Plans for SRO development by SPADI-Alliance in Japan

TAKU GUNJI

CENTER FOR NUCLEAR STUDY

THE UNIVERSITY OF TOKYO

SPADI-Alliance

SPADI Alliance

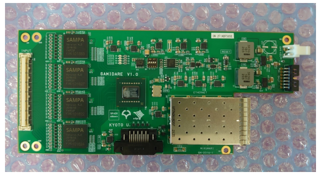
Signal processing and data acquisition infrastructure alliance

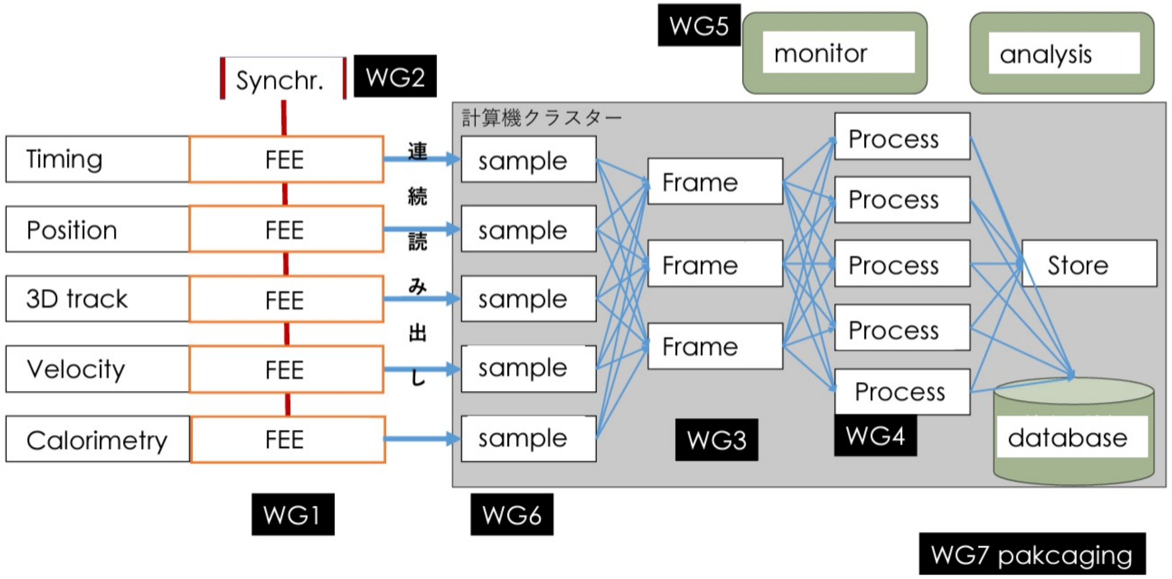
- References:
  - Collaboration meeting in January (slides)
  - SRO XI workshop at Hawaii in December, 2023 (4 presentations from SPADI-Alliance)
- Backgrounds:
  - Streaming DAQ is a common need in many facilities in Japan (RIBF, RCNP, J-PARC, …)
  - No enough people in the facilities to develop electronics, FEE, and DAQ for SRO.
  - Goal is to develop the entire SRO system (electronics, FEE, SRO software, analysis software) and standardize the system to share the identical system with many facilities.
- SPADI-Alliance was established in Japan for standardization of the SRO system
  - >120 researchers and 20 institutes from different experiments and from different facilities

Working Groups

- WG1: ASIC and FEE
  - ASICs for MPPC and Gaseous detector
  - FE boards for MPPC, DC, Si, TPC
  - TDC (“AMANEQ”), waveform digitizer
- WG2: Timing distribution & data transfer
  - “MIKUMARI” system (light FPGA)
  - SiTCP, Fakernet
- WG3: NestDAQ (NEtwork based Streaming DAQ)
  - fairMQ + Redis based scalable system
  - WebUI (configuration, control, monitoring)
- WG4: Event processing
  - Event reconstruction, calibration
  - Hardware accelerators (FPGA, GPU) and AI/ML
- WG6: Computing
- WG7: Packaging
- Analysis SW (“Artemis”)

SAMPA FEE





WG1
Frontend Electronics
Streaming type
Charge ASD board
Voltage ASD board
WF Digitizer board
Control Firmware dev.

WG2
Clock synch. / Data Transfer
General Clock Synch.
High throughput
Intra-board transfer

WG3
Acquisition software framework (NestDAQ + …)
Streaming type
FairMQ-based Scalable DAQ
Sampling, Time frame build,
Event build, Monitoring…
Format

WG4
Event processing
Acceleration using GPU/FPGA
Zero suppression
Calibration, Clustering,
Tracking, PID,

WG5
User Interface
Control, Monitor, Configure,

Trial with
SlowDash

WG6
Computing infrastr.
High throughput
Large volume
Flow and Archive
Power consumption
Interconnect
Networking
…

WG7
Packaging
Standalone system
Popularization
Standardization
Market research
User feedback

Analysis

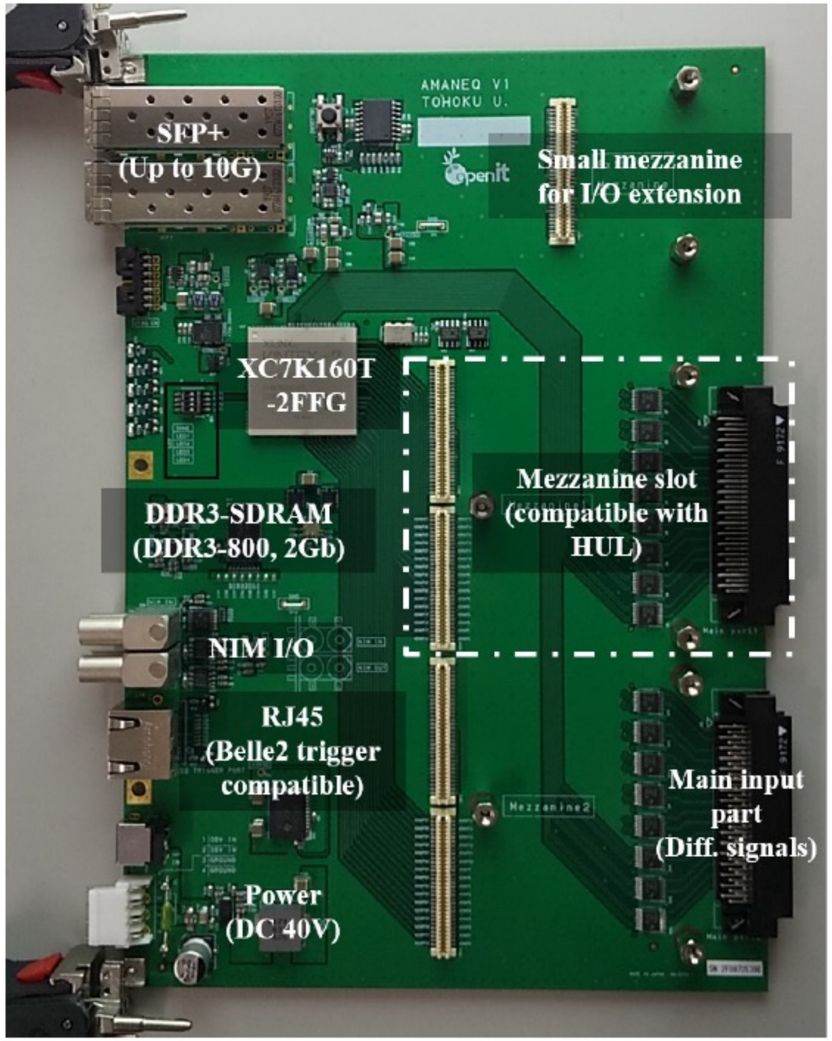
Trial with
Artemis

Streaming type TDC: AMANEQ

- 6U size
- Kintex7 with speed grade -2
  - Transceiver bandwidth up to 10Gbps
  - 64-bit data word, 156M words/sec
- Has two mezzanine slot
  - High resolution (HR-TDC, ~30ps precision) 32 ch/board
    - HR-TDC block is implemented on the mezzanine FPGA
  - Low resolution (LR-TDC, 1ns precision) 128 ch/board

- Belle2 trigger port (master clock)
  - Has a jitter cleaner to clean up the master clock
- DDR3-SDRAM as a de-randomizer
  - DDR3-800 with 16-bit bus width. 2 Gb
  - It allows us to use spill off time for data transfer

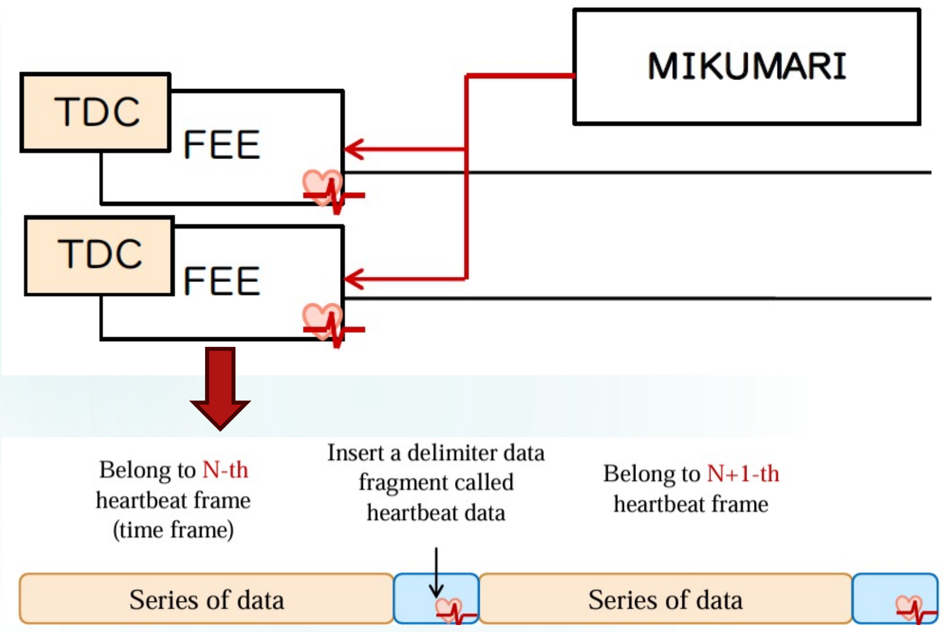
- Clock synchronization with MIKUMARI protocol
- Framing with heartbeat method

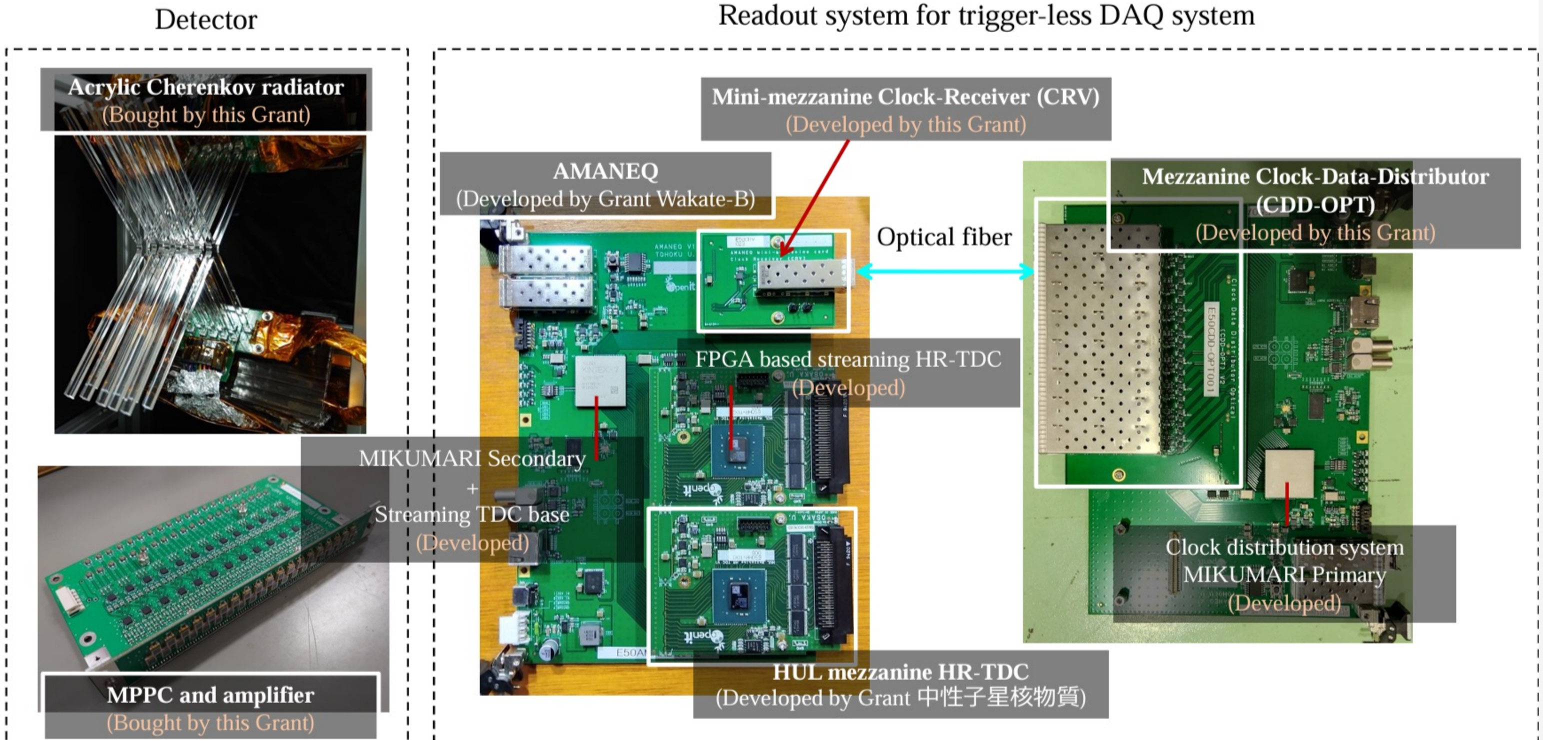


https://indico.bnl.gov/event/20010/contributions/79154/attachments/51297/87719/Hawaii2023-SROXI-honda.pdf

MIKUMARI as timing distribution

Synchronization
Simple & light-weight clock-data-recovery
Frequency synchronization
(MIKUMARI: 水分, R. Honda, IEEE TNS, 70 (6), 1102 (2023).


MIKUMARI
TDC
FEE
TDC
FEE
Belong to N-th heartbeat frame (time frame)
Insert a delimiter data fragment called heartbeat data
Belong to N+1-th heartbeat frame
Series of data
Series of data


Detector
Readout system for trigger-less DAQ system
Acrylic Cherenkov radiator
(Bought by this Grant)
MPPC and amplifier
(Bought by this Grant)
Mini-mezzanine Clock-Receiver (CRV)
(Developed by this Grant)
AMANEQ
(Developed by Grant Wakate-B)
Optical fiber
Mezzanine Clock-Data-Distributor (CDD-OPT)
(Developed by this Grant)
FPGA based streaming HR-TDC
(Developed)
MIKUMARI Secondary
+
Streaming TDC base
(Developed)
Clock distribution system
MIKUMARI Primary
(Developed)
HUL mezzanine HR-TDC
(Developed by Grant 中性子星核物質)

25 ps timing resolution including the synchronization precision.

More detailed slides (by R. Honda at the SRO XI workshop) https://indico.bnl.gov/event/20010/contributions/79154/attachments/51297/87719/Hawaii2023-SROXI-honda.pdf

nestDAQ

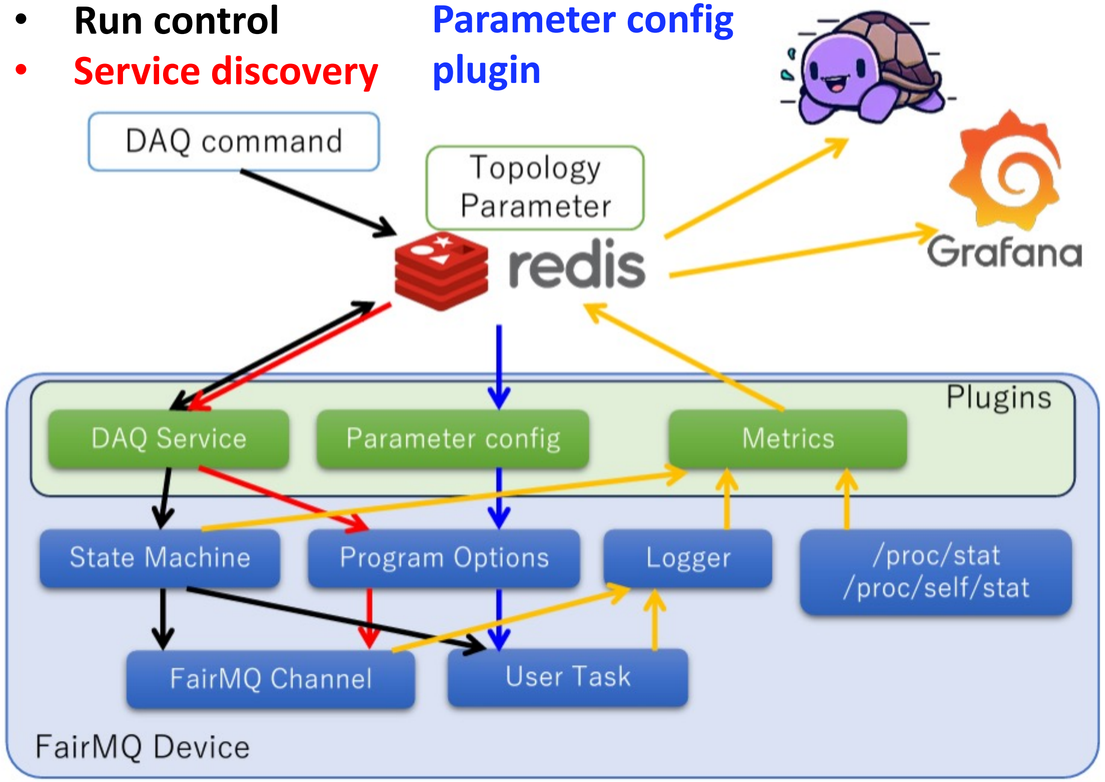
More detailed slides (by Igarashi at the SRO XI workshop)
https://indico.bnl.gov/event/20010/contributions/79156/attachments/51299/87721/streamingws20231128_igarashi.pdf

- nestDAQ (network based streaming DAQ)
  - FairMQ (data transport based on zeroMQ, state machine control, plugins) + Redis (process management and control, in-memory access and fast response)

DAQ service plugin
- Run control
- Service discovery

Metrics plugin

Parameter config plugin

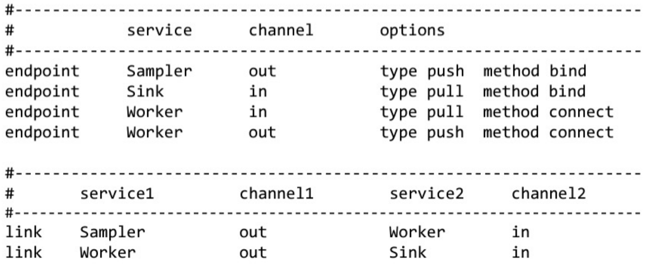


Semi-automatic topology generation based on service registry by Redis

- The database provides information about each process grouped as a function (service), its data channel-ports and their connections

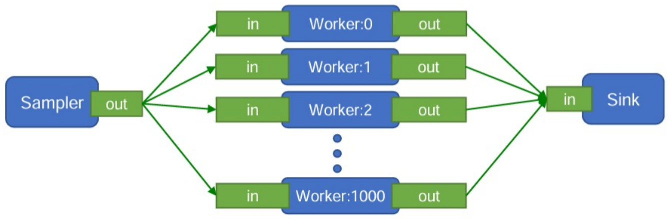
Example: An arbitrary number of worker processes

Topology data on the database



```
#-----------------------------------------------------------------
#            service     channel       options
#-----------------------------------------------------------------
endpoint     Sampler     out           type push  method bind
endpoint     Sink        in            type pull  method bind
endpoint     Worker      in            type pull  method connect
endpoint     Worker      out           type push  method connect

#-----------------------------------------------------------------
#       service1          channel1         service2     channel2
#-----------------------------------------------------------------
link    Sampler           out              Worker       in
link    Worker            out              Sink         in
```

Configured topology structure

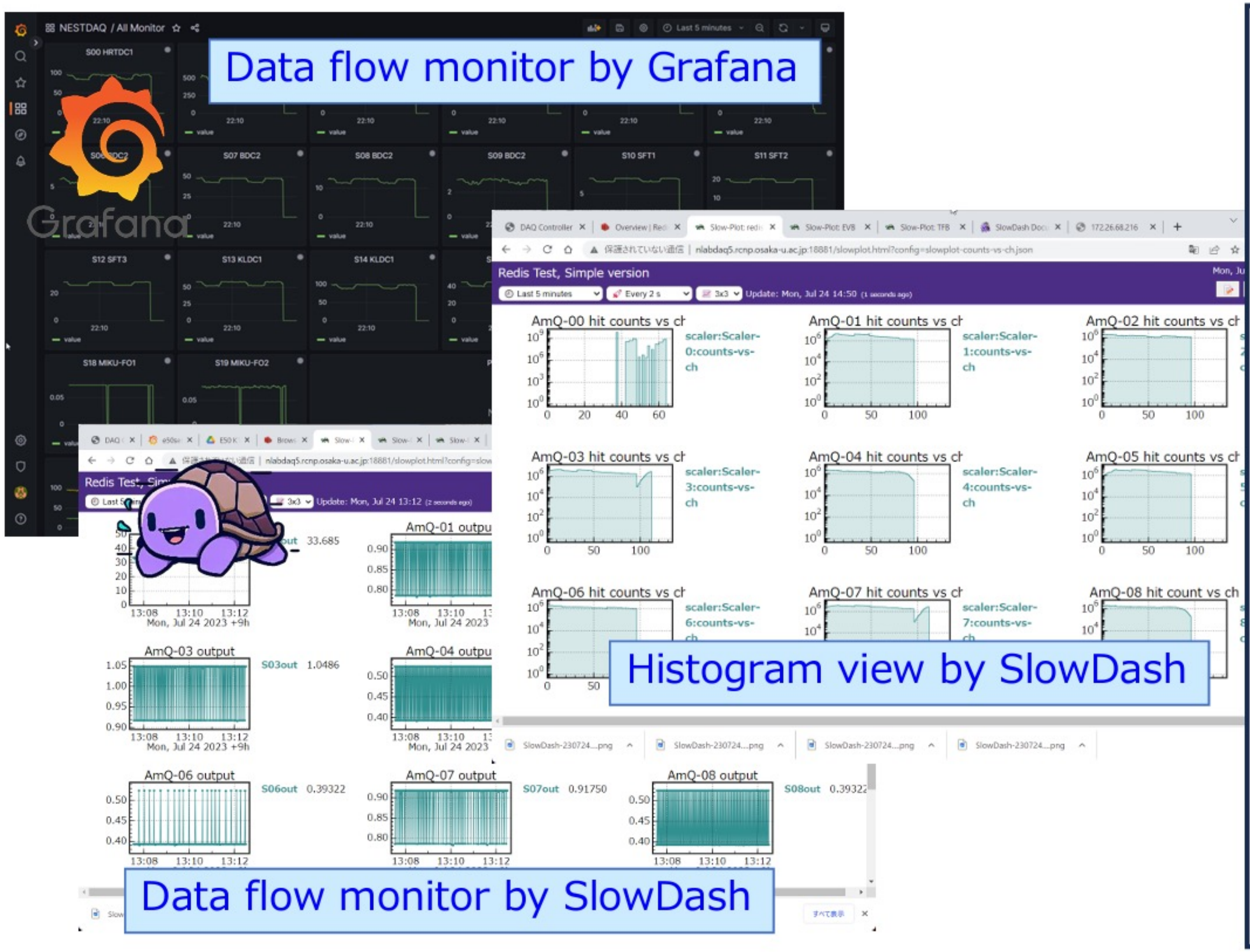


Current developments:

- Triggered readout
- Unification of Data-format
- Mini-booking tool (API for slowdash monitoring)
- Log collector
- API for channel mapping
- Evaluation of scalability

WebUI

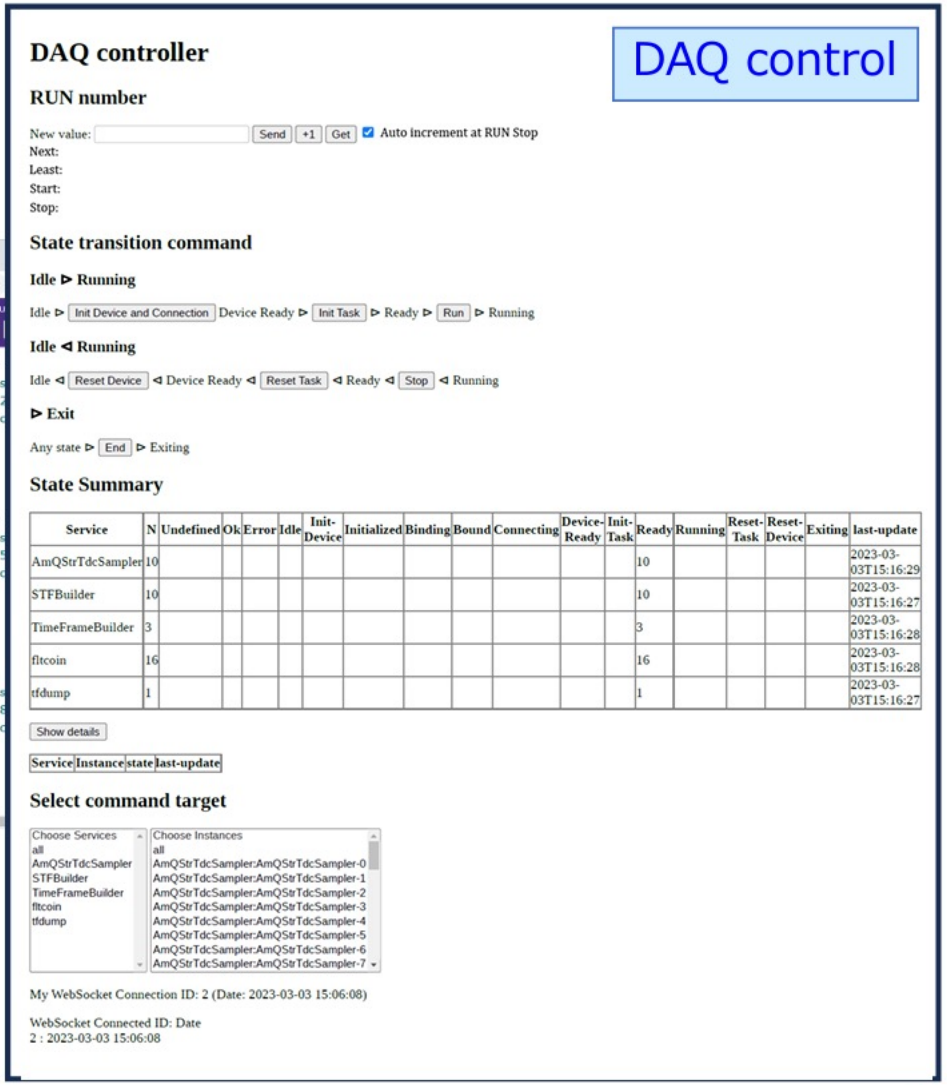
More detailed slides (by Igarashi at the SRO XI workshop)
https://indico.bnl.gov/event/20010/contributions/79156/attachments/51299/87721/streamingws20231128_igarashi.pdf



Data flow monitor by Grafana

Histogram view by SlowDash

Data flow monitor by SlowDash



DAQ control

DAQ controller

RUN number

New value: [ ] Send +1 Get ☑ Auto increment at RUN Stop
Next:
Least:
Start:
Stop:

State transition command

Idle ⊳ Running

Idle ⊳ Init Device and Connection Device Ready ⊳ Init Task ⊳ Ready ⊳ Run ⊳ Running

Idle ⊲ Running

Idle ⊲ Reset Device ⊲ Device Ready ⊲ Reset Task ⊲ Ready ⊲ Stop ⊲ Running

⊳ Exit

Any state ⊳ End ⊳ Exiting

State Summary

Service	N	Undefined	Ok	Error	Idle	Init-Device	Initialized	Binding	Bound	Connecting	Device-Ready	Init-Task	Ready	Running	Reset-Task	Reset-Device	Exiting	last-update
AmQStrTdcSampler	10												10					2023-03-03T15:16:29
STFBuilder	10												10					2023-03-03T15:16:27
TimeFrameBuilder	3												3					2023-03-03T15:16:28
fltcoin	16												16					2023-03-03T15:16:28
tfdump	1												1					2023-03-03T15:16:27

Show details

Service	Instance	state	last-update

Select command target

Choose Services
all
AmQStrTdcSampler
STFBuilder
TimeFrameBuilder
fltcoin
tfdump

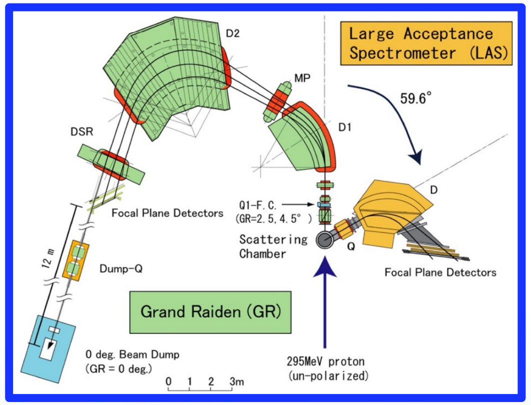
Choose Instances
all
AmQStrTdcSampler:AmQStrTdcSampler-0
AmQStrTdcSampler:AmQStrTdcSampler-1
AmQStrTdcSampler:AmQStrTdcSampler-2
AmQStrTdcSampler:AmQStrTdcSampler-3
AmQStrTdcSampler:AmQStrTdcSampler-4
AmQStrTdcSampler:AmQStrTdcSampler-5
AmQStrTdcSampler:AmQStrTdcSampler-6
AmQStrTdcSampler:AmQStrTdcSampler-7

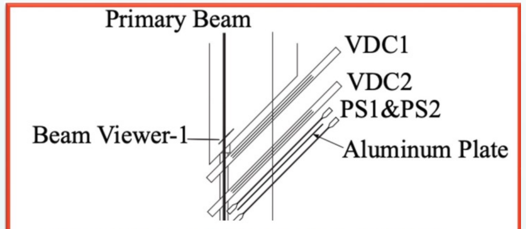
My WebSocket Connection ID: 2 (Date: 2023-03-03 15:06:08)

WebSocket Connected ID: Date
2 : 2023-03-03 15:06:08

* SlowDash is a web based visualization tool developed by S. Enomoto (Washington U.)

SRO tests in Japan (2023)

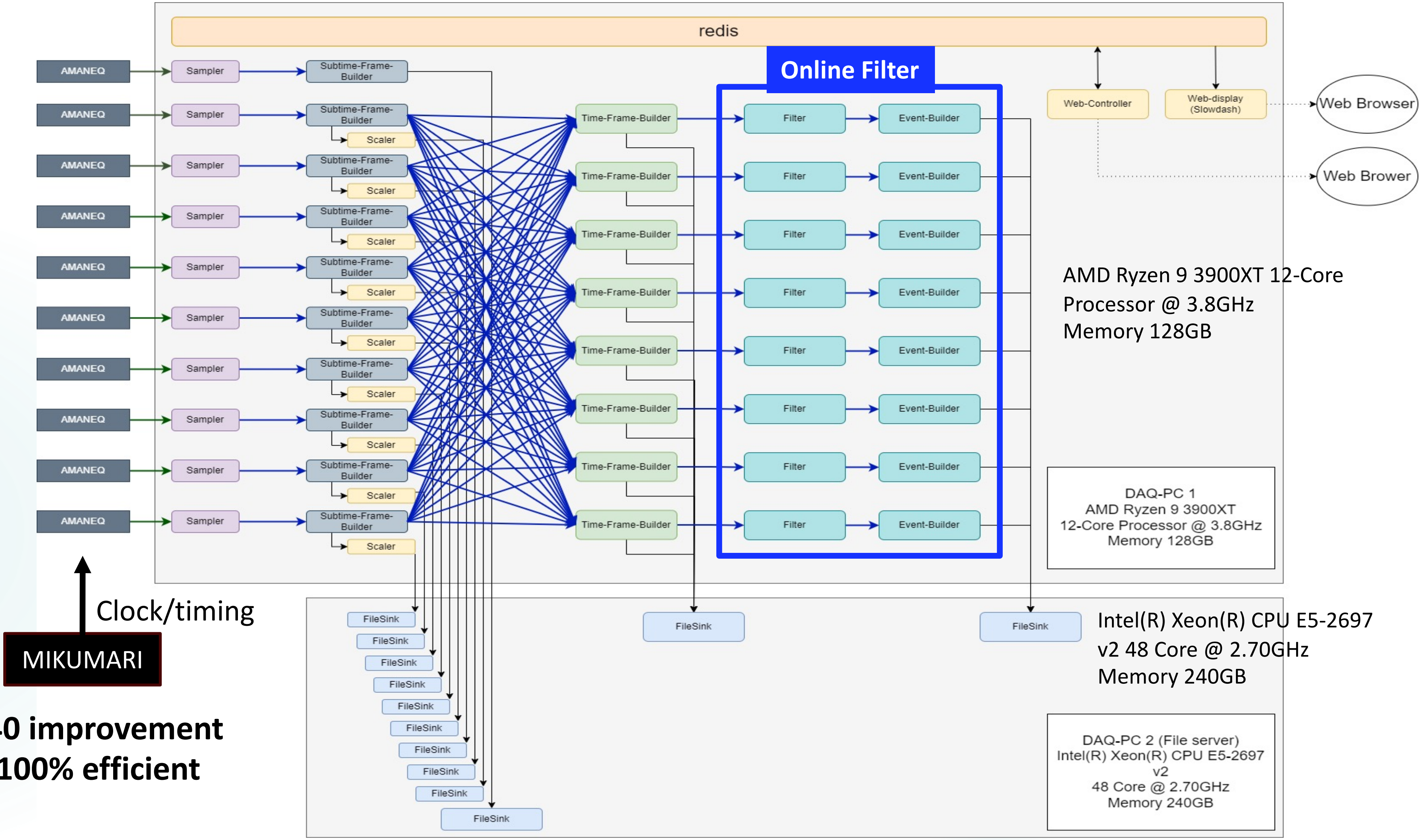




VDC (X1:192, U1:208, X2:192, U2:208)

PS (PS1: 2, PS2: 2)

RF timing, Current integrator



AMD Ryzen 9 3900XT 12-Core Processor @ 3.8GHz
Memory 128GB

DAQ-PC 1
AMD Ryzen 9 3900XT
12-Core Processor @ 3.8GHz
Memory 128GB

Intel(R) Xeon(R) CPU E5-2697 v2 48 Core @ 2.70GHz
Memory 240GB

DAQ-PC 2 (File server)
Intel(R) Xeon(R) CPU E5-2697 v2
48 Core @ 2.70GHz
Memory 240GB

Throughput 200Mbps (x40 improvement from past DAQ system). 100% efficient for 100-200 kcps

J-PARC E50

- J-PARC E50 (charmed baryon spectroscopy) could be used as a testbed for ePIC.

＊Streaming DAQ: Only TDC information

⇒ 20,000–25,000 ch (25 GB/spill ⇒ 12 GB/sec.)
+ FEE: 10Gbps network
+ Timing synchronization (MIKUMARI)

- MPPC detector: 15,000–20,000 ch
  - Scintillating fiber trackers
  - RICH, Beam-RICH, Vth AC

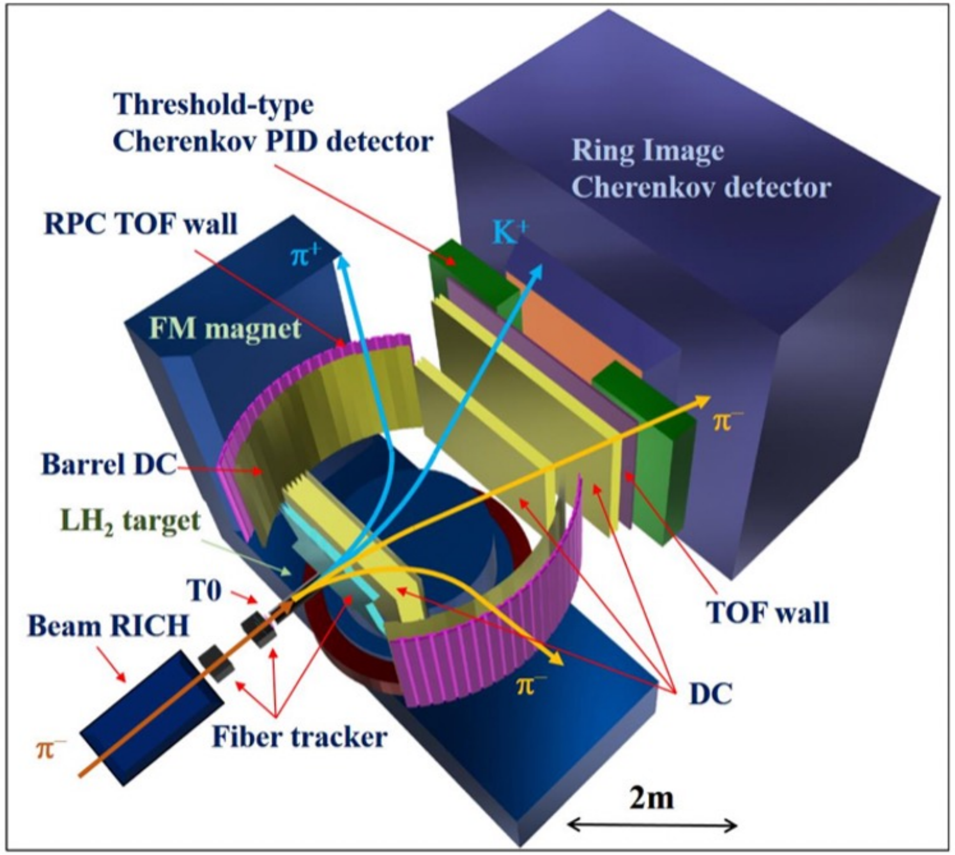
⇒ CIRASAME (ASIC: CITIROC)
- 128 ch/board
- Low-resolution TDC (ΔT_{LSB} ~ 1 ns)

- Timing detector: ~1,000 ch
  - T0, RPC, TOF: Amp/PMT + Discriminator

⇒ AMANEQ board (HR-TDC mezzanine)
- 64 ch/board
- High-resolution TDC (ΔT_{LSB} ~ 20 ps)

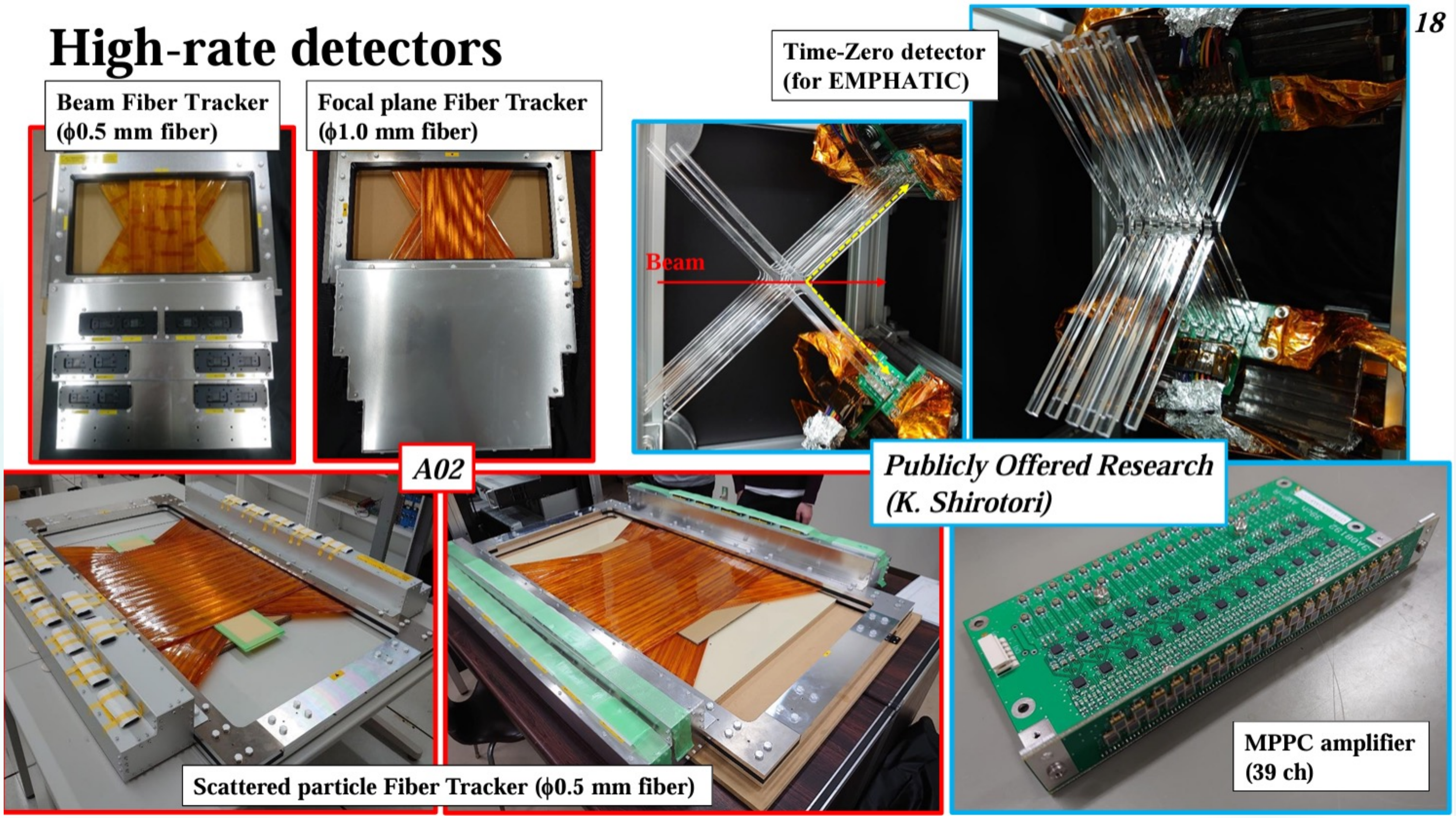
- Drift chamber: ~4,000 ch

⇒ ASAGI(ASD) card + AMANEQ board (DC mezzani
- 32 ch ASD card + 128 ch/board
- Low-resolution TDC (ΔT_{LSB} ~ 1 ns)



30 MHz beam rate, 5% reaction rate (~1.5MHz IR rate)
Only 4-5 particles per reaction → similar conditions as EIC

E50: High Rate detectors



High-rate detectors

18

Beam Fiber Tracker (ϕ0.5 mm fiber)

Focal plane Fiber Tracker (ϕ1.0 mm fiber)

Time-Zero detector (for EMPHATIC)

Beam

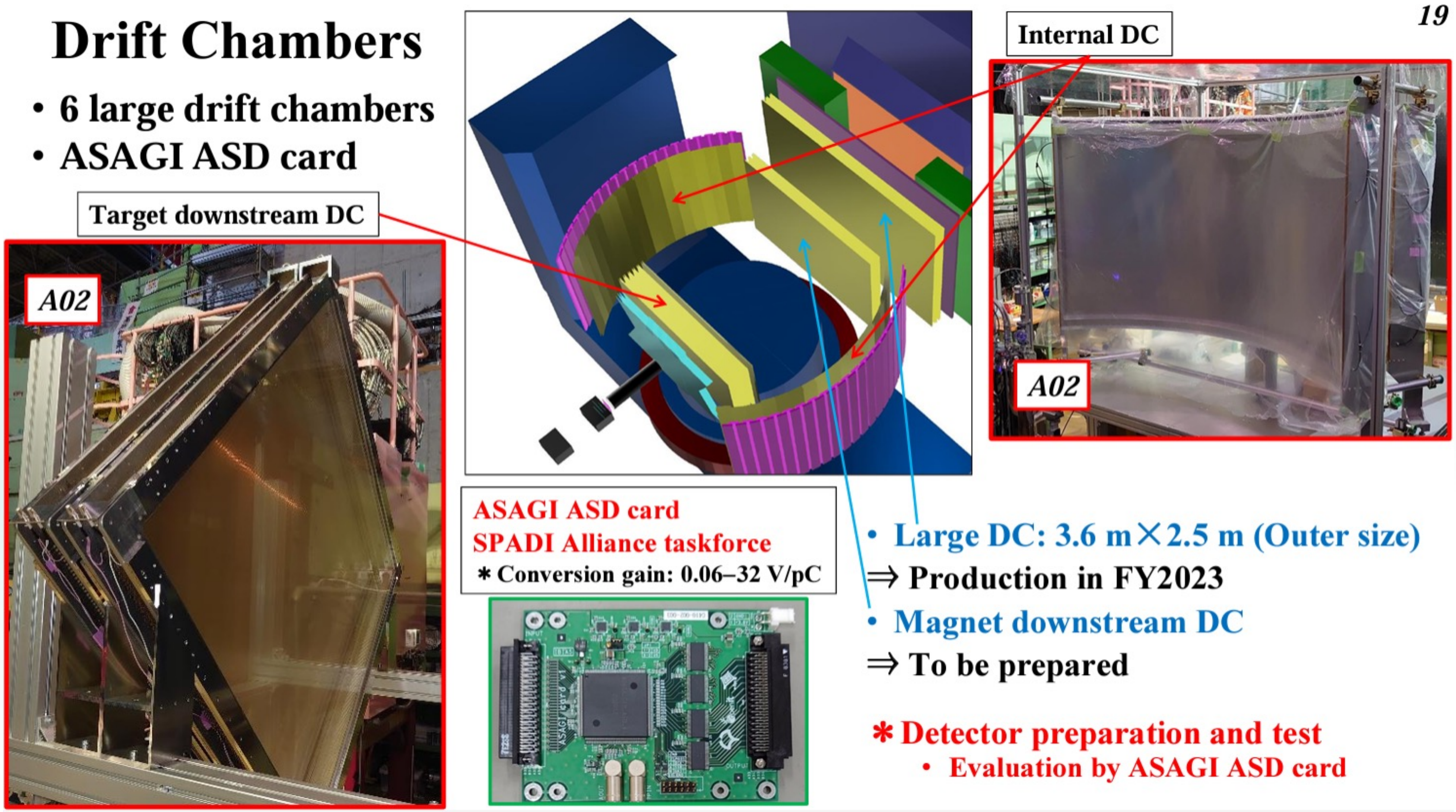
A02

Publicly Offered Research (K. Shirotori)

Scattered particle Fiber Tracker (ϕ0.5 mm fiber)

MPPC amplifier (39 ch)

E50: Drift chambers



19

Drift Chambers

- 6 large drift chambers
- ASAGI ASD card

Target downstream DC

Internal DC

ASAGI ASD card
SPADI Alliance taskforce
＊Conversion gain: 0.06–32 V/pC

- Large DC: 3.6 m×2.5 m (Outer size)

⇒ Production in FY2023

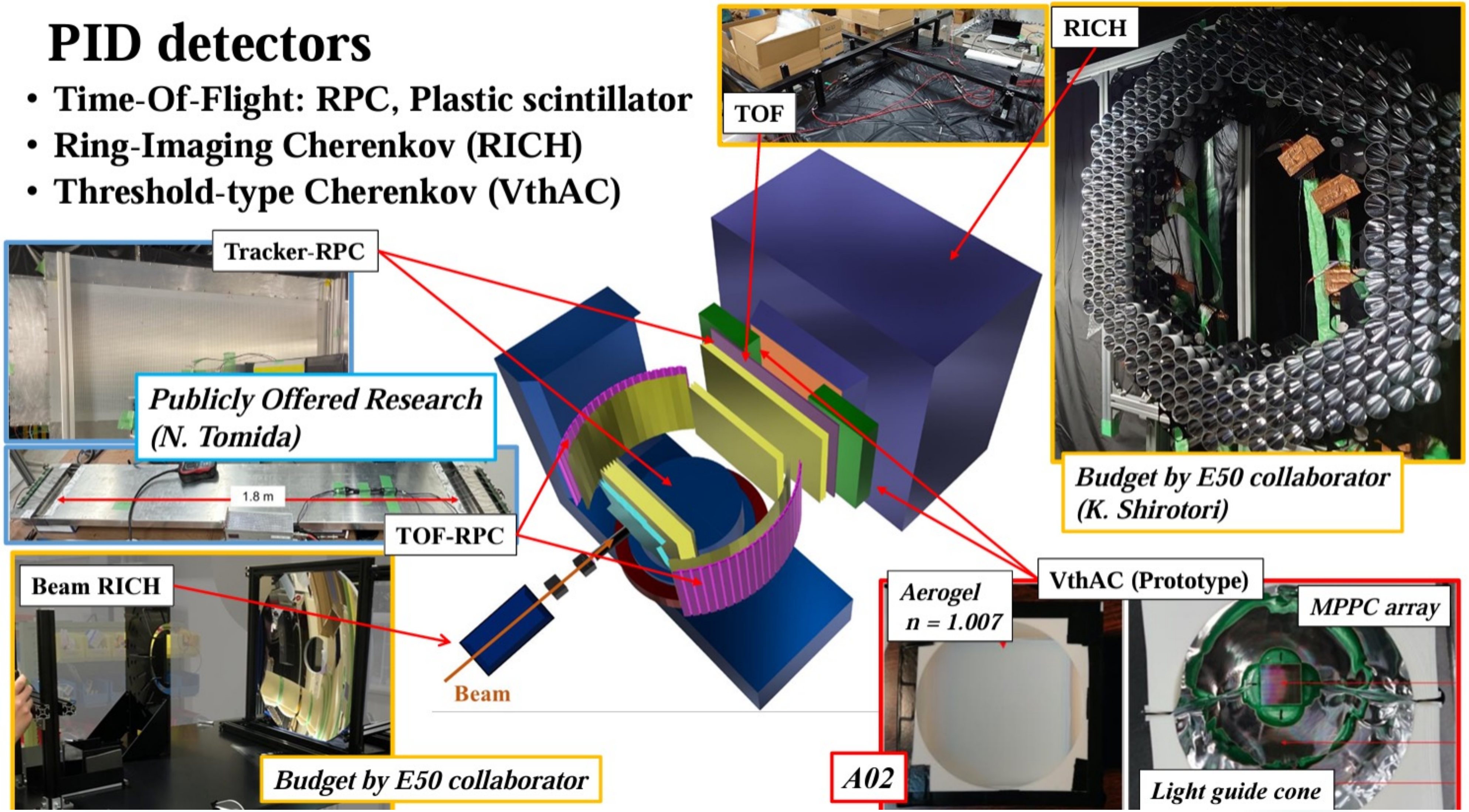
- Magnet downstream DC

⇒ To be prepared

＊Detector preparation and test

- Evaluation by ASAGI ASD card

E50: PID detectors



PID detectors

- Time-Of-Flight: RPC, Plastic scintillator
- Ring-Imaging Cherenkov (RICH)
- Threshold-type Cherenkov (VthAC)

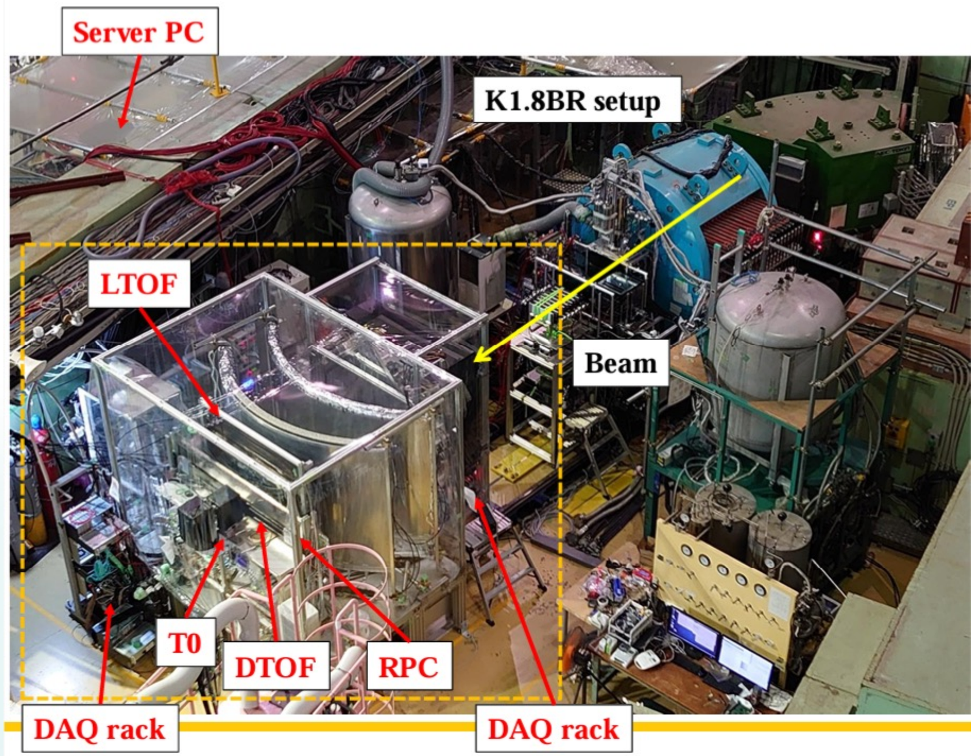
Publicly Offered Research (N. Tomida)

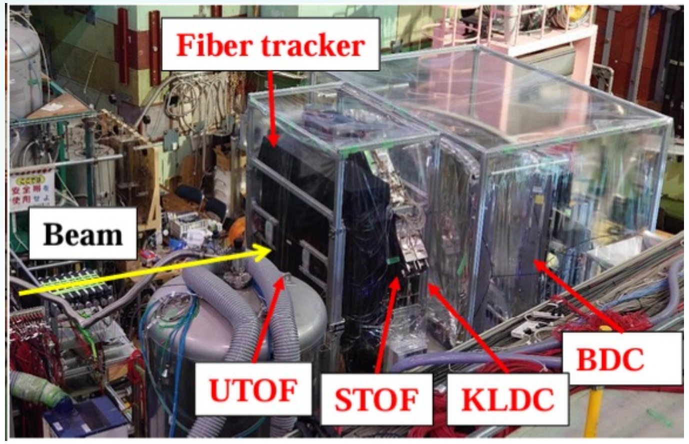
Budget by E50 collaborator (K. Shirotori)

Budget by E50 collaborator

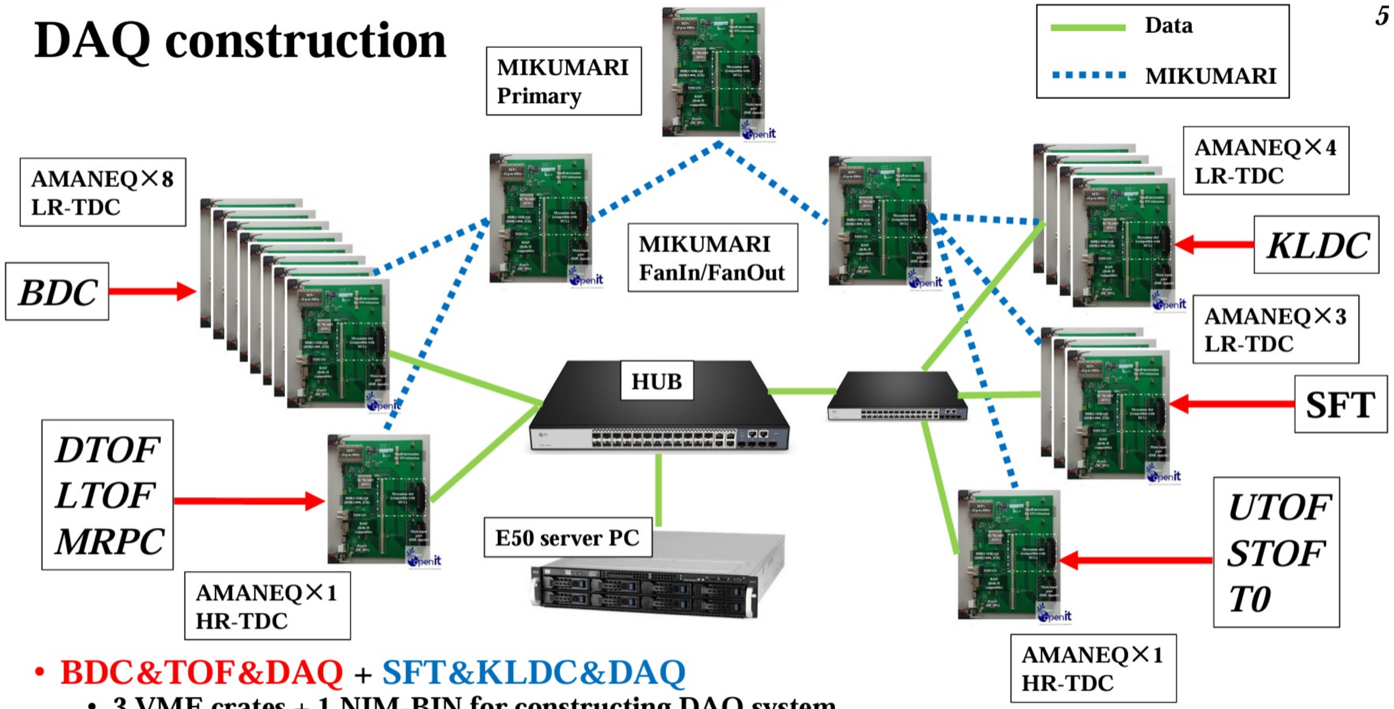
A02

J-PARC E50 as it is now





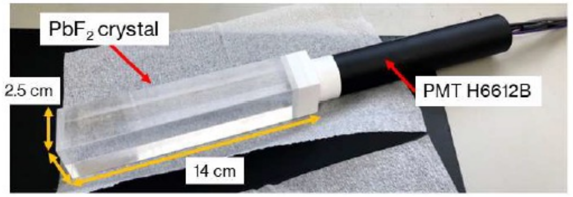
DAQ construction



- BDC&TOF&DAQ + SFT&KLDC&DAQ
  - 3 VME crates + 1 NIM-BIN for constructing DAQ system
- MIKUMARI FanIn/FanOut for synchronizing 20 AMANEQ modules

Next beam time at E50

- Next beamtime (not dedicated to E50 but we can run parasitically) will be from middle of April to the end of May.
  - Kaon/pion (=1/2) mixed beams. <1M/2s spill
  - E73 will be the main-user of this beamtime and E73 calorimeter will be installed upstream of E50.
    - Beam will not be so clean…
- Plans for the SRO development
  - Validation of full chain of SRO readout
  - Saving raw TF data (to replay offline)
  - Tests of various online filter (CPU and GPU)
    - Coincidence filter (T0, TOF, SFT)
    - Event builder (narrow time window)
    - Clustering and tracking (DC)
    - Association between detectors
    - Online PID and online tracking



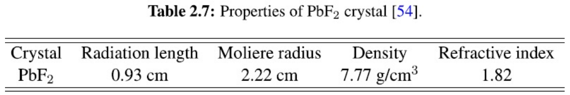
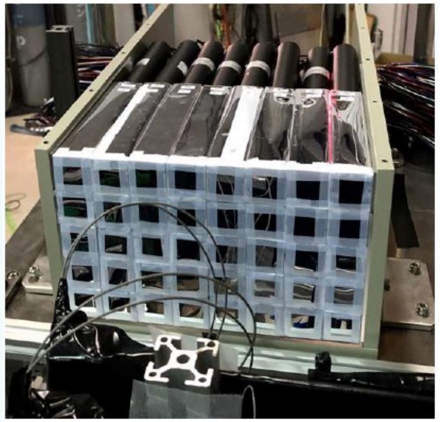
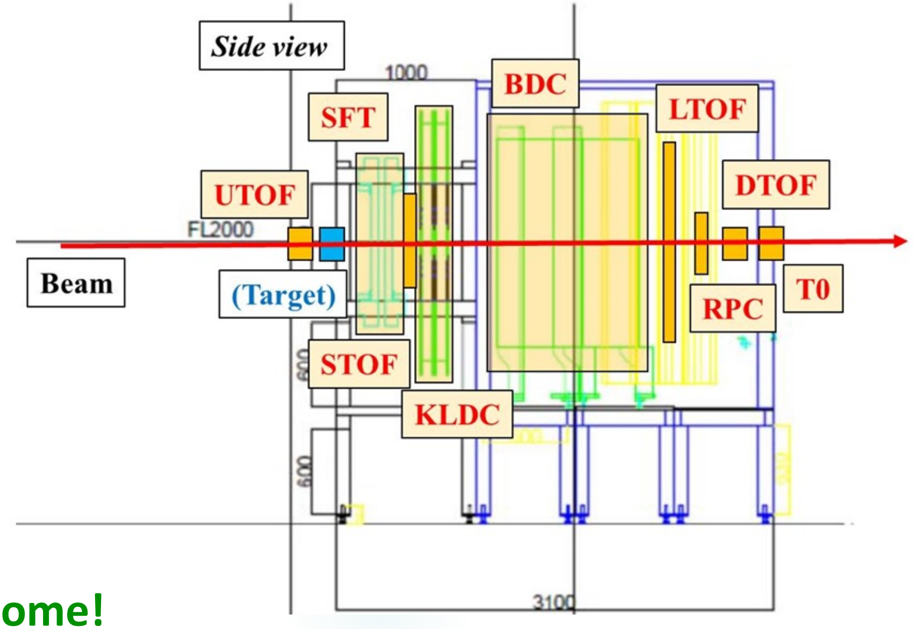


Table 2.7: Properties of PbF$_2$ crystal [54].

Crystal	Radiation length	Moliere radius	Density	Refractive index
PbF$_2$	0.93 cm	2.22 cm	7.77 g/cm^3	1.82





Any collaboration is more than welcome!

Possible contributions for ePIC

- It is beneficial to work together with ePIC in order to make more robust SRO system by exchanging the expertise and knowledge.
- For the moment, human resources for ePIC dedicated in SPADI-A is very limited.
  - (It is good if our activities improve something in SRO for ePIC.)
- Possible areas, where SPADI-A could contribute:
  - Timing distribution system
    - Development of ePIC-GTM system and MIKUMARI (if there are rooms for using MIKUMARI)
  - nestDAQ
    - Could be a central DAQ system for ePIC?
  - Online processing
    - AI/ML and HLS for FPGA on FELIX cards
    - GPU processing on echelon 0 or 1 or 2 (wherever applicable)
  - Providing testbed system to develop SRO DAQ with real hardware and software
    - Propose to use J-PARC E50 experiment (and BELLE2 – will be discussed) as the testbed of Streaming DAQ for ePIC? Of course, there are a lot of differences…

Possible contributions for ePIC

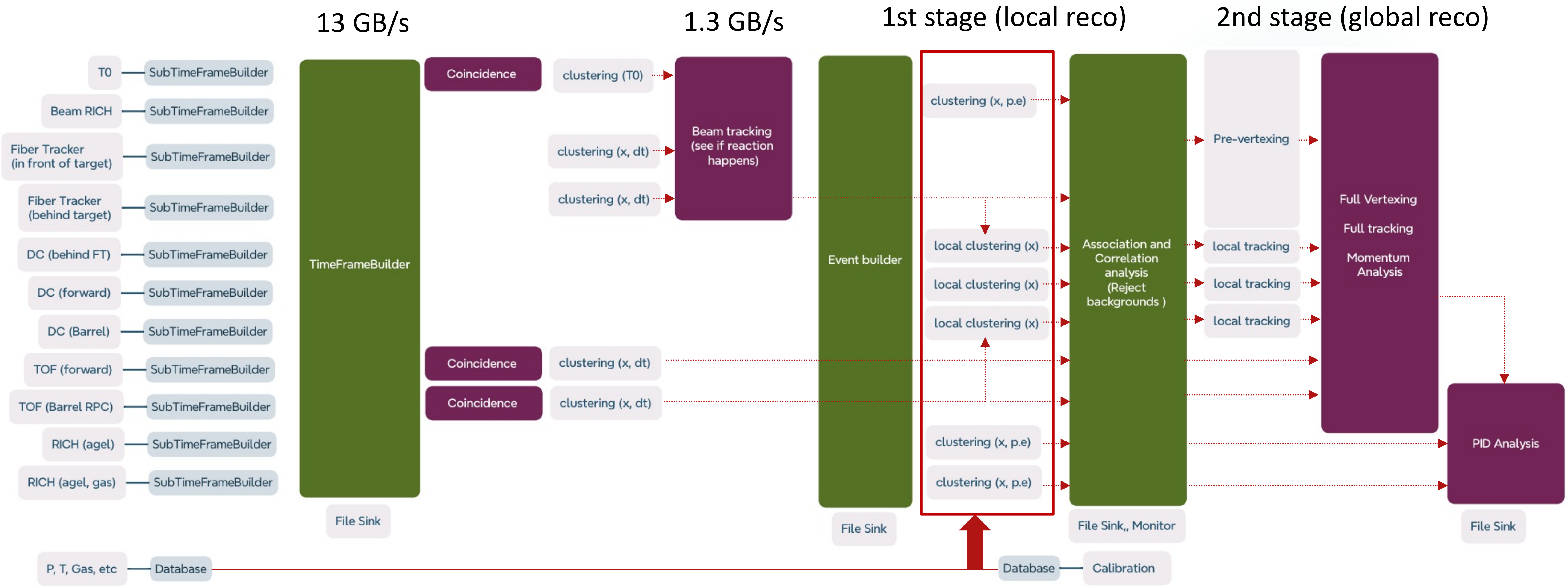
- We still have some differences between our system and ePIC system. We need to develop/check several things to adopt our SRO to ePIC.
  - We have to check the compatibilities even if specs and requirements for WG1 (ASIC, FEC) and WG2 (transmission) are different.
    - We are reading out only TDC. No waveform digitizer has been readout.
      - We have a plan to develop our SRO to handle waveform digitizer.
    - Our system defines HBFrames in AMANEQ (FPGA in TDC module).
    - SiTCP (or Fakernet) protocol is used to push data to the PC (simple 10G NIC card is used).
- nestDAQ could be already used for ePIC?
  - As long as HBFrame is defined, no major obstacles to use nestDAQ in collider mode.
  - Any concerns? Maybe it would be good to collect such list of concerns so that SPADI-A can address one-by-one.
  - Try to install nestDAQ system at Jlab Hall-A BDX experiment (Marco, Mariangela) and check the compatibilities/robustness?
  - If we have simulation data (TFdata from subsystems), we can try to replay in nestDAQ and develop online processing using hardware accelerators for benchmarking.

Backup I

J-PARC E50

- **Workflows for online processing**

30 MHz beam rate
5% reaction rate
TF length = 512usec
(60k beams, 3k events in TF)

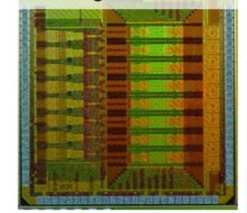


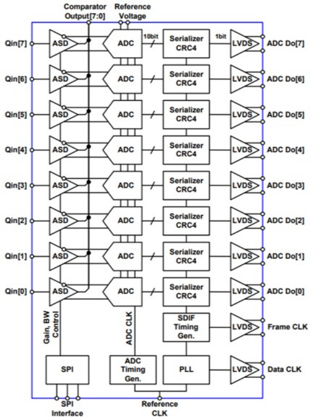
WG1&2 : FEE

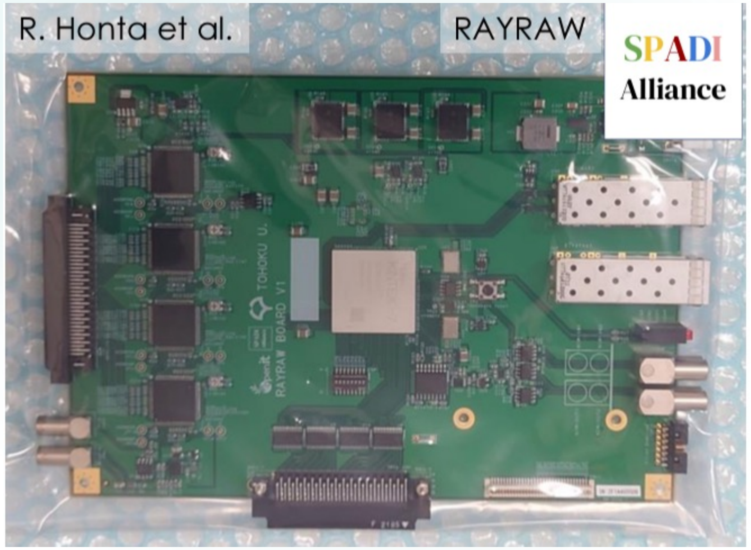
- Many types of FEEs for MPPC, gaseous detectors, Si readout, FADC, TDC, …

MPPC ASIC Board
YAENAMI ASIC (8ch)

65 nm Si CMOS 2 mm角
（Package: LQFP100）

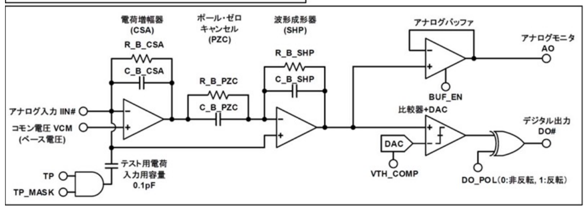


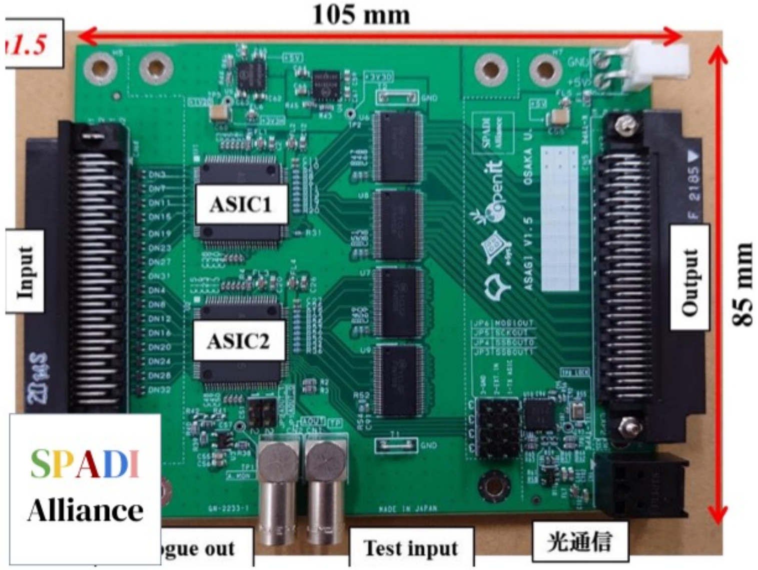




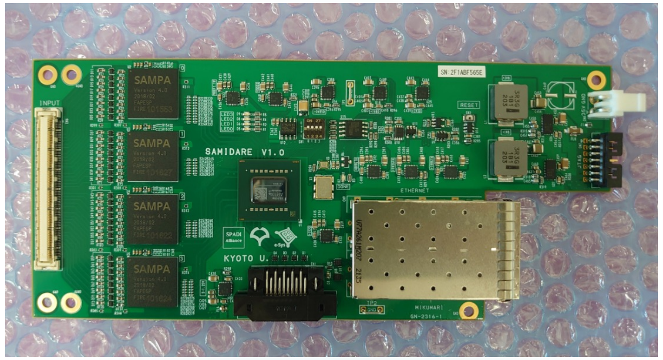
Gas chamber ASD
AGASA ASIC (16ch)

Electric circuit of 1 ASD channel





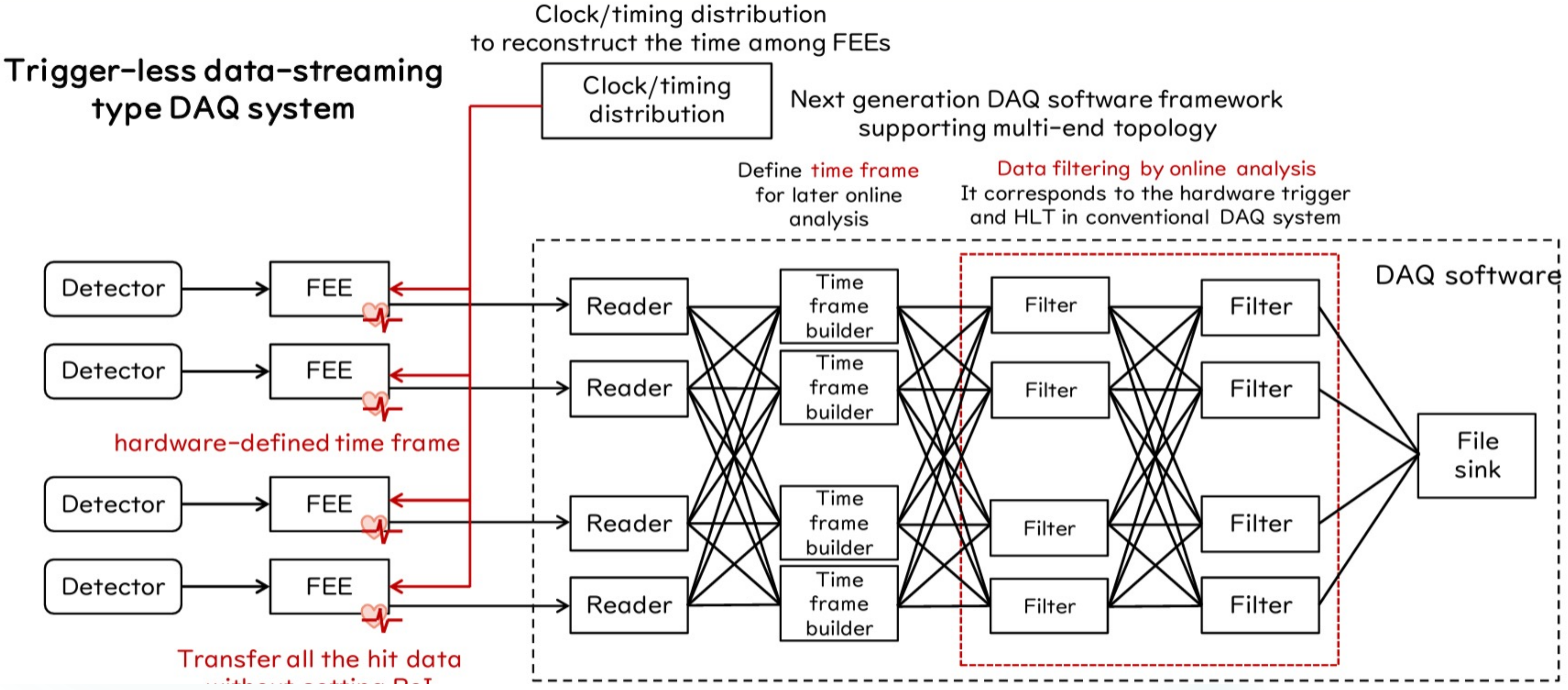
SAMPA chip board “SAMIDARE”



High resolution
FADC MIRA

TDC, Timing distribution
(AMANEQ)

Toward trigger-less data-streaming DAQ system



Clock/timing distribution to reconstruct the time among FEEs

Trigger-less data-streaming type DAQ system

Next generation DAQ software framework supporting multi-end topology

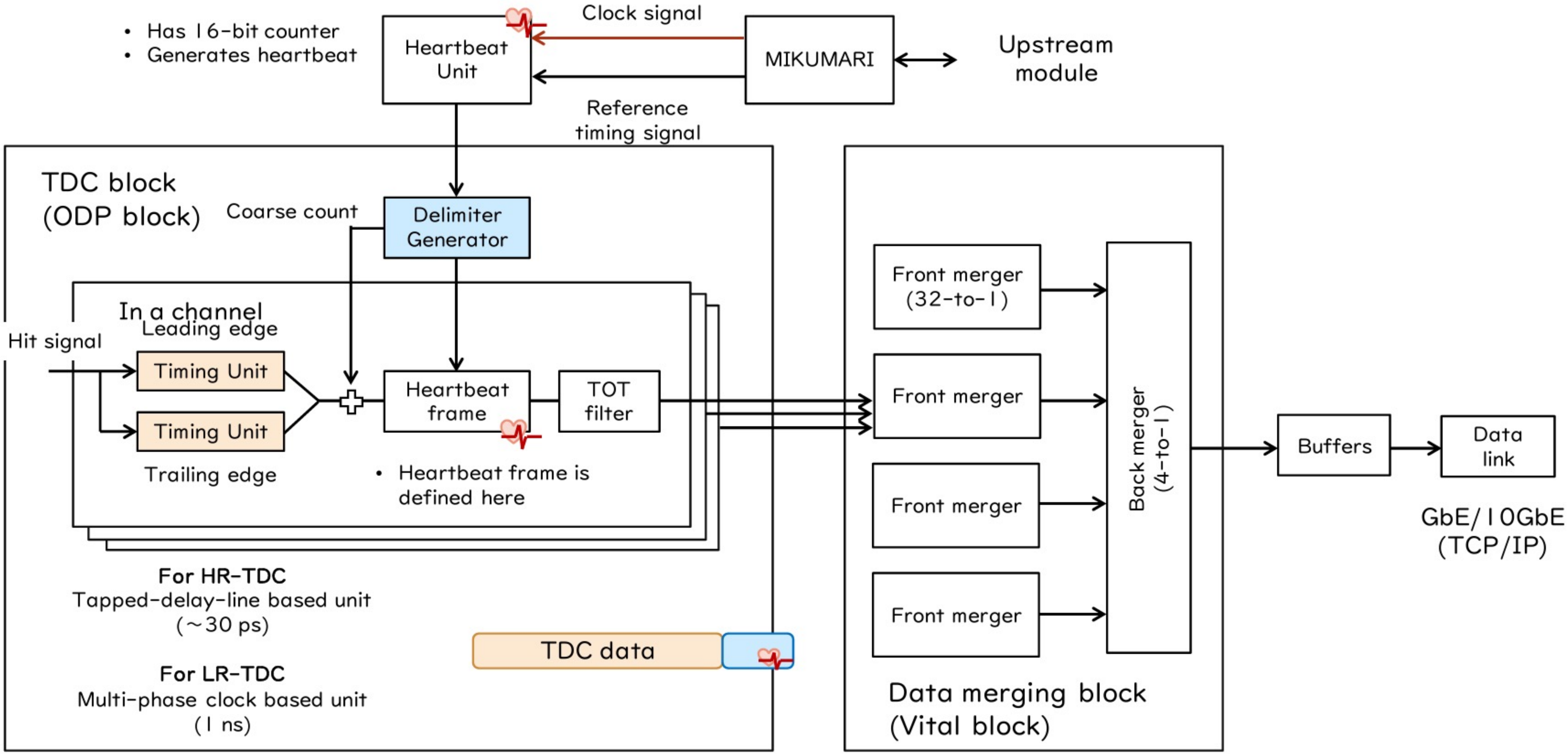
Define time frame for later online analysis

Data filtering by online analysis
It corresponds to the hardware trigger and HLT in conventional DAQ system

hardware-defined time frame

Transfer all the hit data without setting RoI

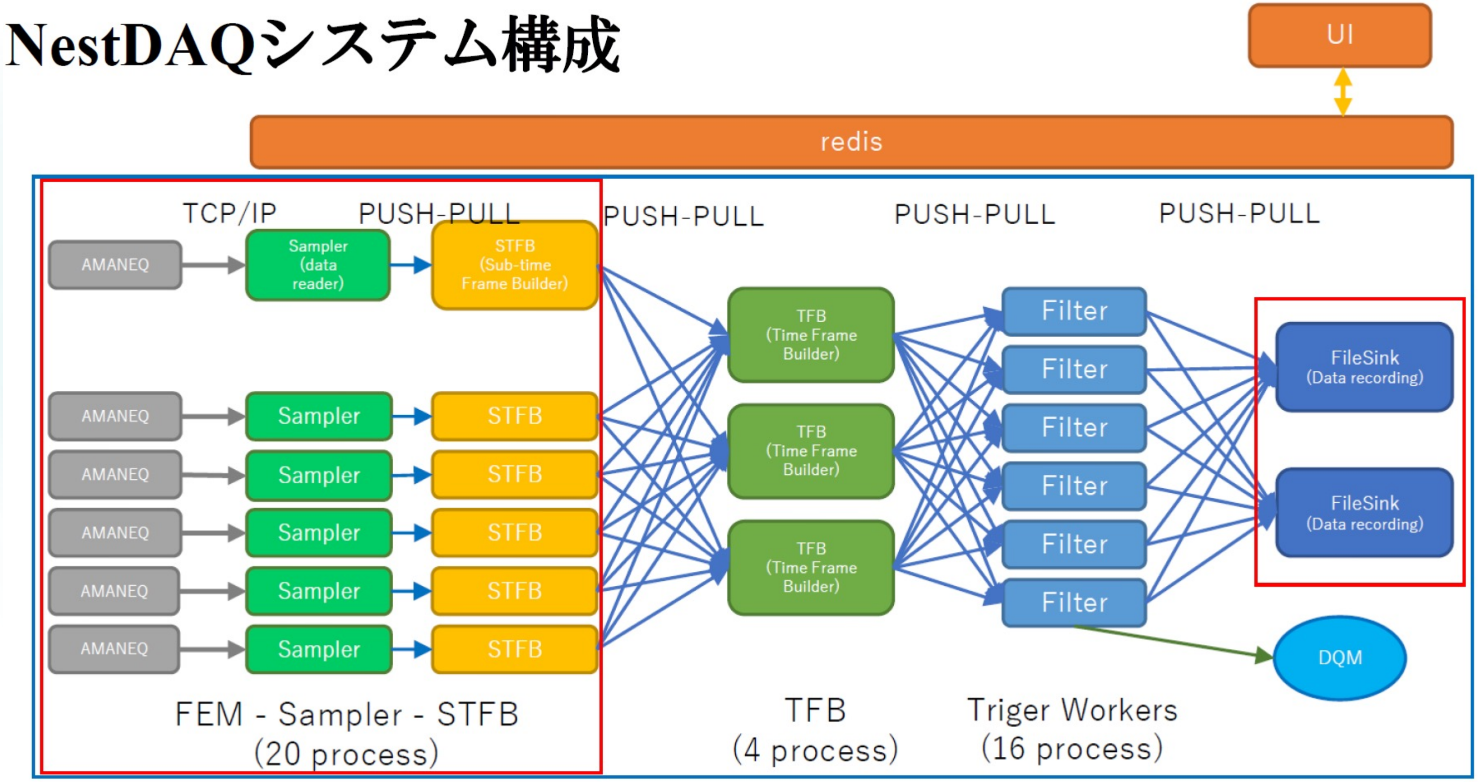
Simplified block diagram of Str-TDC

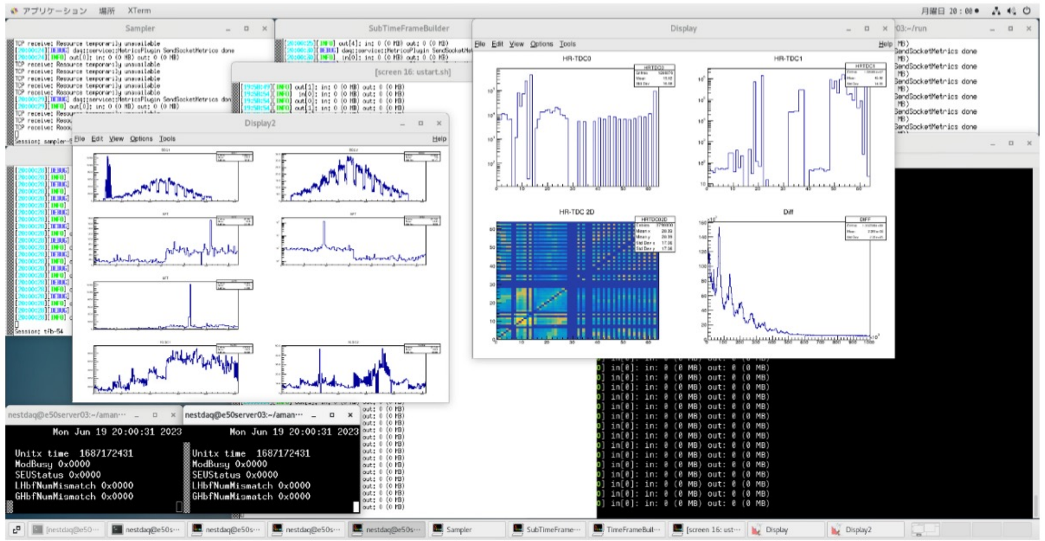


nestDAQ of test exp. at J-PARC

- Beamtime in last June (19.6-21.6)
  - K^- ~ 200k/spill (2sec), π^- ~ 800k/spill
  - Tests of streaming DAQ with “nestDAQ” and online coincidence filtering (based on CPU)

NestDAQシステム構成

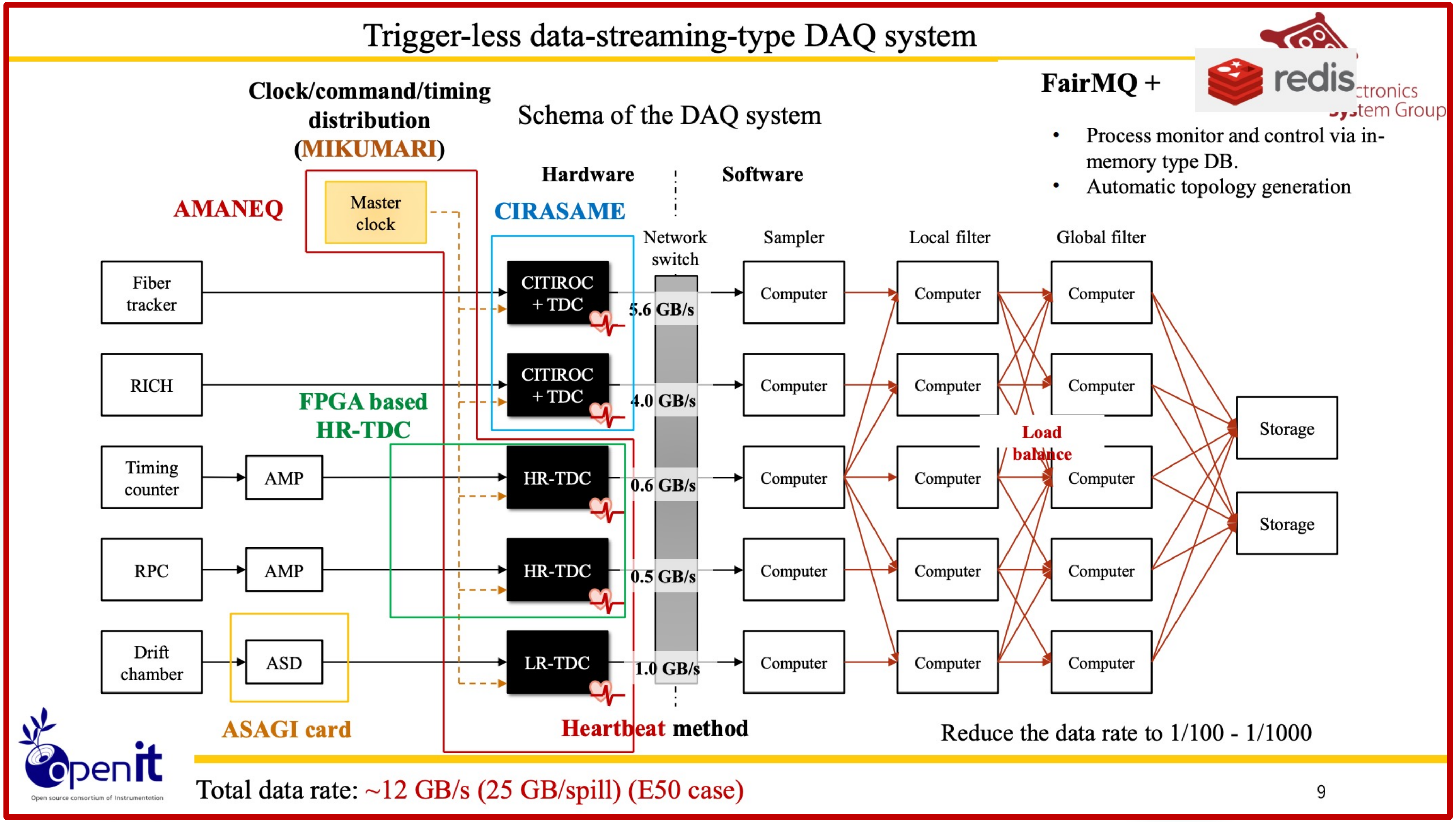




```
        Mon Jun 19 20:00:31 2023

 Unitx time  1687172431
 ModBusy 0x0000
 SEUStatus 0x0000
 LHbfNumMismatch 0x0000
 GHbfNumMismatch 0x0000
```

```
        Mon Jun 19 20:00:31 2023

 Unitx time  1687172431
 ModBusy 0x0000
 SEUStatus 0x0000
 LHbfNumMismatch 0x0000
 GHbfNumMismatch 0x0000
```

J-PARC E50



Trigger-less data-streaming-type DAQ system

Schema of the DAQ system

FairMQ + redis

- Process monitor and control via in-memory type DB.
- Automatic topology generation

Clock/command/timing distribution (MIKUMARI)

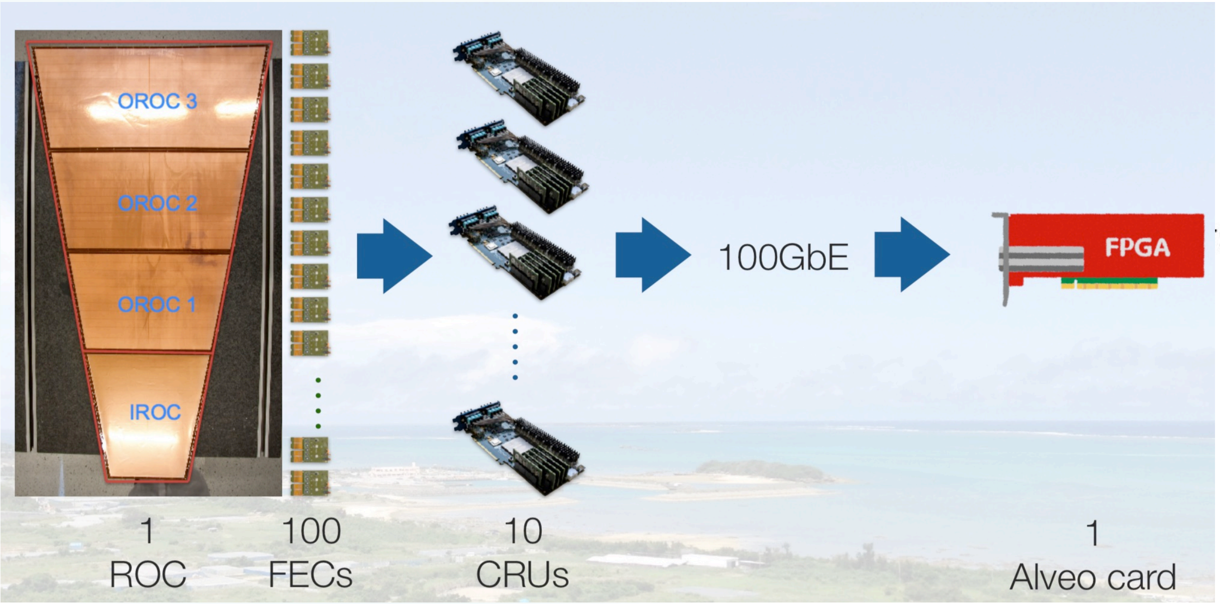
Reduce the data rate to 1/100 - 1/1000

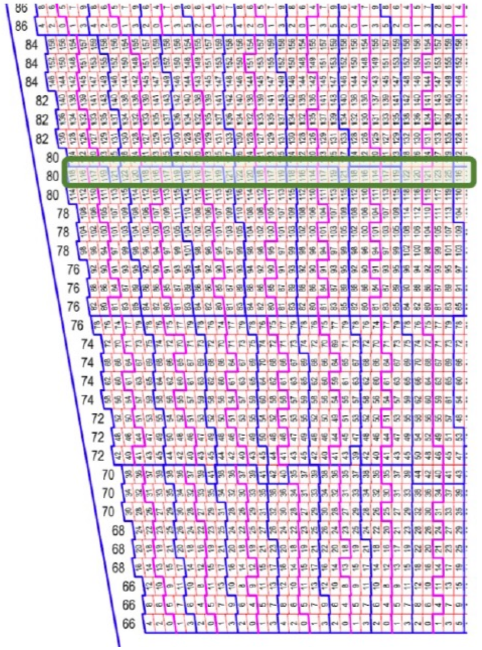
Total data rate: ~12 GB/s (25 GB/spill) (E50 case)

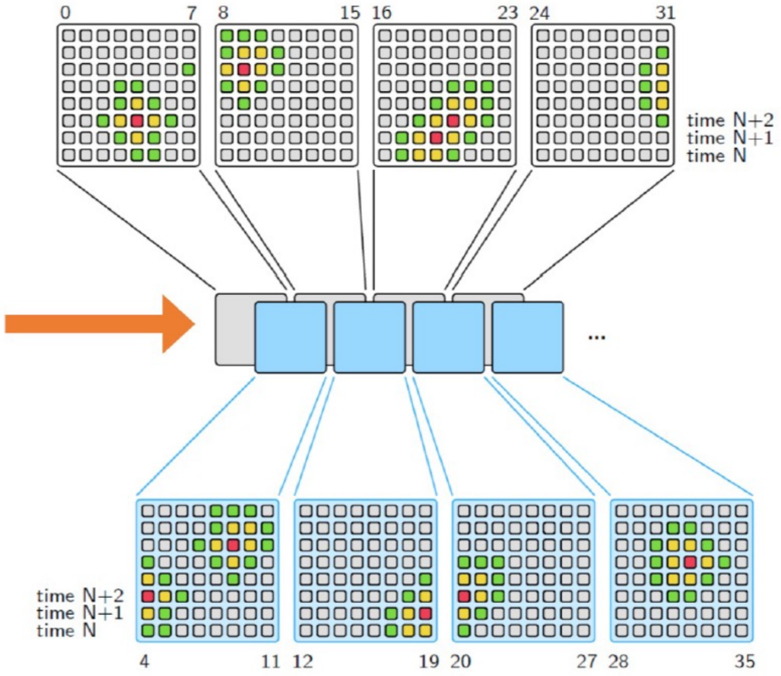
Backup II (from APS/JPS meeting)

TPC Clustering in FPGA (HLS)

- TPC clustering for one sector (currently running on GPU) can be run in FPGA?
  - Find local maxima in pad-timebin 2D space
  - Scan rectangular region in the pipelined way
- Alveo U55C (VU47P, 3 SLRs, 16GB HBM)



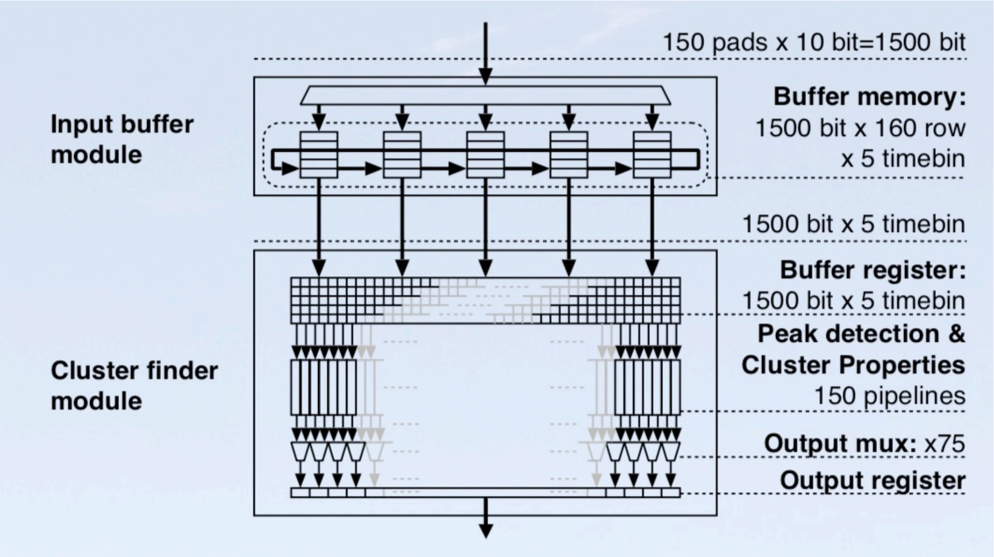


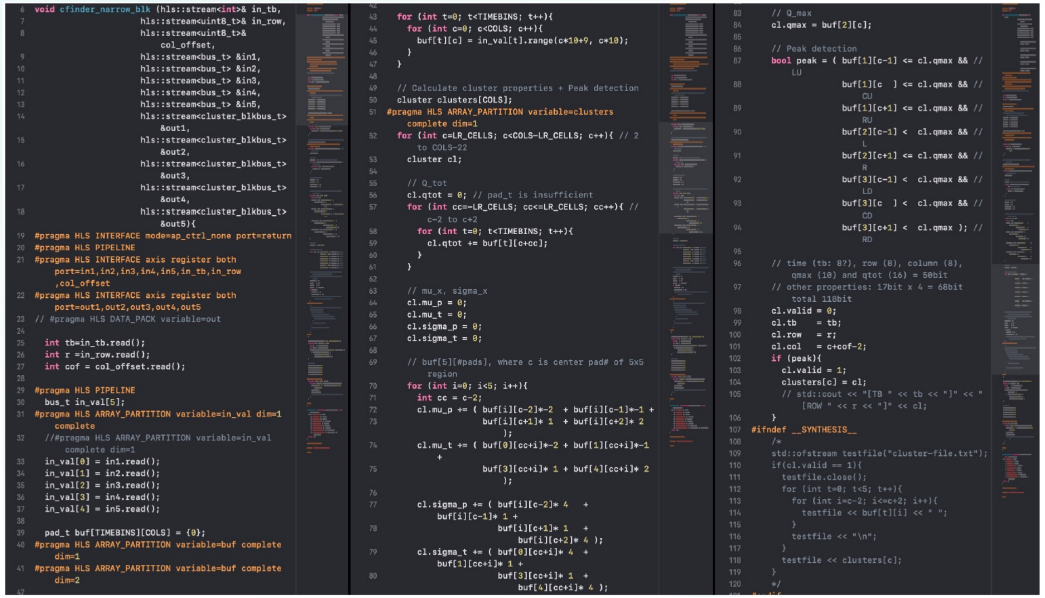


TPC Clustering in FPGA (HLS)

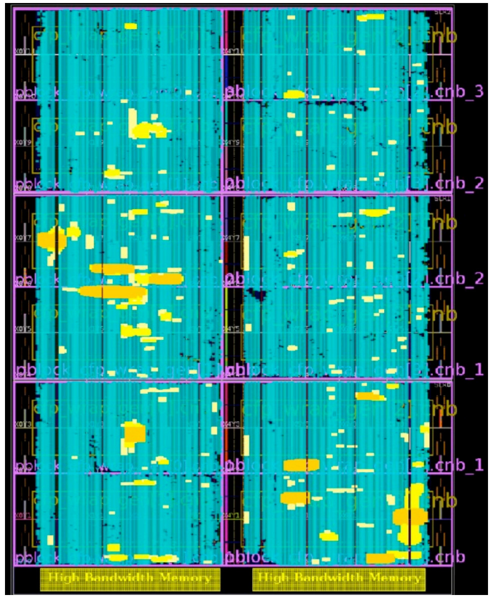
- Data is sent from data source via UDP (100Gbps)
- FPGA logic consists of
  - UDP packet decoder
  - Buffering (150 pads x 10 bits x 160 row x 5 timebins)
  - Clustering (150 pipelines) via pad x timebin [HLS]
  - Output to PCIe

200ns/timebin


150 pads x 10 bit=1500 bit
Input buffer module
Buffer memory: 1500 bit x 160 row x 5 timebin
1500 bit x 5 timebin
Cluster finder module
Buffer register: 1500 bit x 5 timebin
Peak detection & Cluster Properties 150 pipelines
Output mux: x75
Output register



```
void cfinder_narrow_blk (hls::stream<int>& in_tb,
                         hls::stream<uint8_t>& in_row,
                         hls::stream<uint8_t>& col_offset,
                         hls::stream<bus_t> &in1,
                         hls::stream<bus_t> &in2,
                         hls::stream<bus_t> &in3,
                         hls::stream<bus_t> &in4,
                         hls::stream<bus_t> &in5,
                         hls::stream<cluster_blkbus_t> &out1,
                         hls::stream<cluster_blkbus_t> &out2,
                         hls::stream<cluster_blkbus_t> &out3,
                         hls::stream<cluster_blkbus_t> &out4,
                         hls::stream<cluster_blkbus_t> &out5){
#pragma HLS INTERFACE mode=ap_ctrl_none port=return
#pragma HLS PIPELINE
#pragma HLS INTERFACE axis register both port=in1,in2,in3,in4,in5,in_tb,in_row,col_offset
#pragma HLS INTERFACE axis register both port=out1,out2,out3,out4,out5
// #pragma HLS DATA_PACK variable=out

  int tb=in_tb.read();
  int r =in_row.read();
  int cof = col_offset.read();

#pragma HLS PIPELINE
  bus_t in_val[5];
#pragma HLS ARRAY_PARTITION variable=in_val dim=1 complete
  //#pragma HLS ARRAY_PARTITION variable=in_val complete dim=1
  in_val[0] = in1.read();
  in_val[1] = in2.read();
  in_val[2] = in3.read();
  in_val[3] = in4.read();
  in_val[4] = in5.read();

  pad_t buf[TIMEBINS][COLS] = {0};
#pragma HLS ARRAY_PARTITION variable=buf complete dim=1
#pragma HLS ARRAY_PARTITION variable=buf complete dim=2

  for (int t=0; t<TIMEBINS; t++){
    for (int c=0; c<COLS; c++){
      buf[t][c] = in_val[t].range(c*10+9, c*10);
    }
  }

  // Calculate cluster properties + Peak detection
  cluster clusters[COLS];
#pragma HLS ARRAY_PARTITION variable=clusters complete dim=1
  for (int c=LR_CELLS; c<COLS-LR_CELLS; c++){ // 2 to COLS-22
    cluster cl;

    // Q_tot
    cl.qtot = 0; // pad_t is insufficient
    for (int cc=-LR_CELLS; cc<=LR_CELLS; cc++){ // c-2 to c+2
      for (int t=0; t<TIMEBINS; t++){
        cl.qtot += buf[t][c+cc];
      }
    }

    // mu_x, sigma_x
    cl.mu_p = 0;
    cl.mu_t = 0;
    cl.sigma_p = 0;
    cl.sigma_t = 0;

    // buf[5][#pads], where c is center pad# of 5x5 region
    for (int i=0; i<5; i++){
      int cc = c-2;
      cl.mu_p += ( buf[i][c-2]*-2  + buf[i][c-1]*-1 +
                   buf[i][c+1]* 1  + buf[i][c+2]* 2 );
      cl.mu_t += ( buf[0][cc+i]*-2 + buf[1][cc+i]*-1 +
                   buf[3][cc+i]* 1 + buf[4][cc+i]* 2 );

      cl.sigma_p += ( buf[i][c-2]* 4   + buf[i][c-1]* 1 +
                      buf[i][c+1]* 1   + buf[i][c+2]* 4 );
      cl.sigma_t += ( buf[0][cc+i]* 4  + buf[1][cc+i]* 1 +
                      buf[3][cc+i]* 1  + buf[4][cc+i]* 4 );

    // Q_max
    cl.qmax = buf[2][c];

    // Peak detection
    bool peak = ( buf[1][c-1] <= cl.qmax && // LU
                  buf[1][c  ] <= cl.qmax && // CU
                  buf[1][c+1] <= cl.qmax && // RU
                  buf[2][c-1] <  cl.qmax && // L
                  buf[2][c+1] <= cl.qmax && // R
                  buf[3][c-1] <  cl.qmax && // LD
                  buf[3][c  ] <  cl.qmax && // CD
                  buf[3][c+1] <  cl.qmax ); // RD

    // time (tb: 8?), row (8), column (8), qmax (10) and qtot (16) = 50bit
    // other properties: 17bit x 4 = 68bit total 118bit
    cl.valid = 0;
    cl.tb    = tb;
    cl.row   = r;
    cl.col   = c+cof-2;
    if (peak){
      cl.valid = 1;
      clusters[c] = cl;
      // std::cout << "[TB " << tb << "]" << " [ROW " << r << "]" << cl;
    }
#ifndef __SYNTHESIS__
    /*
    std::ofstream testfile("cluster-file.txt");
    if(cl.valid == 1){
      testfile.close();
      for (int t=0; t<5; t++){
        for (int i=c-2; i<=c+2; i++){
          testfile << buf[t][i] << " ";
        }
        testfile << "\n";
      }
      testfile << clusters[c];
    }
    */
```



- CRU 0-2: 48 rows
- CRU 3-5: 49 rows
- CRU 6-9: 55 rows

We achieved to run at 290 MHz

> 55 rows x 5 MSPS = 275 MHz

Clustering for one sector can run without latencies

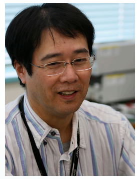
	Used	Available	%
LUT	714,307	1,303,680	54.8
FF	814,523	2,607,360	31.2
BRAM	846.5	2,016	42.0

E50 test bench

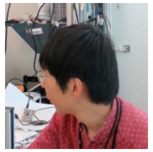
SPADI Alliance

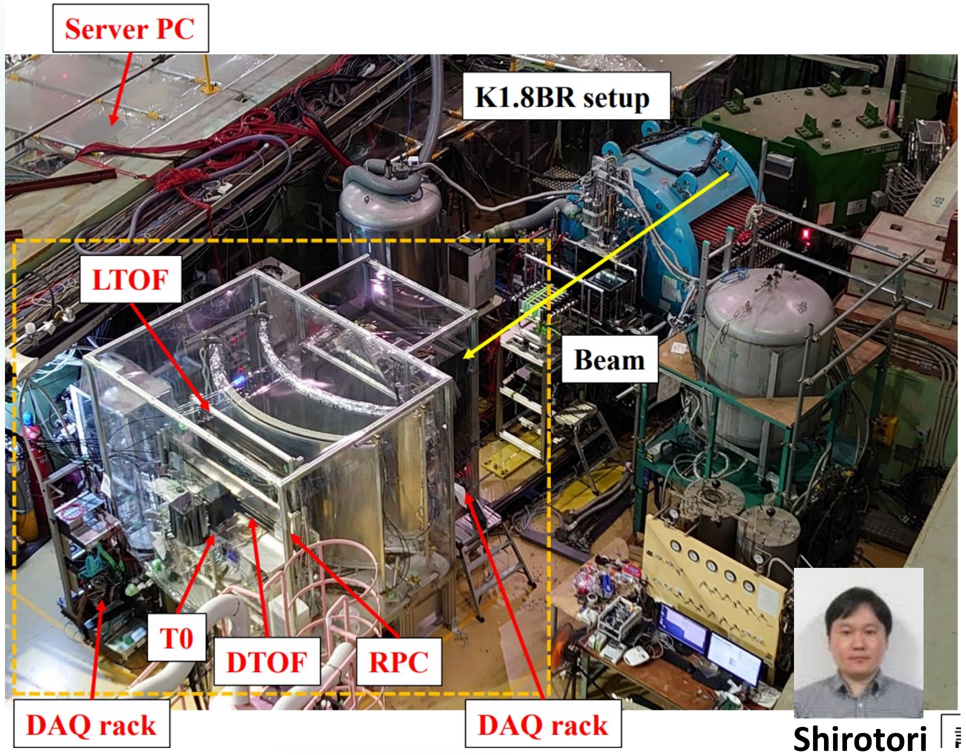
- Beamtime in last June (19.6-21.6)
  - K^- ~ 200k/spill (2sec), π^- ~ 800k/spill
  - Tests of streaming DAQ with “nestDAQ” and online coincidence filtering (based on CPU)

Igarashi

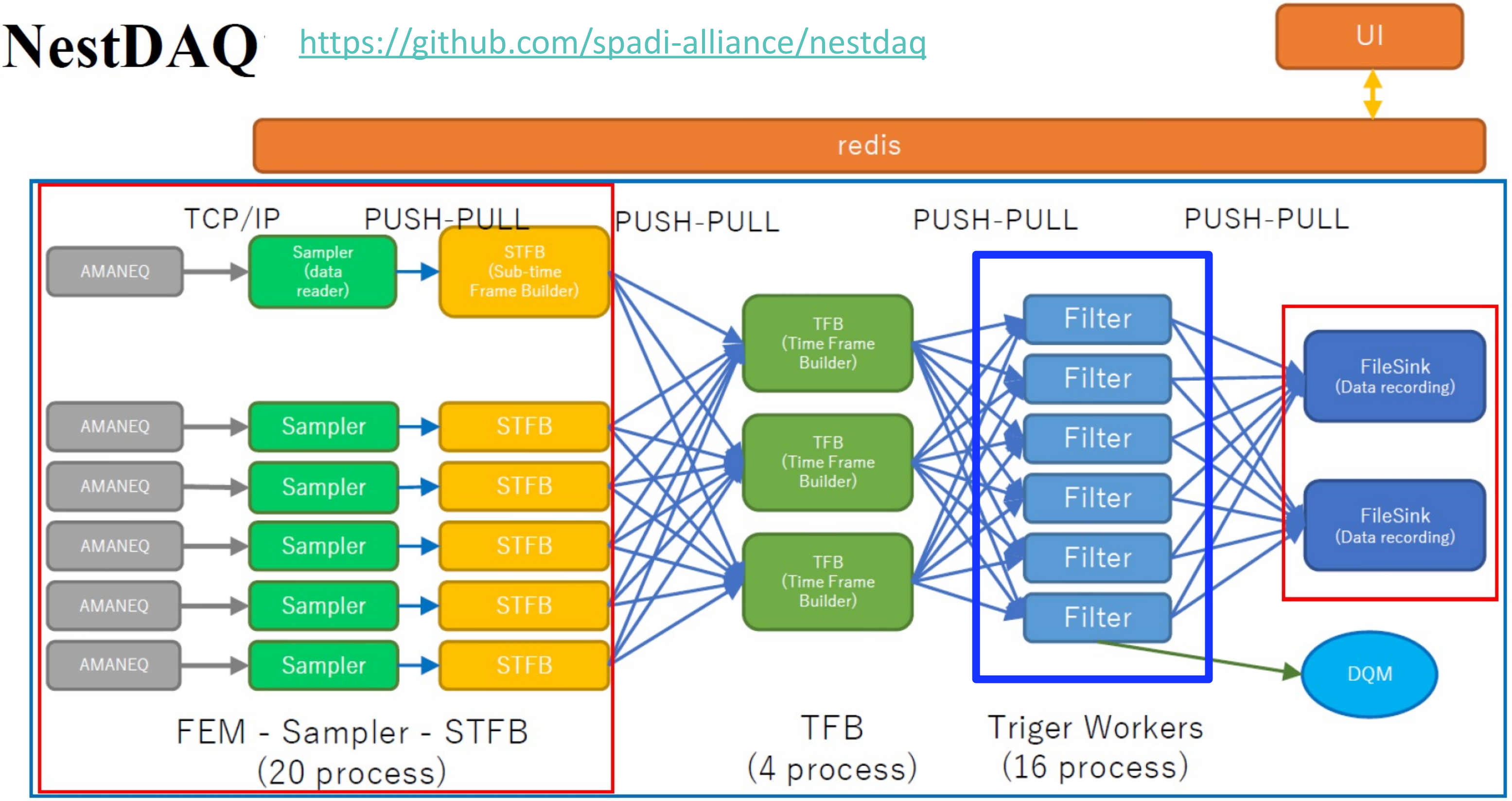


Takahashi





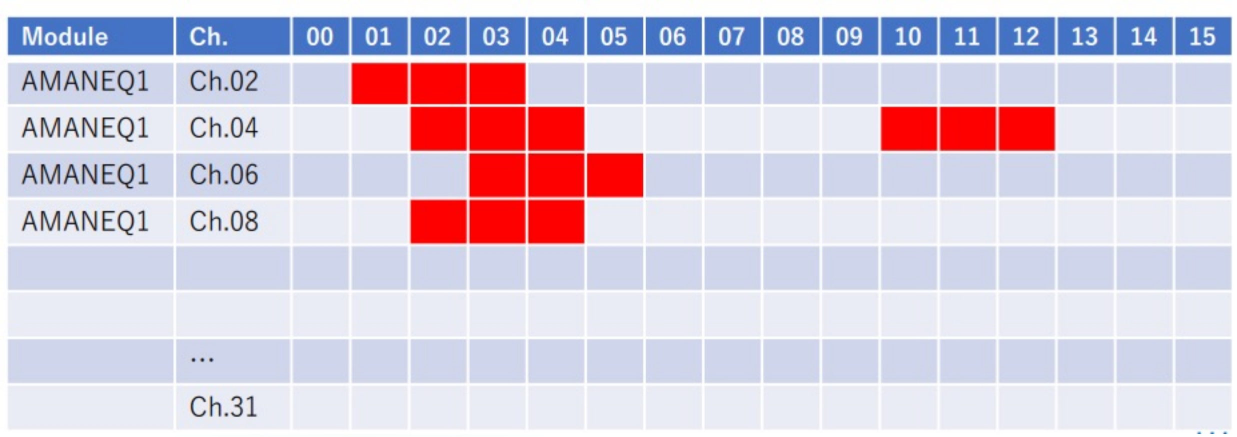
NestDAQ https://github.com/spadi-alliance/nestdaq



Online coincidence filter

General Purpose Logic Filter

- Create LUT https://github.com/spadi-alliance/nestdaq-user-impl
- Make hit markers (every 4nsec – from TDC) in the array for Heart-beat frame (524usec)
- Scan the array and see if all entries are fired (LUT = true)
- **Running on CPU (will be ported in GPU)**

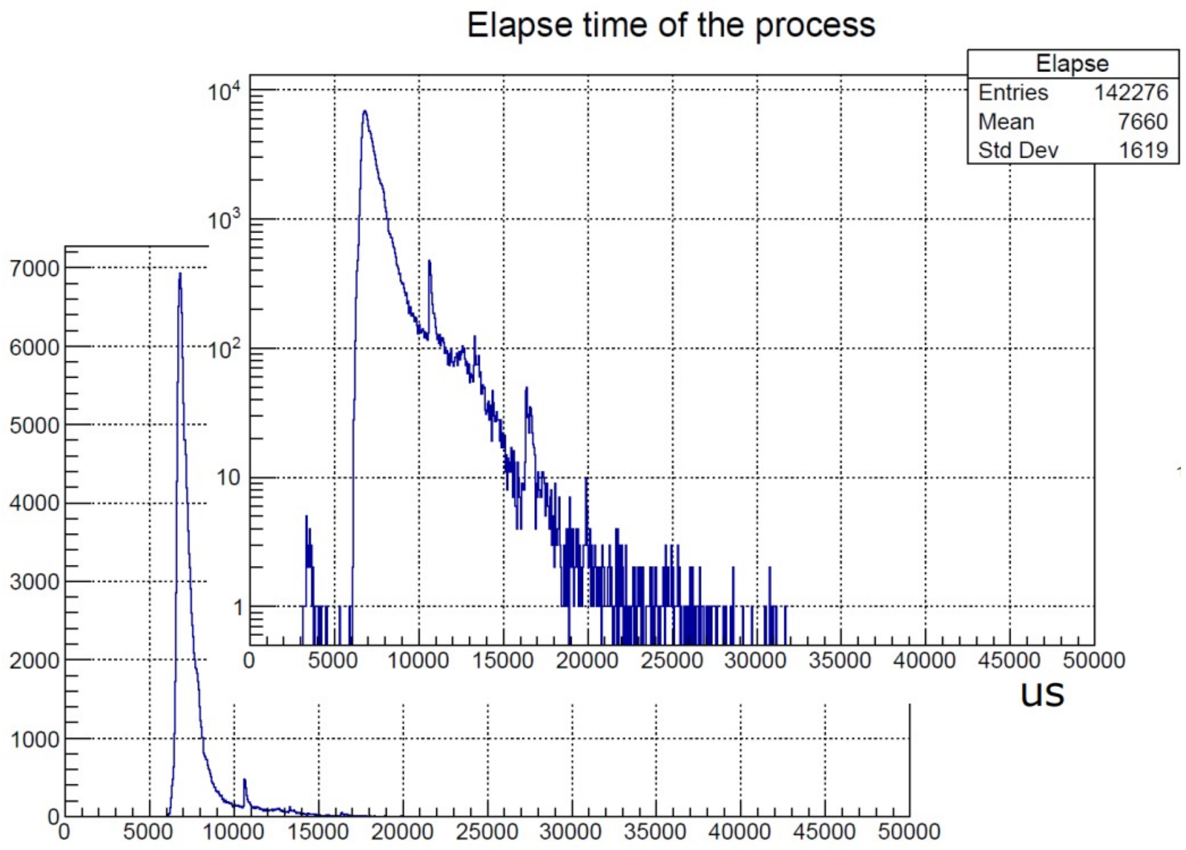


Module	Ch.	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
AMANEQ1	Ch.02		🟥	🟥	🟥												
AMANEQ1	Ch.04			🟥	🟥	🟥						🟥	🟥	🟥			
AMANEQ1	Ch.06				🟥	🟥	🟥										
AMANEQ1	Ch.08			🟥	🟥	🟥											
	…																
	Ch.31																

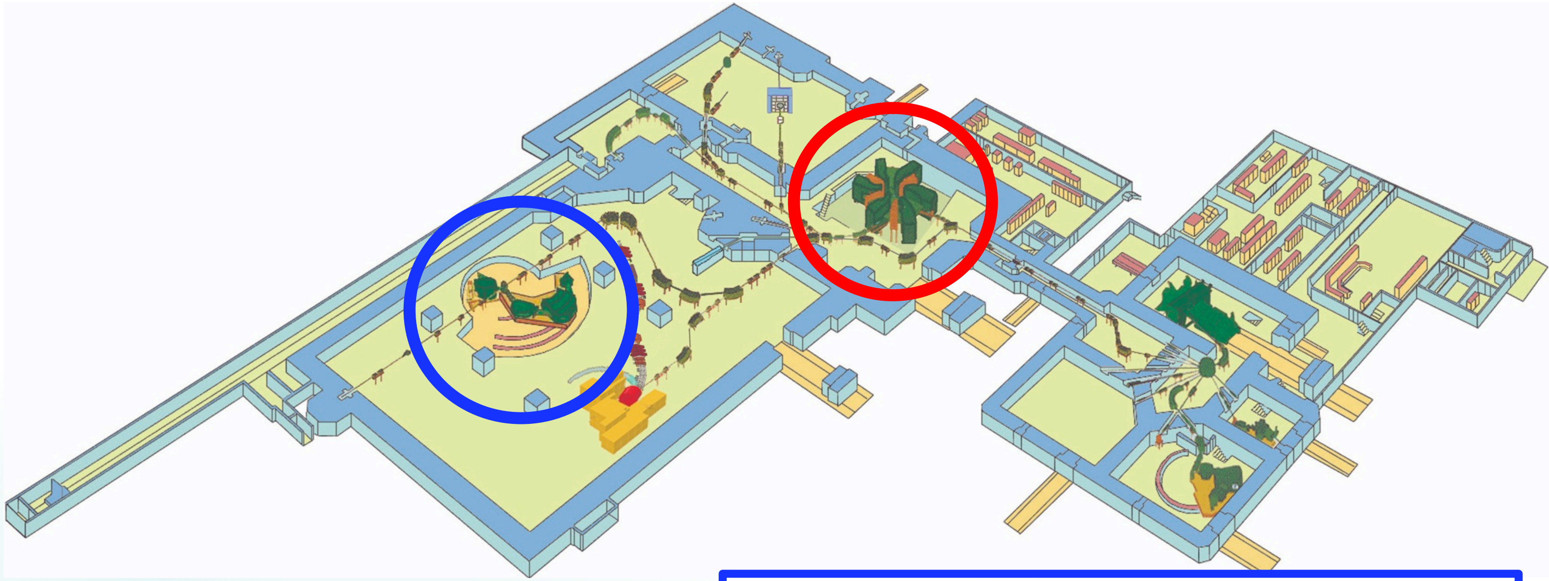
…

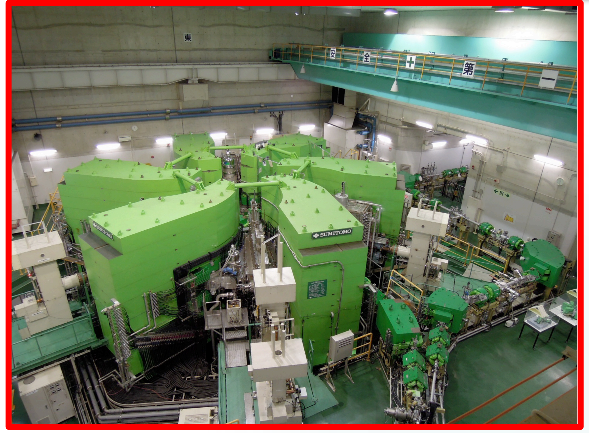
Processing time for 1 HBF (524us)

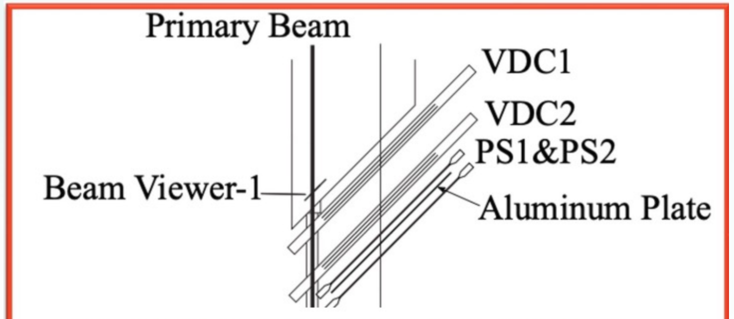
- Average 7.6msec -> 15 processors



RCNP and Streaming DAQ



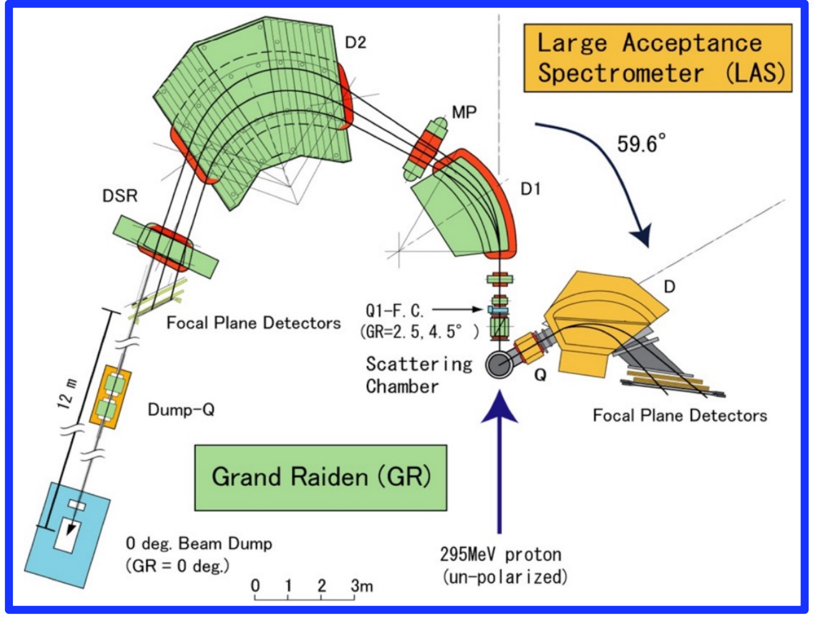


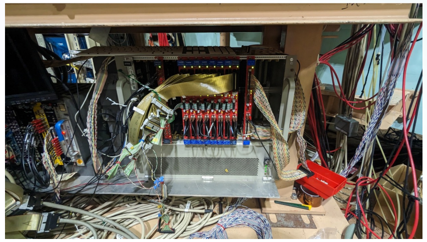


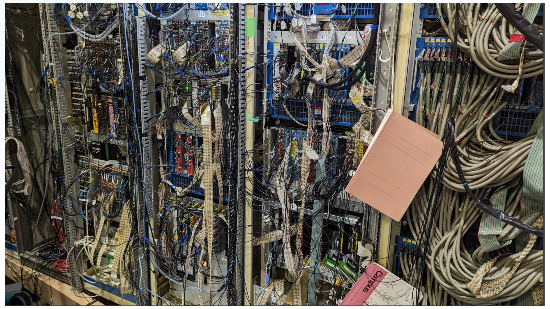
VDC (X1:192, U1:208, X2:192, U2:208)

PS (PS1: 2, PS2: 2)

RF timing, Current integrator





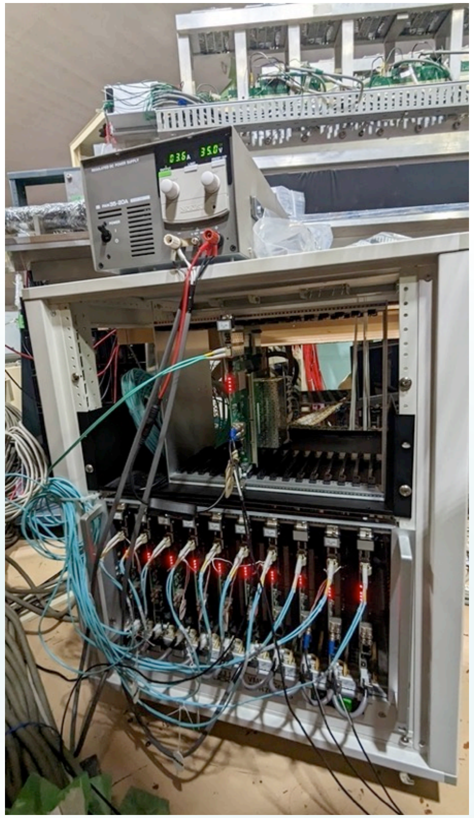


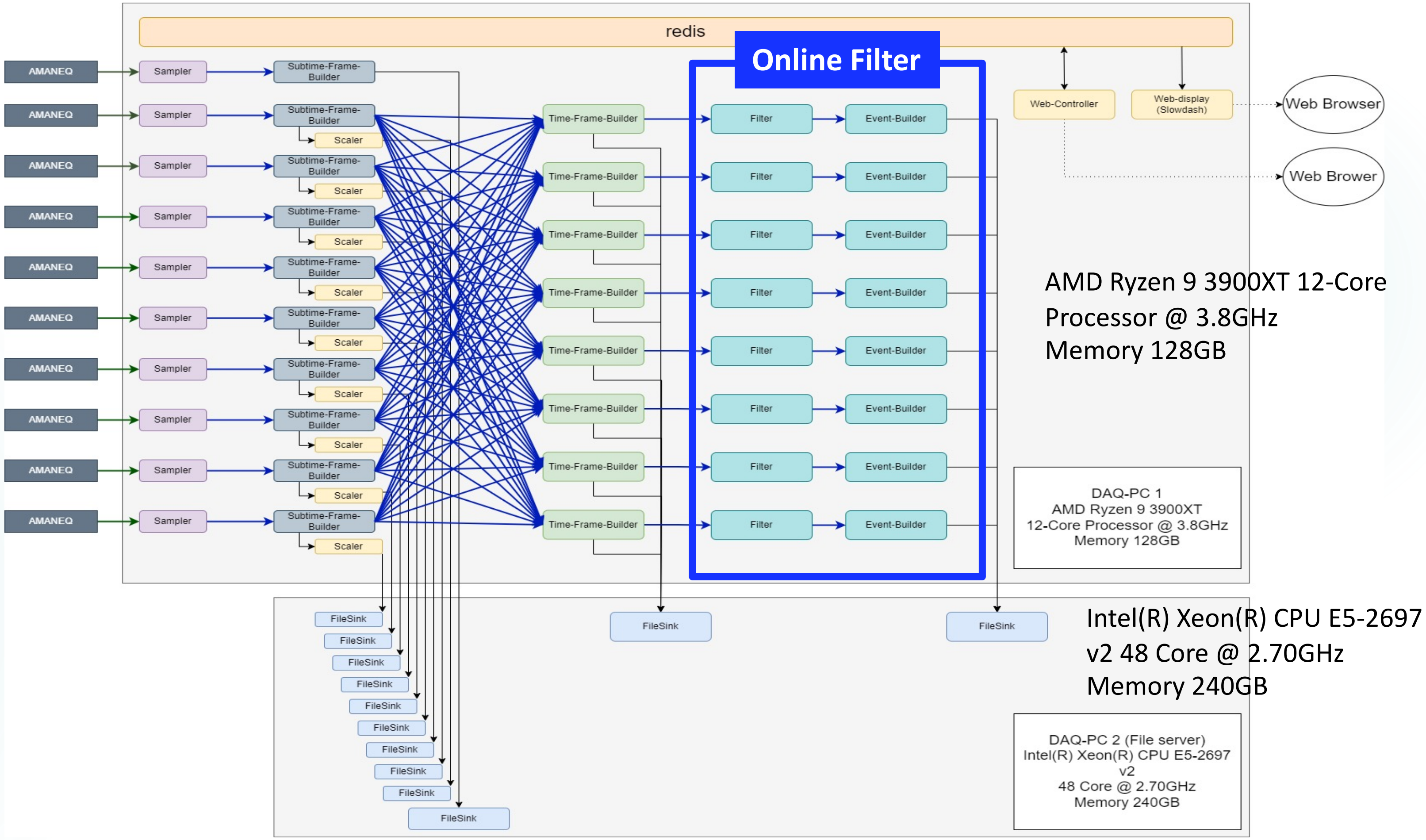
Using multi-event buffering in VME / CAMAC modules, readout is limited up to 50kHz but DAQ rate is still limited by the DAQ busy.

We aim at reading up to O(100kHz) - 1MHz using streaming DAQ.

Physics runs using online filters

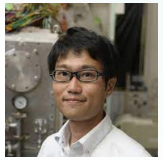
- Physics runs at GR beamline using nest DAQ





AMD Ryzen 9 3900XT 12-Core
Processor @ 3.8GHz
Memory 128GB

Intel(R) Xeon(R) CPU E5-2697
v2 48 Core @ 2.70GHz
Memory 240GB

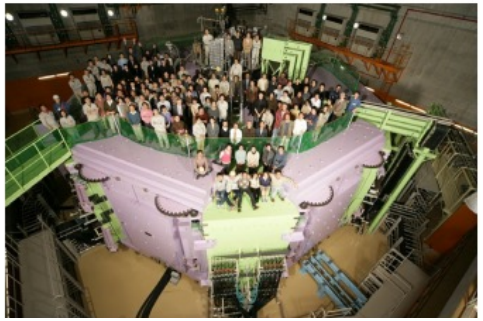


Ota

Kobayashi

Throughput 200Mbps (x40 improvement from past DAQ system). 100% efficient for 100-200 kcps
Next step is to implement online tracking, multi-hit identification (ex (d, 2p)), and PID (ex, (α, ^{6}He))

RIBF and Streaming DAQ

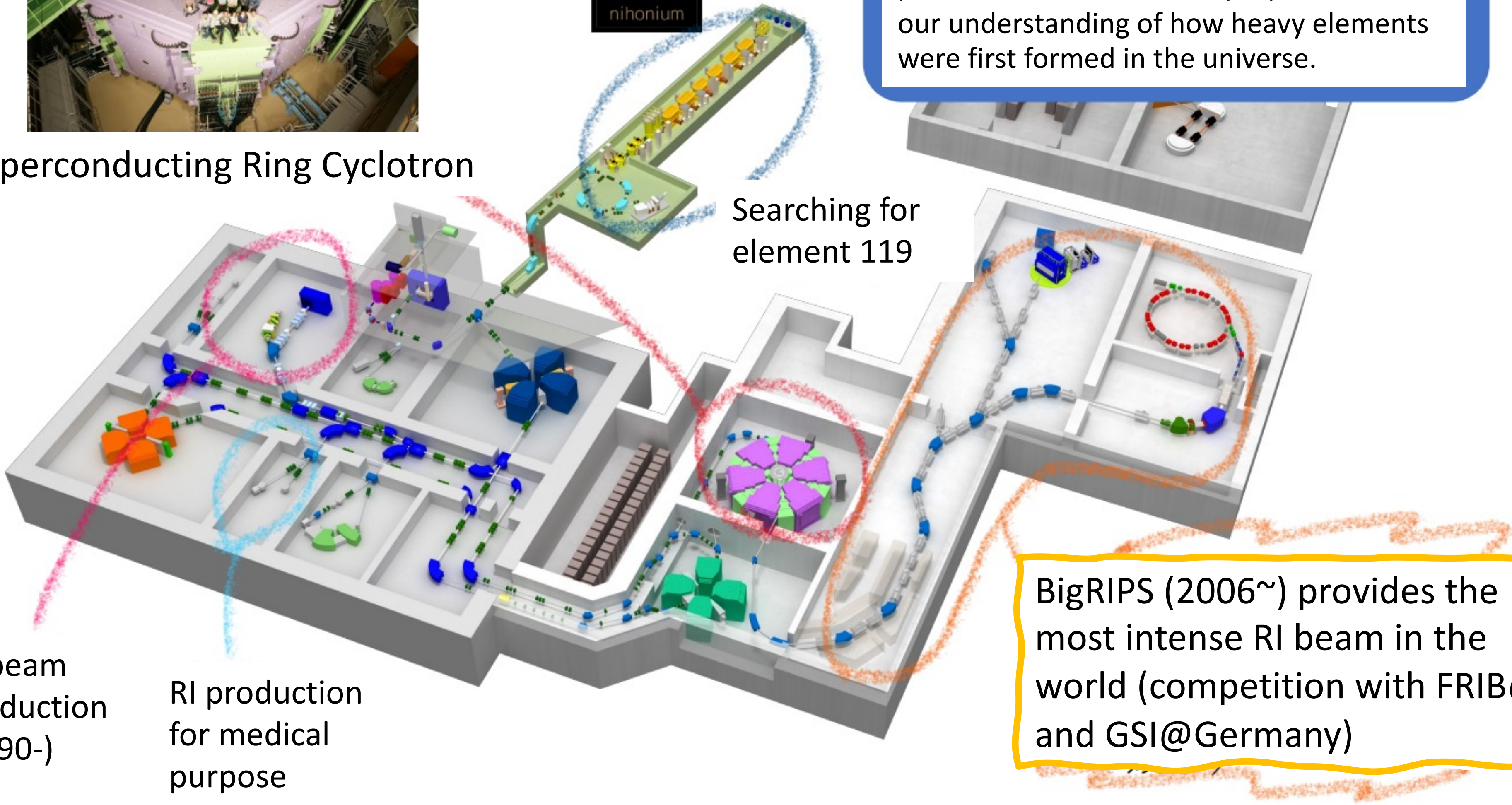


Superconducting Ring Cyclotron

Nihonium

113
Nh
nihonium

RIBF is a next-generation heavy-ion research facility to establish the ultimate picture of nuclei and thus propel forward our understanding of how heavy elements were first formed in the universe.



Searching for element 119

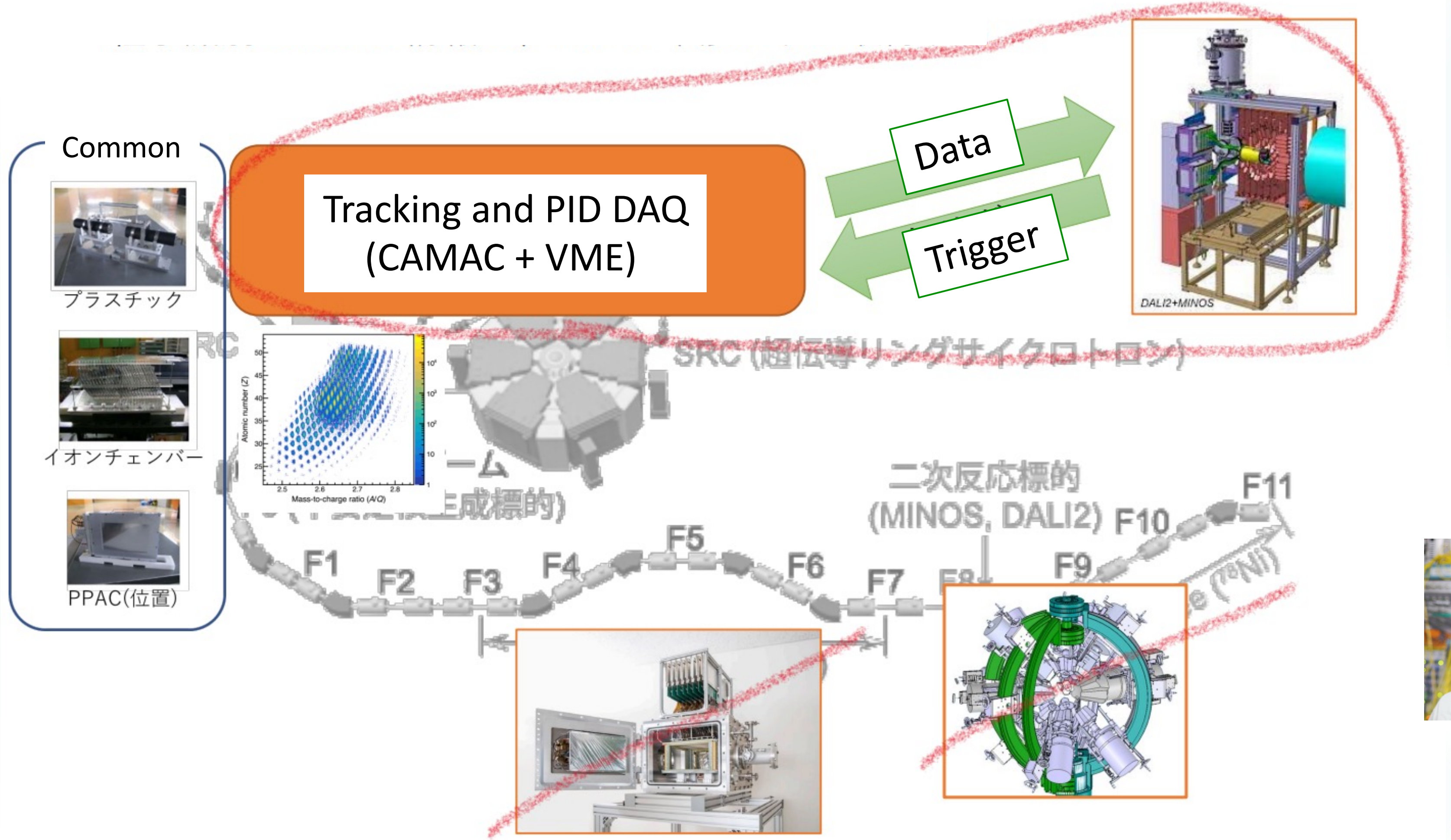
RI beam production (1990-)

RI production for medical purpose

BigRIPS (2006~) provides the most intense RI beam in the world (competition with FRIB@US and GSI@Germany)

Current RIBF running strategies

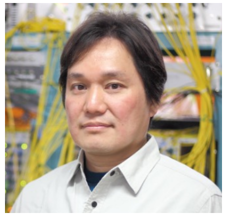
- **Common beamline detectors (Plastic, ion chamber, PPAC) are occupied by only one experiment**
- This should be shared with other experiments without any interferences → Streaming DAQ


Common

Tracking and PID DAQ
(CAMAC + VME)

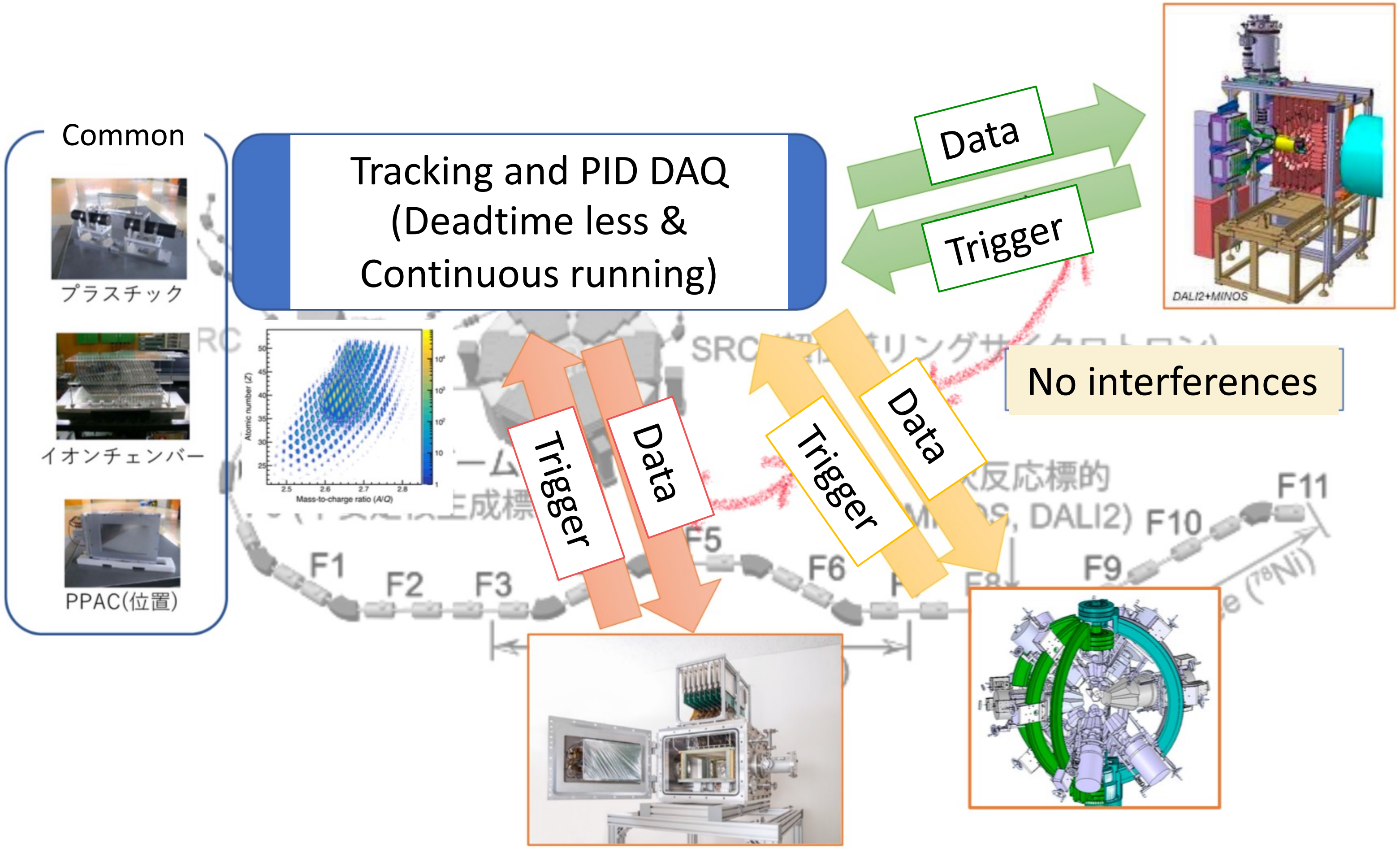
Data

Trigger


Baba

Current RIBF running strategies

- Common beamline detectors (Plastic, ion chamber, PPAC) are occupied by only one experiment
- **This should be shared with other experiments without any interferences → Streaming DAQ**



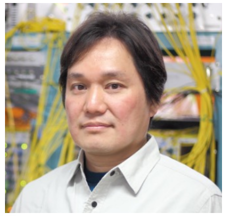
Common

Tracking and PID DAQ
(Deadtime less &
Continuous running)

Data

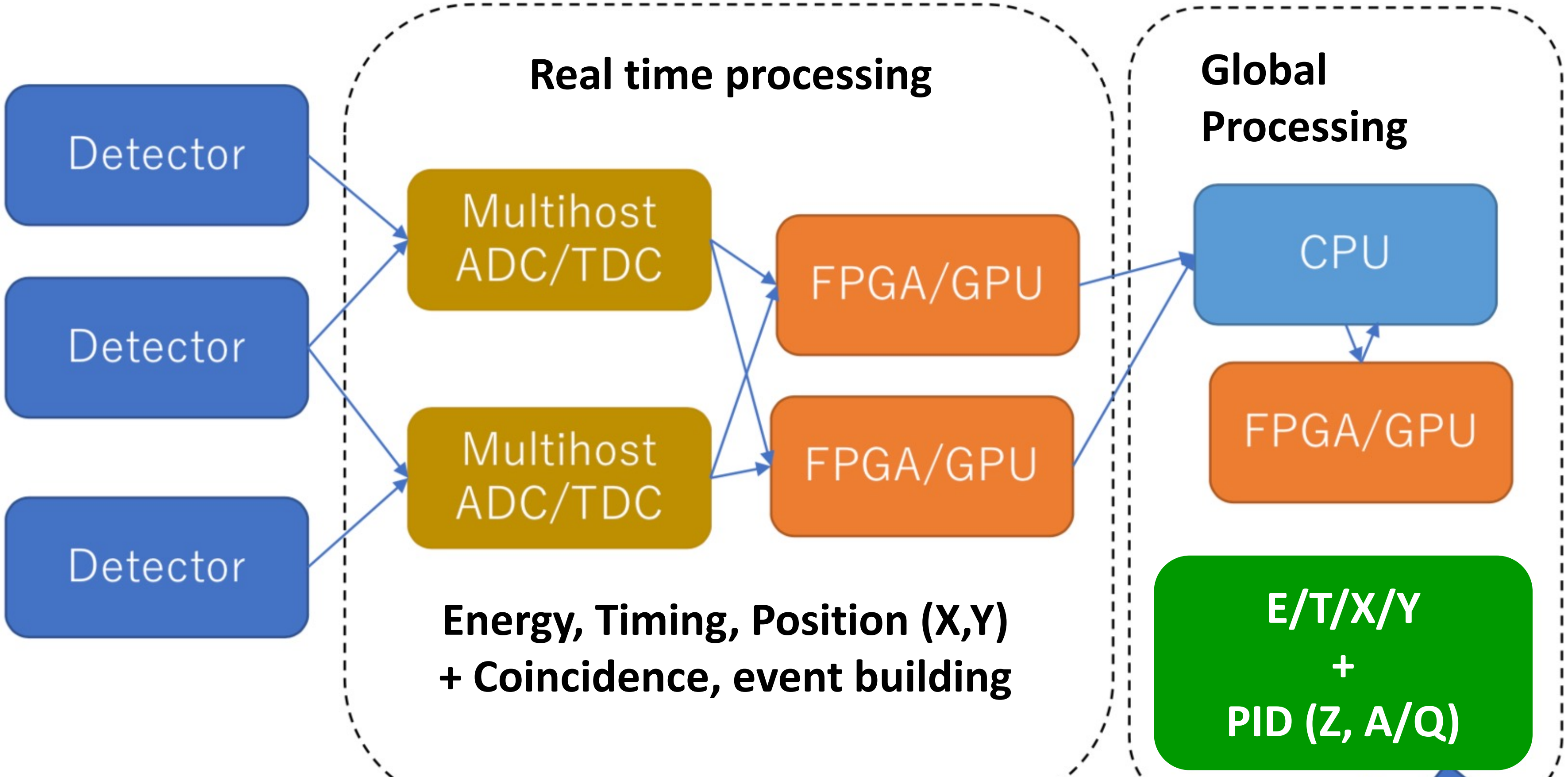
Trigger

No interferences



Baba

Streaming readout in RIBF



Real time processing

Global Processing

Detector

Detector

Detector

Multihost ADC/TDC

Multihost ADC/TDC

FPGA/GPU

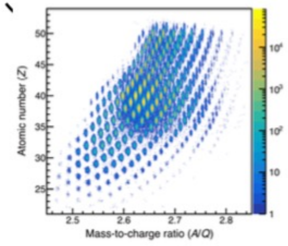
FPGA/GPU

CPU

FPGA/GPU

E/T/X/Y
+
PID (Z, A/Q)

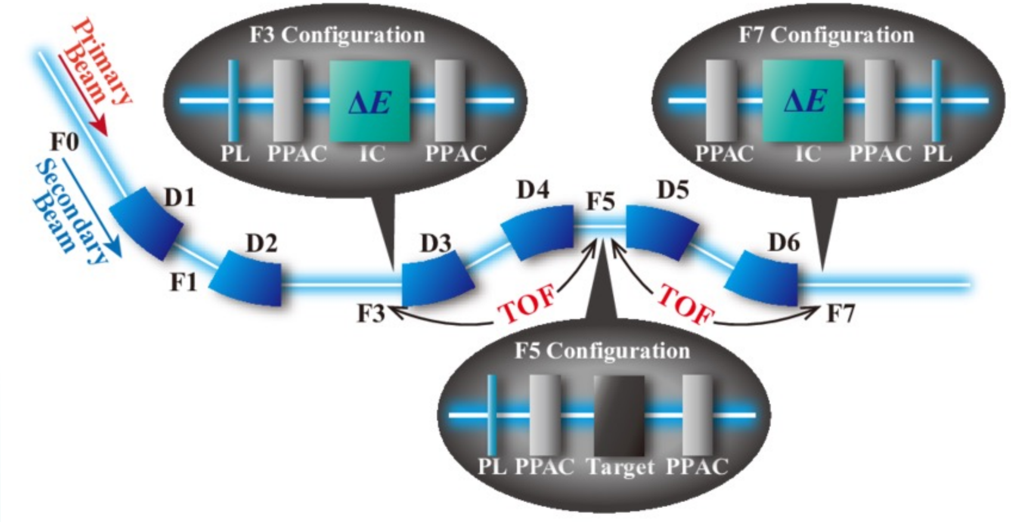
Energy, Timing, Position (X,Y)
+ Coincidence, event building



Estimated data rate ~ 10 MB/s
100 MB/s as maximum?

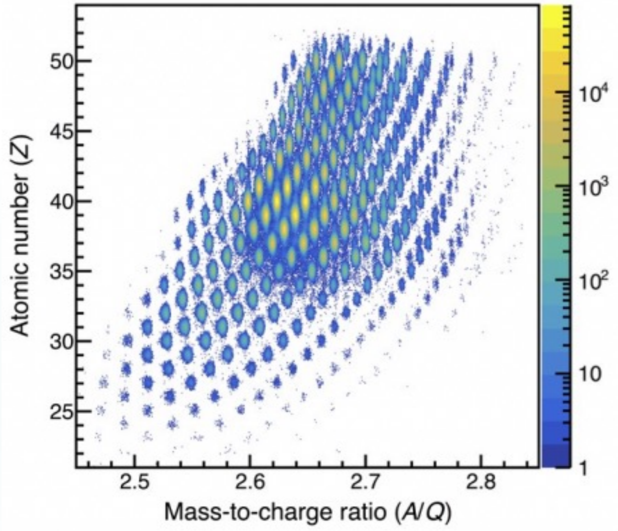
FPGA Processing of BLD data

Ichinohe (C07.4)



BigRIPS

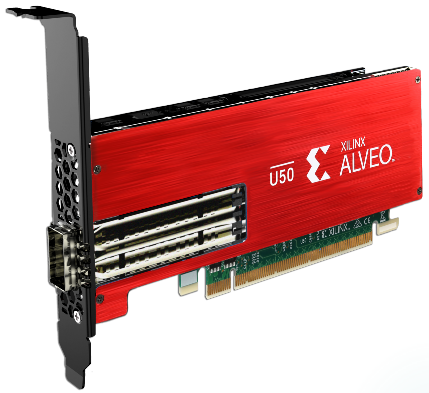
- Plastic (F3, F7) TOF -> β
- PPAC (F3, F5, F7) Tracking -> Bρ
- Ion chamber (F7) dE + β -> Z
- Bρ + β -> A/Q

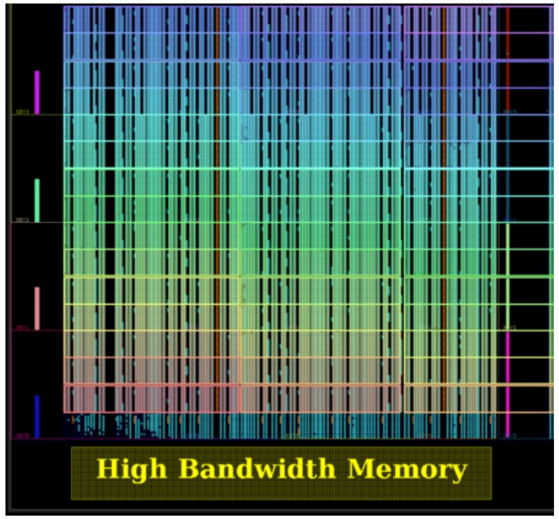


Online process consists of

- PPAC (F3, F5, F7)
  - 4 layers of (x,y) : tracking (angle, position) through chi2 fits
  - Combine F3-F7 to get Bρ
- Plastic (F3, F7)
  - Average timing from 2 PMTs
  - β from F3 ad F7
- Ion chamber (F7)
  - 6 layers / IC : take geometric mean of raw – pedestal
  - Extraction of Z from Bethe-Bloch
- A/Q vs. Z

Alveo U50(8GB HBM)

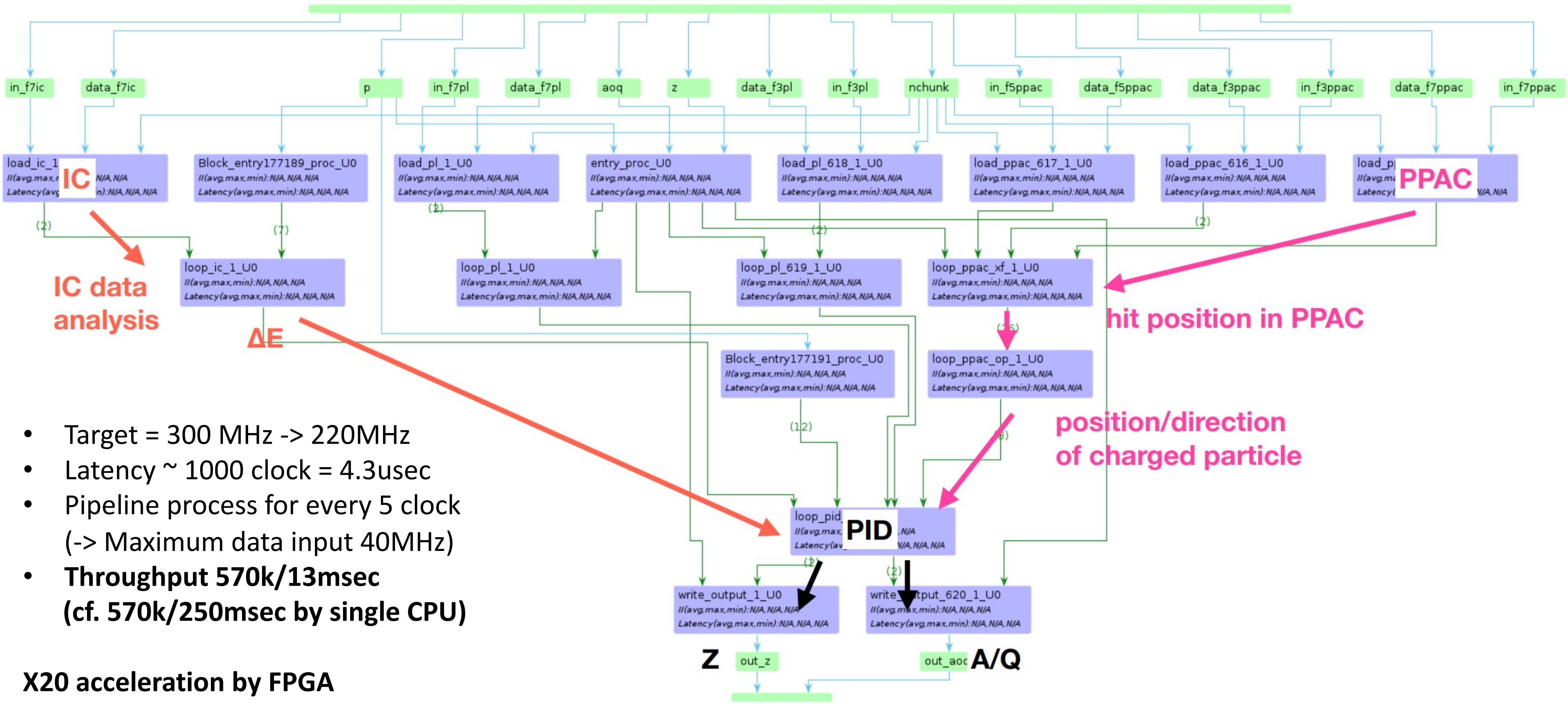




FPGA Processing of BLD data

Ichinohe (C07.4)

Full dataflow of PID procedure



- Target = 300 MHz -> 220MHz
- Latency ~ 1000 clock = 4.3usec
- Pipeline process for every 5 clock (-> Maximum data input 40MHz)
- **Throughput 570k/13msec (cf. 570k/250msec by single CPU)**

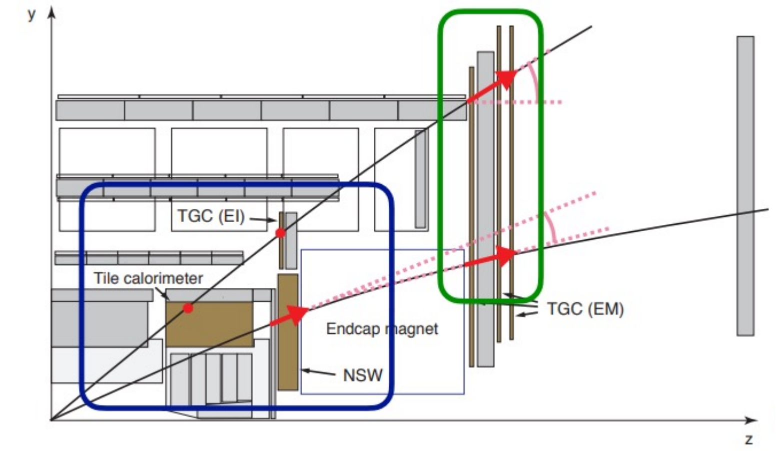
X20 acceleration by FPGA

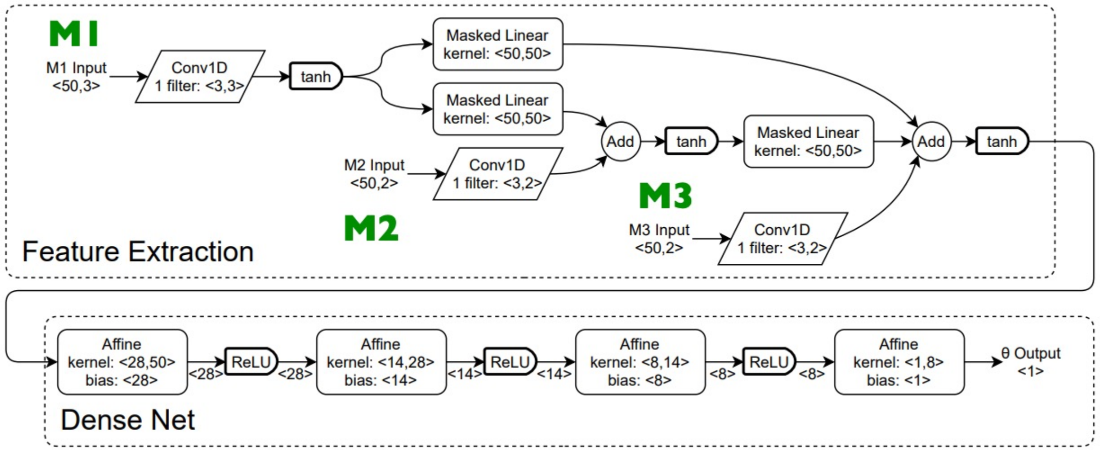
Further developments : hls4ml

Examples of developments in Japan (ATLAS and Belle2)

US also has a lot of activities with hls4ml for sPHENIX, J-lab, LHC experiments, EIC ….

Horii et al. (Nagoya)





M1

M1 Input <50,3>

Conv1D 1 filter: <3,3>

tanh

Masked Linear kernel: <50,50>

Masked Linear kernel: <50,50>

Add

tanh

Masked Linear kernel: <50,50>

Add

tanh

M2 Input <50,2>

Conv1D 1 filter: <3,2>

M3

M3 Input <50,2>

Conv1D 1 filter: <3,2>

M2

Feature Extraction

Affine kernel: <28,50> bias: <28>

<28> ReLU <28>

Affine kernel: <14,28> bias: <14>

<14> ReLU <14>

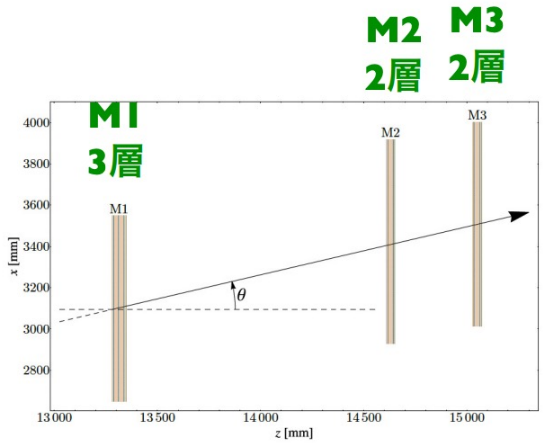
Affine kernel: <8,14> bias: <8>

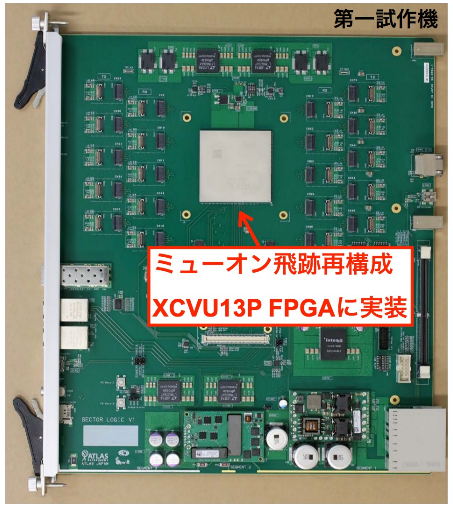
<8> ReLU <8>

Affine kernel: <1,8> bias: <1>

θ Output <1>

Dense Net

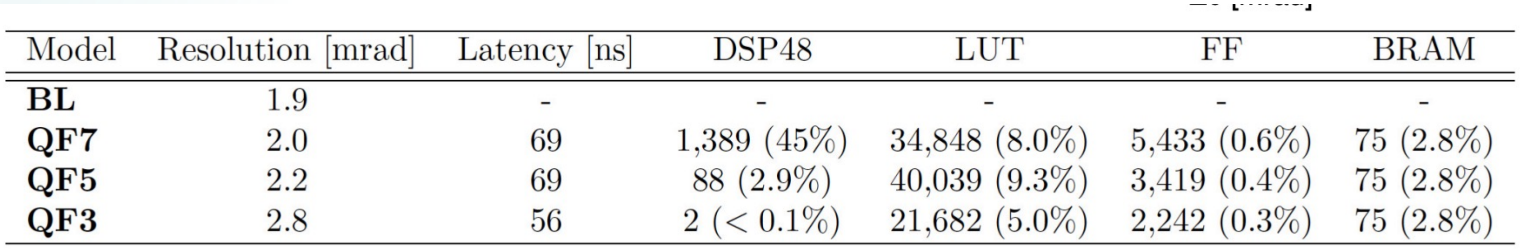




ATLAS Muon tracking using hls4ml

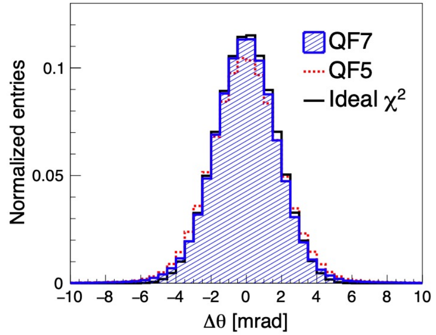
nima.2022.167546

160 MHz clock, Latency = 100nsec



Model	Resolution [mrad]	Latency [ns]	DSP48	LUT	FF	BRAM
BL	1.9	-	-	-	-	-
QF7	2.0	69	1,389 (45%)	34,848 (8.0%)	5,433 (0.6%)	75 (2.8%)
QF5	2.2	69	88 (2.9%)	40,039 (9.3%)	3,419 (0.4%)	75 (2.8%)
QF3	2.8	56	2 (< 0.1%)	21,682 (5.0%)	2,242 (0.3%)	75 (2.8%)

リソース使用率は、Super Logic Regionあたりの値

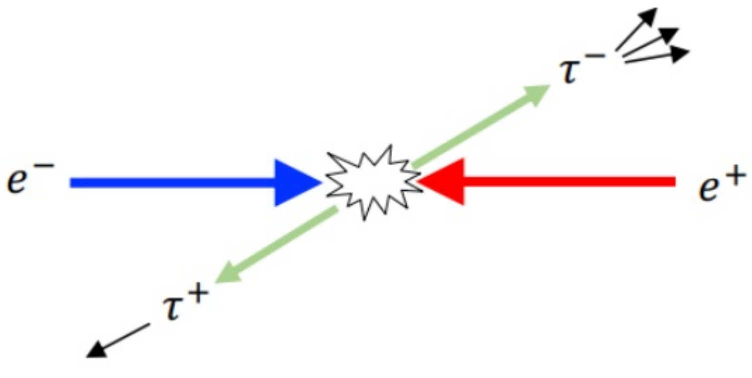


Further developments ： hls4ml

Examples of developments in Japan (ATLAS and Belle2)

US also has a lot of activities with hls4ml for sPHENIX, J-lab, LHC experiments, EIC ….

Koga et al. (KEK)

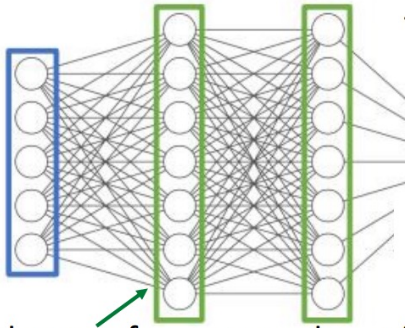


-Input a few tens nodes:

-CDC charged track (ϕ,p_t)

-ECL calorimeter cluster (θ,ϕ,E)

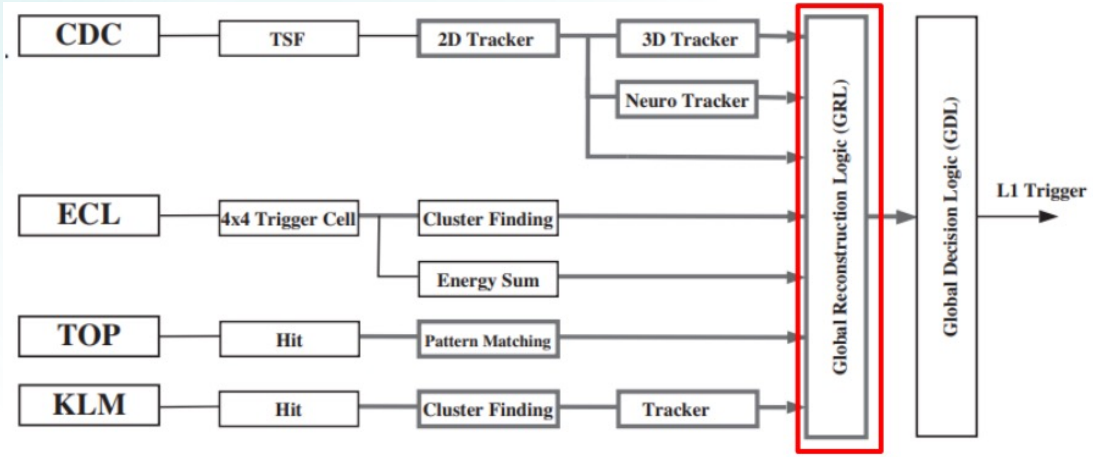
-KLM muon track (θ,ϕ)



-Output 1 node:

-Signal probability

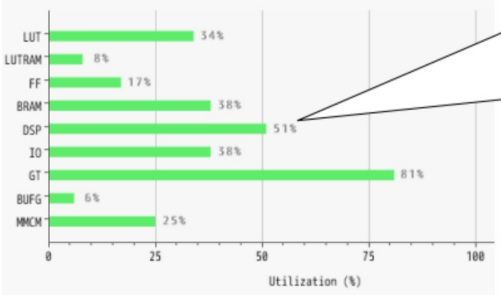
hidden layer: a few tens nodes, a few layers



-Successfully implemented to Xilinx Virtex Ultrascale (XCVU080)

-include not only NN but also original logics on GRL

-latency: 14clock × 127MHz = ~0.11μs



-DSP usage increased from 0% to 51% by implementation of NN

-Others are not so affected

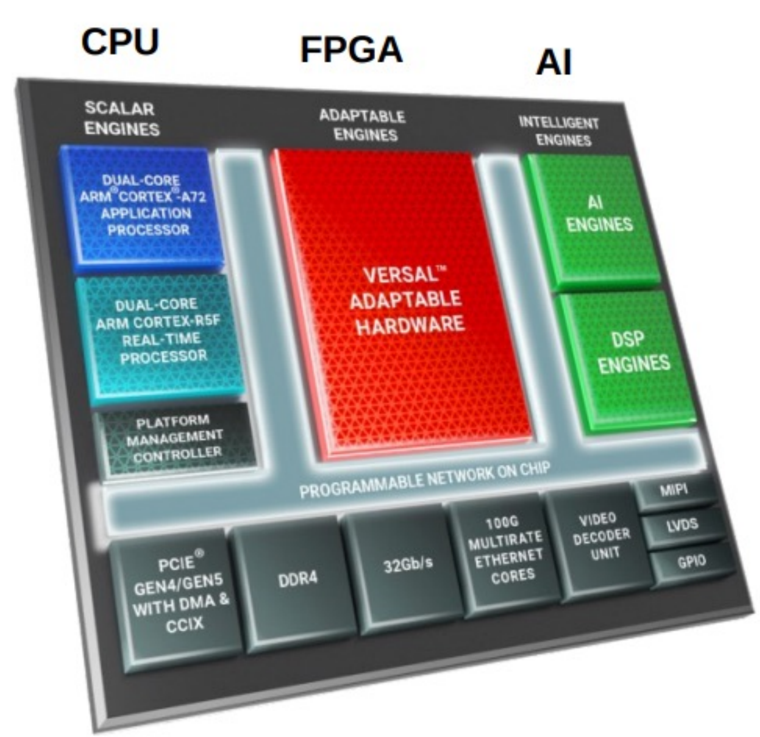
-Will be used for the next physics operation in 2024

Further developments : Versal

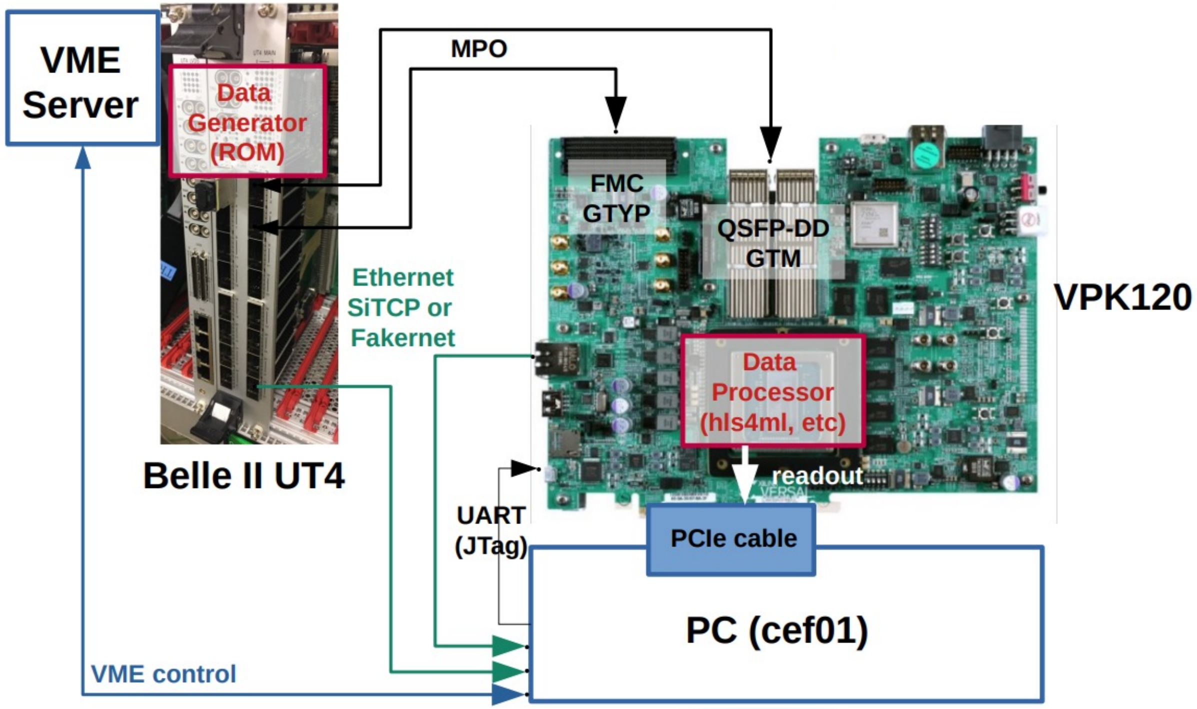
Versal Projects in Japan (Collider electronics Forum in Japan)

The features of Versal series:

- ACAP SoC.
- AI/DSP engine: interface to implement ML core into firmware.
- High Bandwidth Memory (HBM).
- Larger number of cells + High transmission bandwidth.


CPU
FPGA
AI
SCALAR ENGINES
ADAPTABLE ENGINES
INTELLIGENT ENGINES
DUAL-CORE ARM® CORTEX®-A72 APPLICATION PROCESSOR
DUAL-CORE ARM CORTEX-R5F REAL-TIME PROCESSOR
PLATFORM MANAGEMENT CONTROLLER
VERSAL™ ADAPTABLE HARDWARE
AI ENGINES
DSP ENGINES
PROGRAMMABLE NETWORK ON CHIP
PCIE® GEN4/GEN5 WITH DMA & CCIX
DDR4
32Gb/s
100G MULTIRATE ETHERNET CORES
VIDEO DECODER UNIT
MIPI
LVDS
GPIO

Test bench in KEK (lead by HEP community) and extension plans



Further developments ： FELIX Card

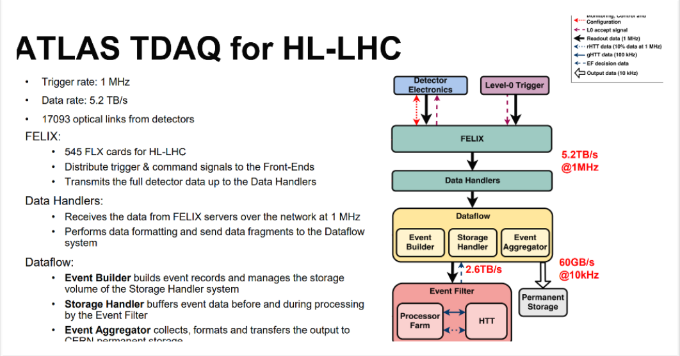
Front-End Link eXchange (FELIX)

https://atlas-project-felix.web.cern.ch/atlas-project-felix/

- Custom FPGA based PCIe cards
- Readout, trigger, clock distribution, Slow Control, BUSY
- Router between FE serial links and commercial network

The FLX-182 card ATLAS Phase-2 Upgrade (Run4)

- FPGA: AMD Versal Prime VM1802
- 4 Firefly transceivers to support 24 bidirectional optical links
  - Up to 25 Gb/s per link
- 1 Firefly for LTI/TTC interface
  - New protocol for Timing, Trigger and Control
  - 100Gb/s Ethernet or White Rabbit are optional
- 16-lane PCIe Gen4 interface (240 Gb/s)
  - 2x 8 lanes bifurcated



ATLAS TDAQ for HL-LHC

- Trigger rate: 1 MHz
- Data rate: 5.2 TB/s
- 17093 optical links from detectors

FELIX:

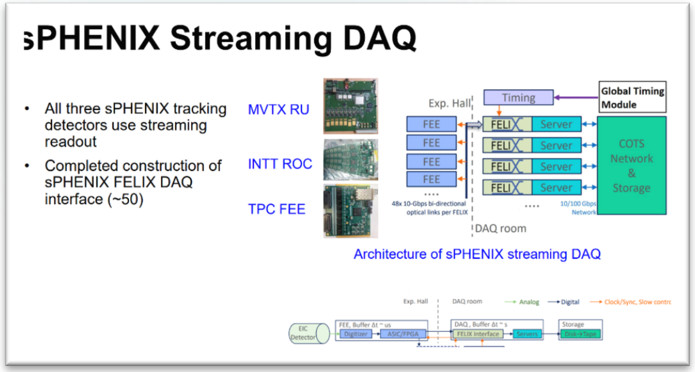
- 545 FLX cards for HL-LHC
- Distribute trigger & command signals to the Front-Ends
- Transmits the full detector data up to the Data Handlers

Data Handlers:

- Receives the data from FELIX servers over the network at 1 MHz
- Performs data formatting and send data fragments to the Dataflow system

Dataflow:

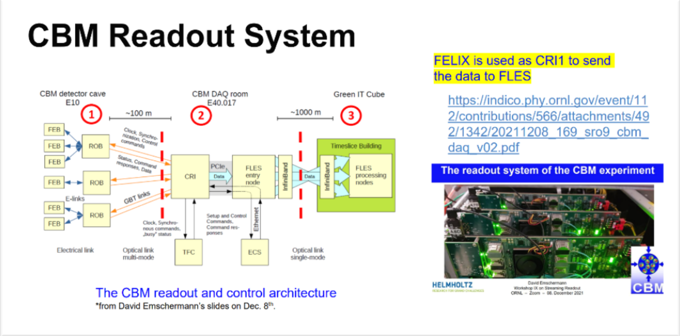
- **Event Builder** builds event records and manages the storage volume of the Storage Handler system
- **Storage Handler** buffers event data before and during processing by the Event Filter
- **Event Aggregator** collects, formats and transfers the output to CERN permanent storage



sPHENIX Streaming DAQ

- All three sPHENIX tracking detectors use streaming readout
- Completed construction of sPHENIX FELIX DAQ interface (~50)

Architecture of sPHENIX streaming DAQ



CBM Readout System

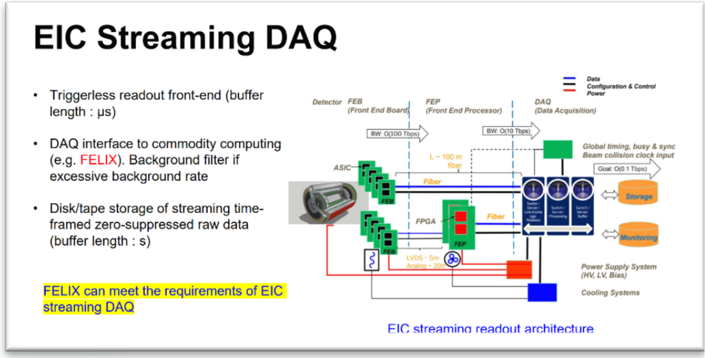
FELIX is used as CRI1 to send the data to FLES

https://indico.phy.ornl.gov/event/112/contributions/566/attachments/492/1342/20211208_169_sro9_cbm_daq_v02.pdf

The readout system of the CBM experiment

The CBM readout and control architecture

*from David Emschermann's slides on Dec. 8th.



EIC Streaming DAQ

- Triggerless readout front-end (buffer length : μs)
- DAQ interface to commodity computing (e.g. FELIX). Background filter if excessive background rate
- Disk/tape storage of streaming time-framed zero-suppressed raw data (buffer length : s)

FELIX can meet the requirements of EIC streaming DAQ

EIC streaming readout architecture

FELIX becoming more widely used in various experiments