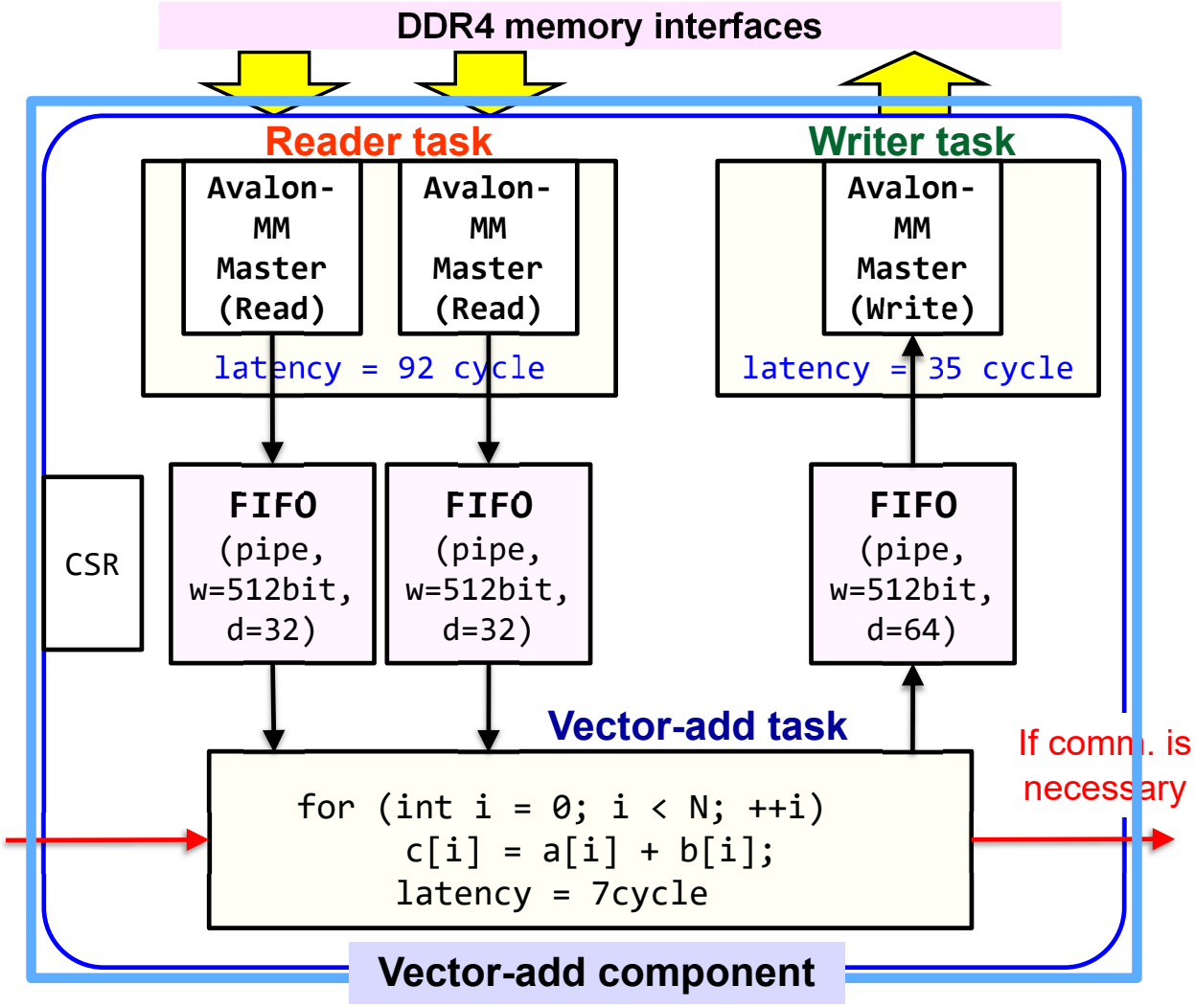

```
      pipe_c.write(outputC);
    }
}
```

Vector-add task

Programming Example with High-Level Synthesis (HLS)

Example: Vector-add module

- **Concurrently-working hardware modules**
  - ✓ Reader task module
  - ✓ Writer task module
  - ✓ Vector-add task module
- **Top module**
  - ✓ Control-status registers
- **Program host software separately**



HLS Code for Vector-add 2 of 2

Top module

```
component hls_avalon_slave_component
void mm_master_component(
  hls_avalon_slave_register_argument avmm_master_port<float, 1> &a,
  hls_avalon_slave_register_argument avmm_master_port<float, 2> &b,
  hls_avalon_slave_register_argument avmm_master_port<float, 3> &c,
  hls_avalon_slave_register_argument uint32_t N,
  hls_avalon_slave_register_argument uint32_t stride_a,
  hls_avalon_slave_register_argument uint32_t stride_b,
  hls_avalon_slave_register_argument uint32_t stride_c)
{
  ihc::launch<mm_reader                >(&a, &b, N, stride_a, stride_b);
  ihc::launch<mm_writer                >(&c, N, stride_c);
  ihc::launch<vector_add_part5_stride>(N);

  ihc::collect<vector_add_part5_stride>();
  ihc::collect<mm_writer               >();
  ihc::collect<mm_reader               >();
}
```

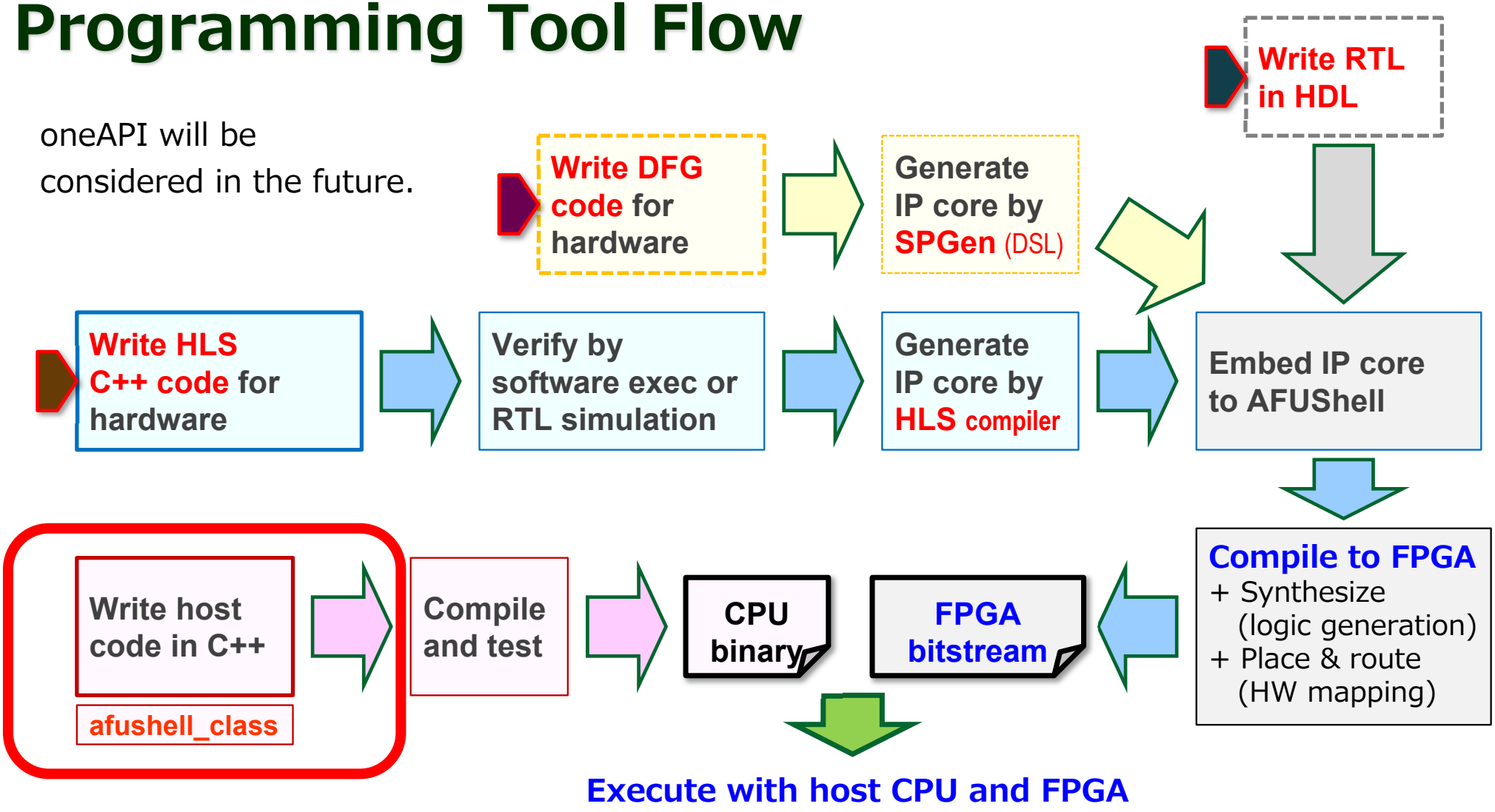
Instantiate

Main function

```
int main(int argc, char **argv)
{

  unsigned int TEST_SIZE = std::atoi(argv[1]);
  const uint32_t stride_size = 2;
  const uint32_t elem_size = TEST_SIZE * stride_size;

  float A[elem_size];
  float B[elem_size];
  float C[elem_size];

  // mm_master interface class instances
  avmm_master_port<float, 1> mm_A(A, sizeof(float) * elem_size);
  avmm_master_port<float, 2> mm_B(B, sizeof(float) * elem_size);
  avmm_master_port<float, 3> mm_C(C, sizeof(float) * elem_size);

  // prepare the input data
  for (uint32_t i = 0; i < elem_size; i++)
    {
      A[i] = static_cast<float>(i); //rand();
      B[i] = static_cast<float>(i); //rand();
    }

  // Run the component
  mm_master_component(mm_A, mm_B, mm_C, TEST_SIZE,
                      stride_size, stride_size, stride_size);

  return 0;
}
```

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.

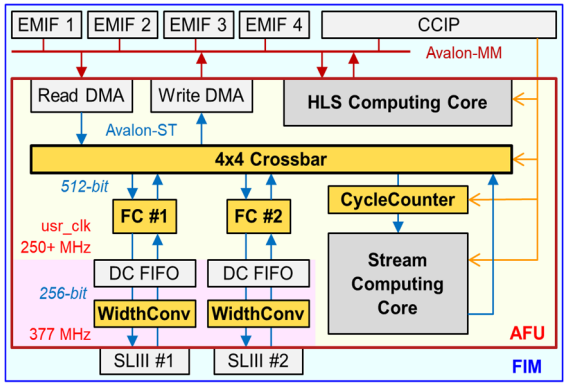
- ✓ Instantiate **AFUShell object**
- ✓ Open object
- ✓ Use modules / services
  - ➢ *Crossbar*
  - ➢ *Hardware cycle-counter*
  - ➢ *DMA (Host-FPGA, FPGA-FPGA)*
  - ➢ *Computing core*
- ✓ Close object

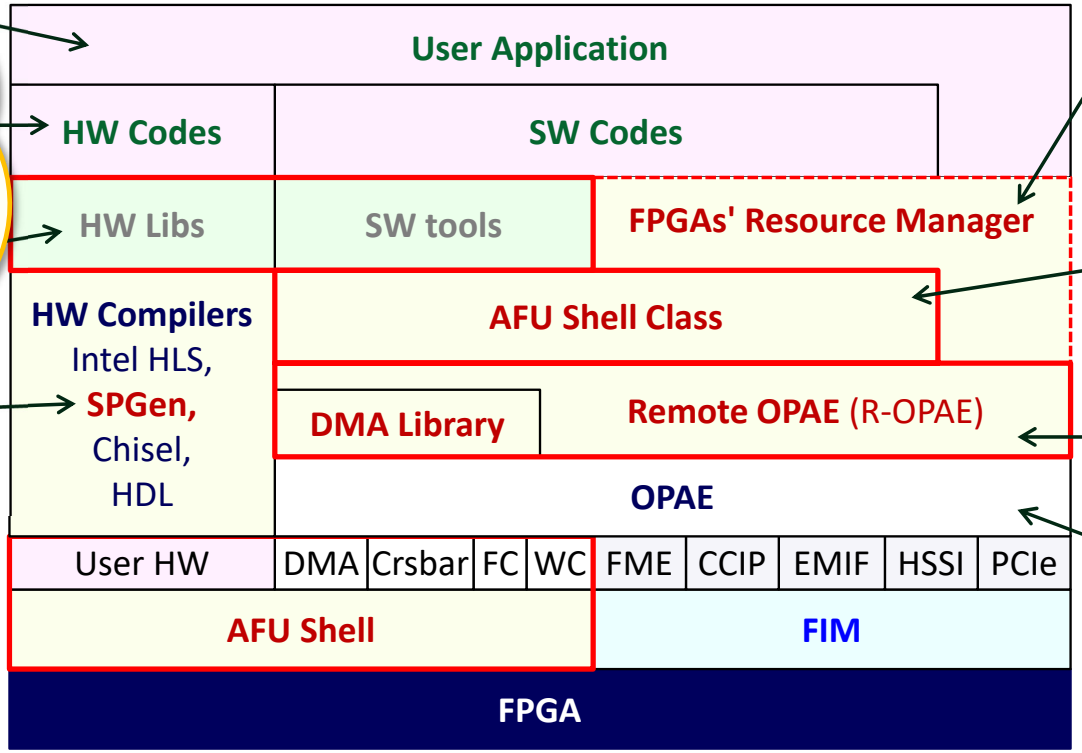
```
int very_simple_example(void)
{
  uint32_t allCycles, validCycles, csr;
  uint64_t bytes = 1024*1024*64;
  char *begin_ptr = (char *)malloc(sizeof(char)*bytes);

  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;
}
```

System Stack of ESSPER

Call for Joint researches:

- 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

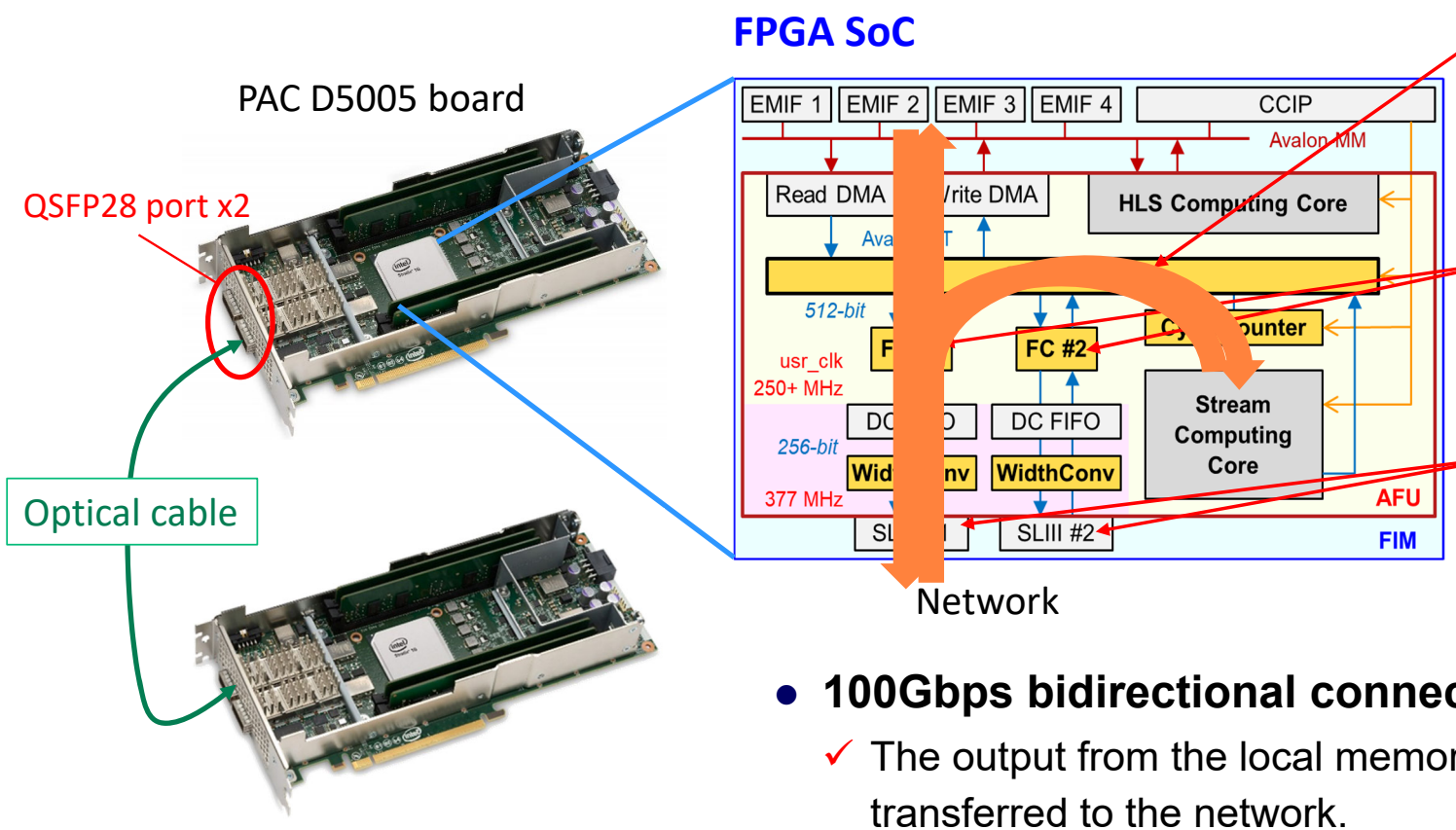
AFU Shell class
- Object of AFU shell
- Abstraction of HW

Remote OPAE
- Software bridge using Infiniband Verbs

OPAE
- Low-level driver

Application Example using Inter-FPGA Network

Direct Connection Network (DCN)



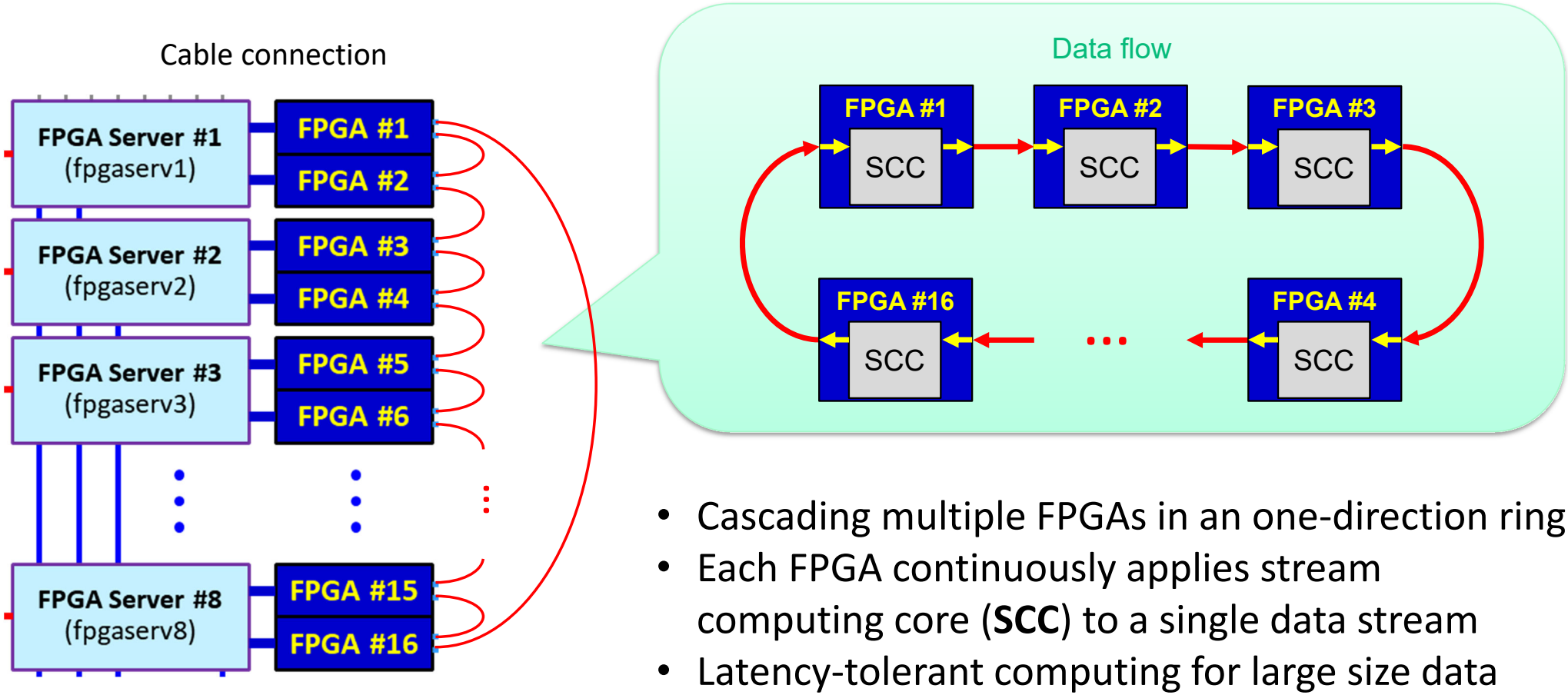
Control data routing among local memory, computing core, and network

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.

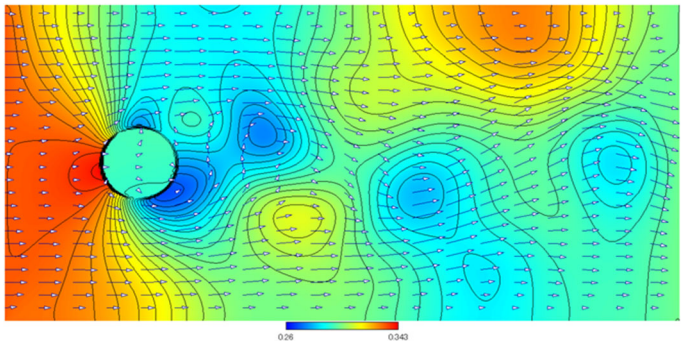
Stream Computing with a Ring Network

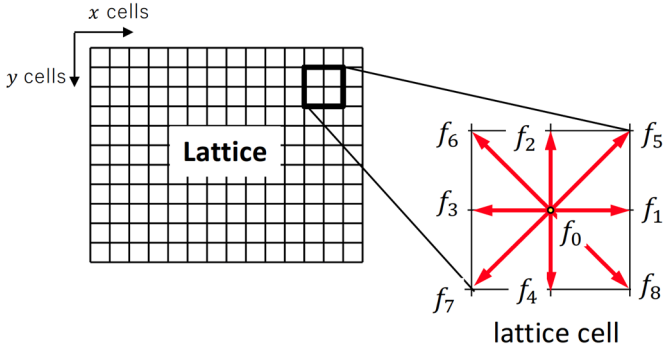


- Cascading multiple FPGAs in an one-direction ring
- Each FPGA continuously applies stream computing core (**SCC**) to a single data stream
- Latency-tolerant computing for large size data

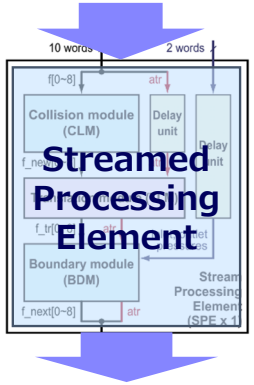
Streamed Fluid Simulation with Multi FPGAs

2D Lattice Boltzmann Method (LBM)





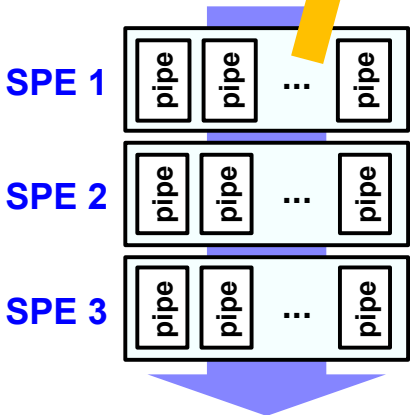
input streams (6.4 GB/s)

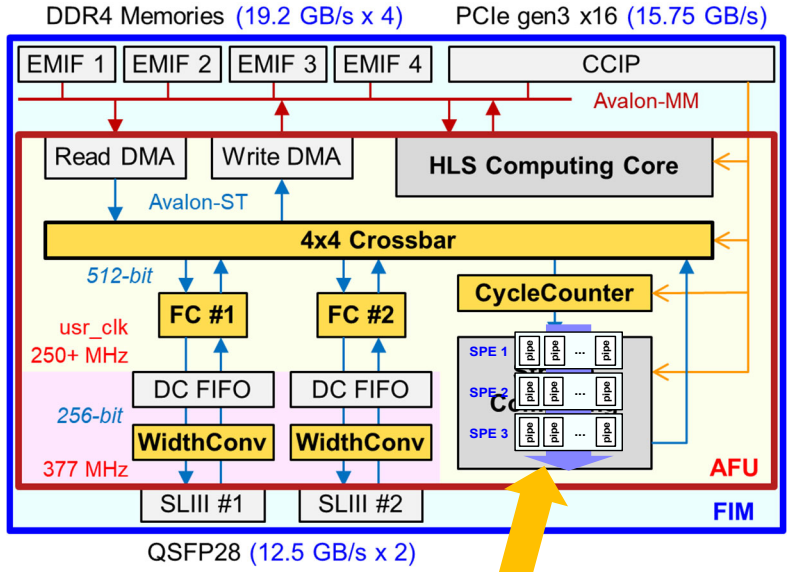


output streams

2D LBM SPE

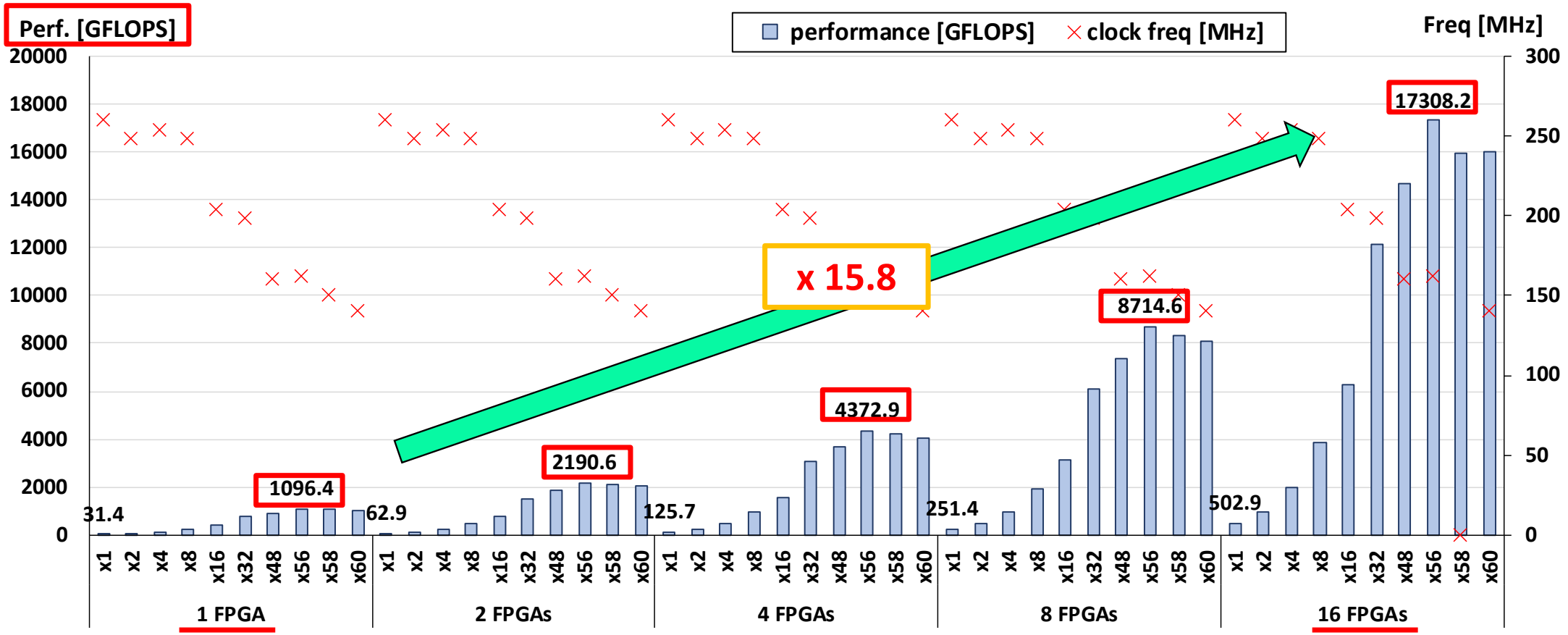
Cascading SPEs





Performance of 2D LBM with 100G DCN

Computational performance (FLOPS) when processing about 2GB data



On-going Research Projects Using ESSPER

On-going (Joint) Research Projects

Hardware

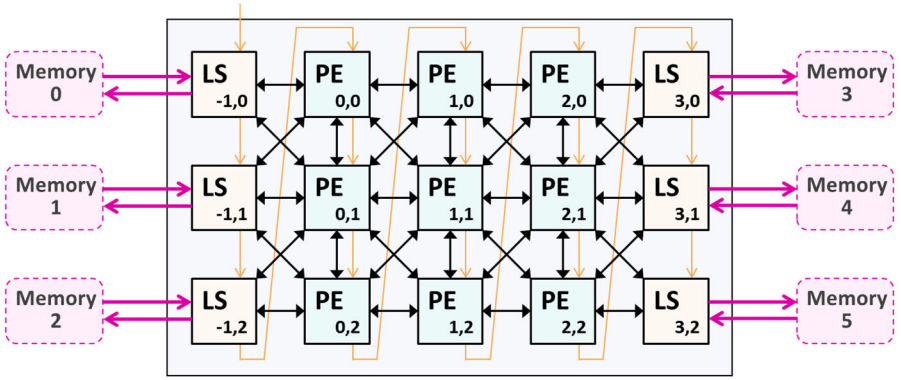
- ✓ Processor Team — **CGRA**
- ✓ Kumamoto Univ — **AI Engine** (ReNA)

System Software

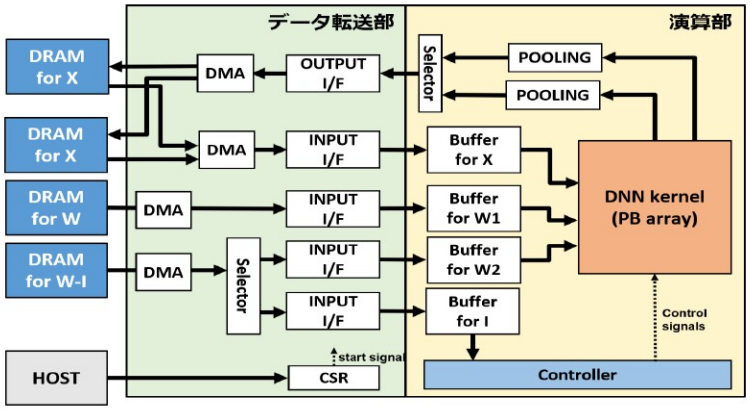
- ✓ RIKEN — RPC for FPGAs
- ✓ Tohoku Univ — neoSYCL (on Fugaku)

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 of Graph
- ✓ Processor Team — Hardwired MNIST
- ✓ JAIST — Sound rendering



Riken CGRA (coarse-grained reconfigurable array)



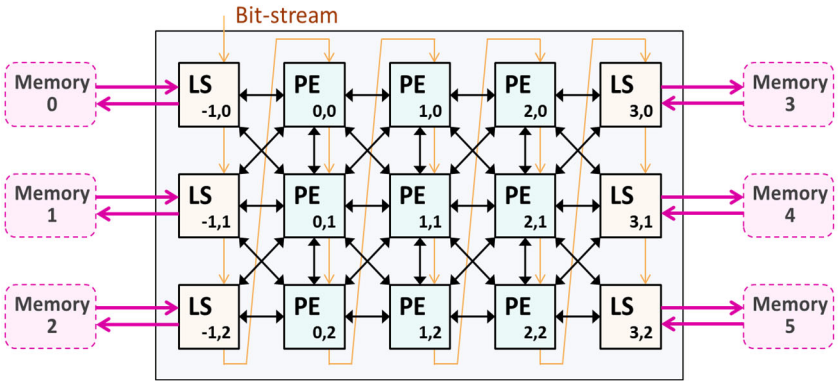
AI Engine, ReNA

on-going project

Design Space Exploration of CGRA (Riken)

FPGA emulator/overlay of coarse-grained reconfigurable array (CGRA) for HPC

- ✓ Processor Research Team, Riken R-CCS
- ✓ **Exploring design space of CGRA for ASIC**
  - ➢ Various configurations available with library modules such as FIFO, Mux, ALU
- ✓ **CGRA compiler** (by Tokyo university)
  - ➢ Data-flow graph (DFG) of a loop kernel in OpenMP
  - ➢ Place and route by Genetic Algorithm
  - ➢ Benchmarking (Stencil, Convolution, FFT, etc.)
- ✓ **Initial design completed**
  - ➢ System Verilog
  - ➢ Verified by RTL simulation
  - ➢ Preparing for FPGA-based implementation



Overall structure of CGRA (size: parameterized)

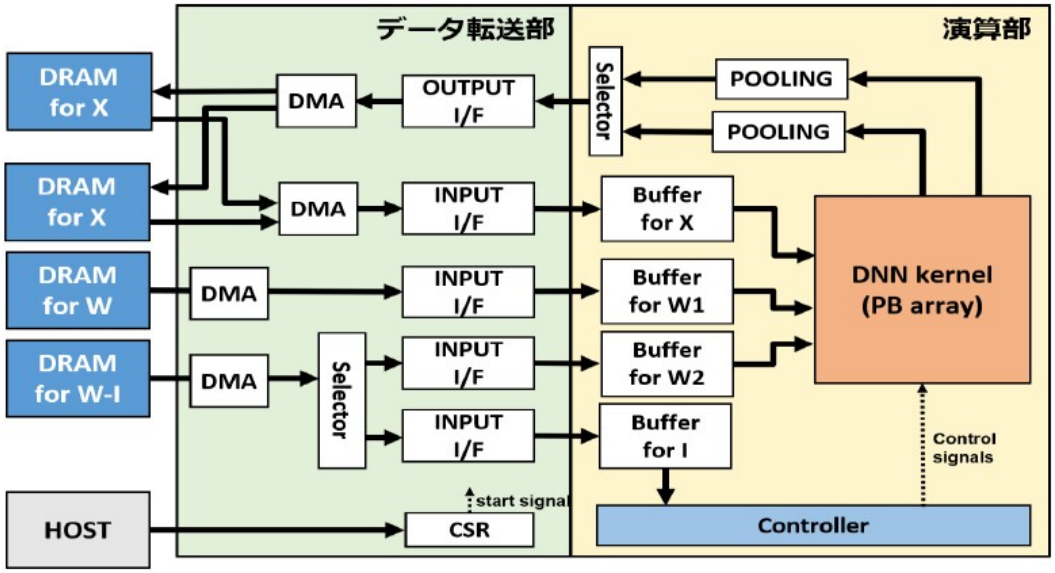
Mapping examples on CGRAs（16x16, 8x16）

on-going project

ReNA: Architecture for CNN Inference (Kumamoto U)

Transplant Inference processor ReNA developed for edge ASIC to FPGA

- ✓ Laboratory of Prof. Iida @ Kumamoto U
- ✓ **Achieve highly-scalable inference with multiple FPGAs**
  - ➢ Extend the processor over FPGAs using inter-FPGA network
- ✓ **64x64 systolic array**
  - ➢ FMA x 64^2 = 8192 parallel
  - ➢ Convolution and all-to-all computations optimized by dedicated mappings
  - ➢ Various models available
- ✓ **Initial implementation completed for single FPGA**
  - ➢ Verilog HDL



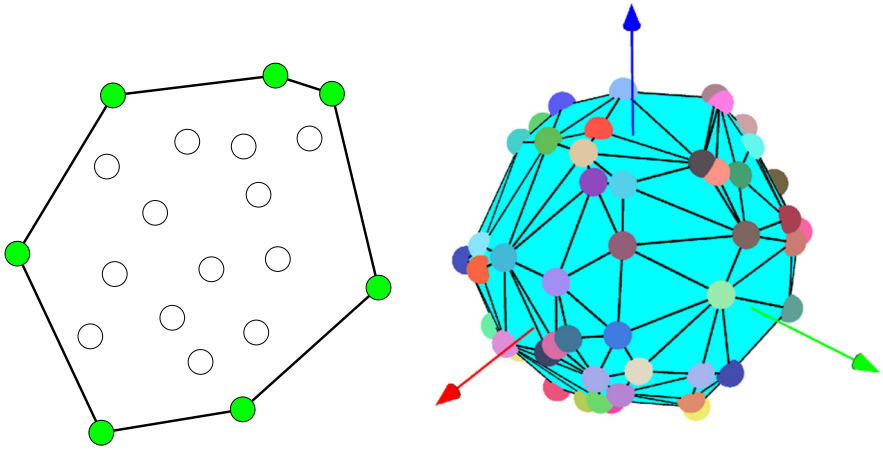
Architecture of ReNA

on-going project

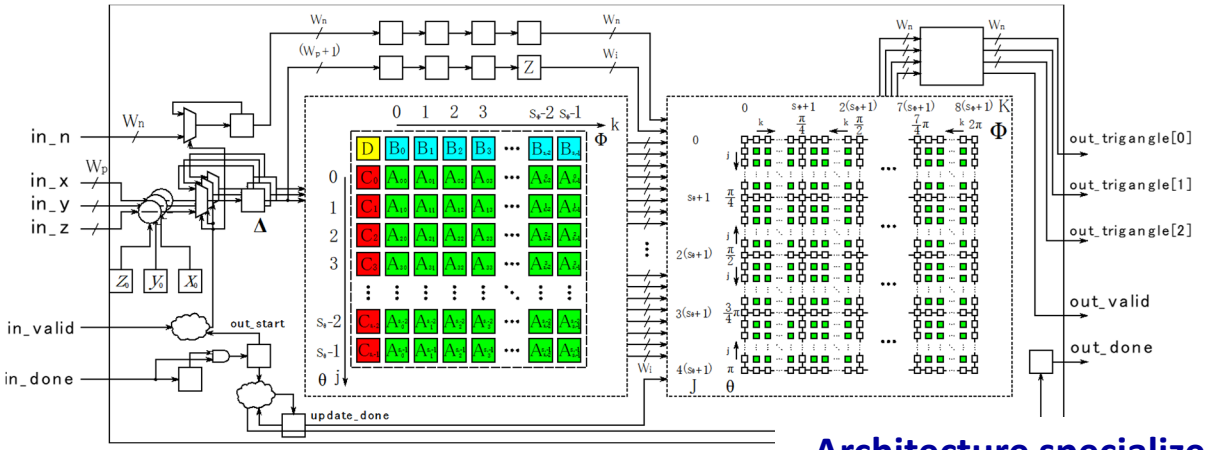
Architecture for Convex Hull Generation (Nagasaki U)

Acceleration of Convex Hull Generation with point clouds using multiple FPGAs

- ✓ Laboratory of Prof. Shibata @ Nagasaki U
- ✓ **Applications of Convex Hull**
  - ➢ Delaunay diagram construction/area estimation/registration/image processing, etc.
  - ➢ Object collision detection
  - ➢ Approximation of moving objects and obstacles in path planning
  - ➢ physics simulator
  - ➢ Real-time rendering of point clouds
- ✓ **Pipelining for higher throughput and lower latency than GPUs**
- ✓ **Initial implementation completed**
  - ➢ SystemVerilog
  - ➢ Comparison with Qhull software



Convex hull : The smallest convex set enclosing a point set



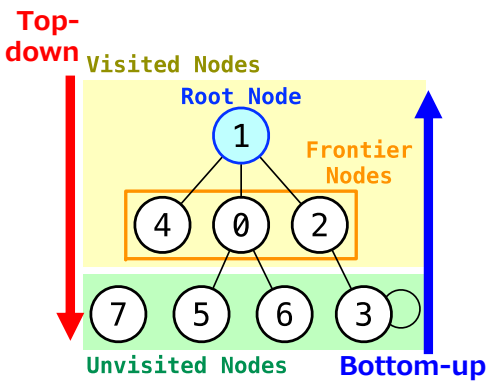
Architecture specialized for convex hull generation

on-going project

Specialized Hardware for BFS by HLS (Hiroshima city U)

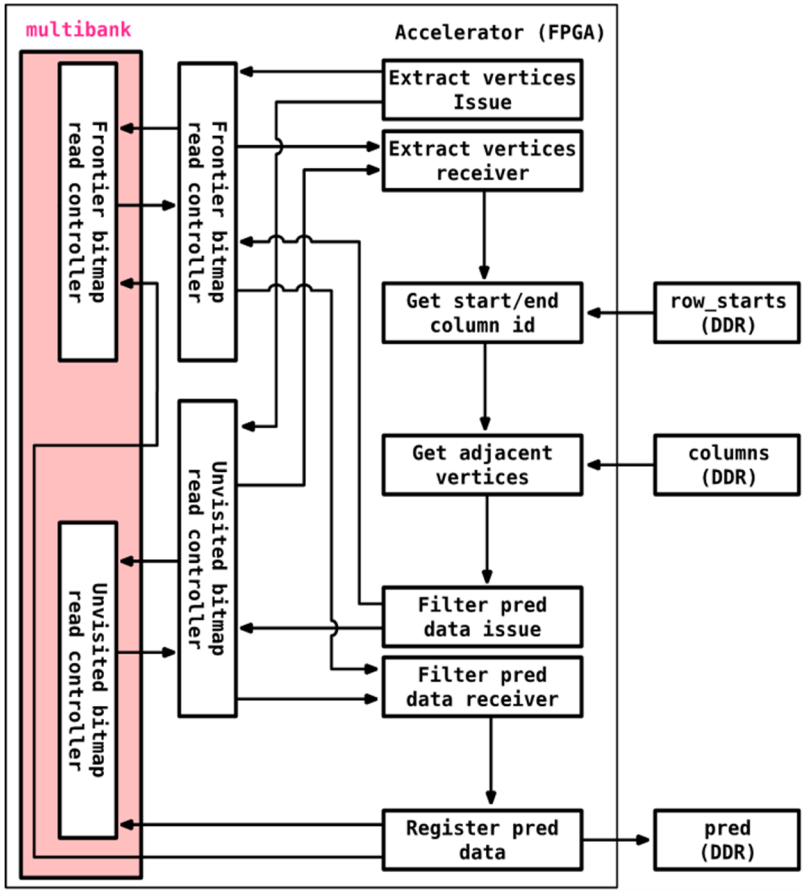
BFS Accelerator HyGTA

- ✓ Laboratory of Prof. Tanigawa @ Hiroshima city U
- ✓ **Hybrid Graph Traversal Accelerator**
  - ➢ Hybrid algorithm combining **Top-down** and **Bottom-up** searches
- ✓ **Implement by HLS, demonstrate and evaluate with FPGA**
- ✓ **Pipelining, latency hiding, efficient memory sub-system**
  - ➢ Pipelined BFS
  - ➢ Cache memory for adjacent-node data
  - ➢ Multi-banked bitmaps for visited-node record
  - ➢ Effective use of memory access patterns specific in BFS



Graph500 Ranking（BFS as of Nov, 2021）

RANK	MACHINE	SCALE	GTEPS
32	ENIAD （FPGA）	26	783.75



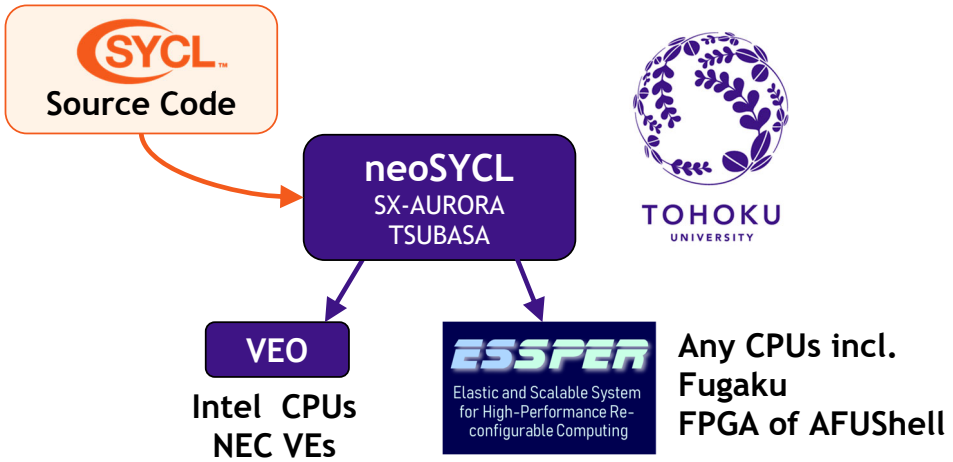
Specialized hardware for BFS accelerator “HyGTA”

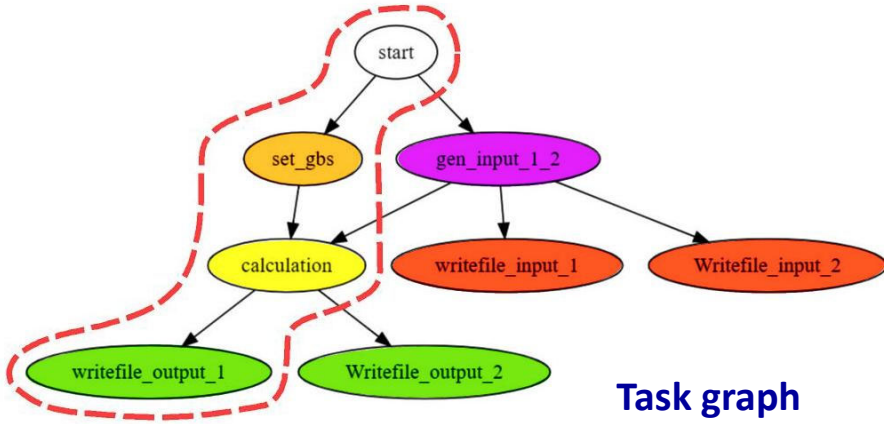
on-going project

Task off-loading to FPGAs by own SYCL (Tohoku U)

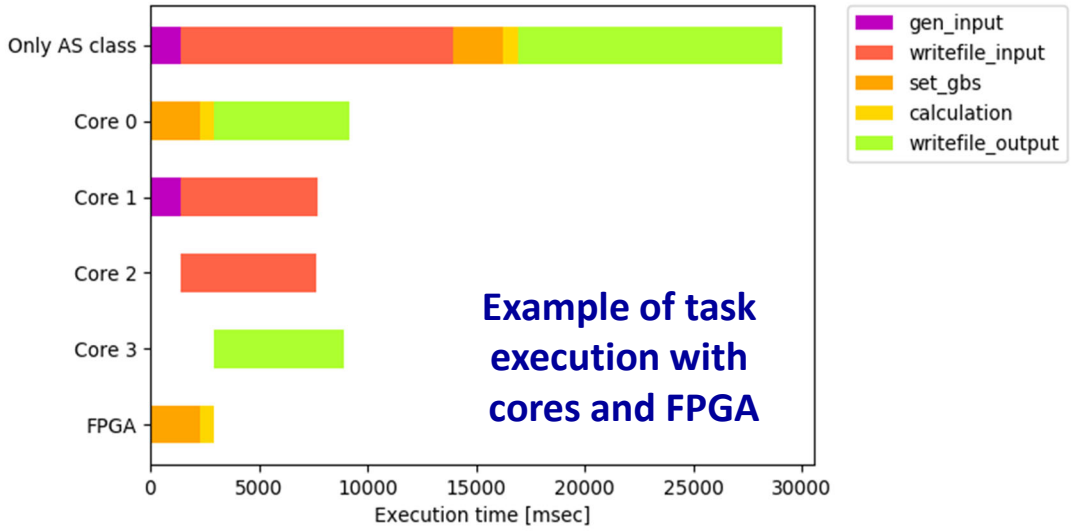
neoSYCL : yet another SYCL implementation

- ✓ Laboratory of Prof. Takizawa @ Tohoku U
- ✓ neoSYCL originally developed for NEC Vector Processor
- ✓ Support FPGA and AFUShell of ESSPER
- ✓ Dynamic task scheduler
- ✓ Tasks can be off-loaded from Fugaku to FPGAs.





Task graph



Example of task execution with cores and FPGA

Summary

Goal	**Explore new architectures & programming env.** with reconfigurable HPC
This project	**ESSPER**: Elastic and Scalable System for High-Performance Reconfigurable Computing
PoC for	- *Programming with sufficient customizability* - *Techniques to scale performance* - *Interoperability with existing systems (Fugaku)*

Future work

- ✓ **More applications with multi-FPGAs**
- ✓ System software for FPGAs incl. resource management
- ✓ Development of new FPGA board (Intel Agilex-M @7nm)

Looking forward to collaborating with you!

Hiring researchers & internship students