PAUL SCHERRER INSTITUT

PSI

Filip Leonarski :: Beamline Data Scientist :: MX Data Group

Jungfraujoch: a Data Acquisition and On-the-fly Analysis System for HDR MX

14th August 2021

Topics

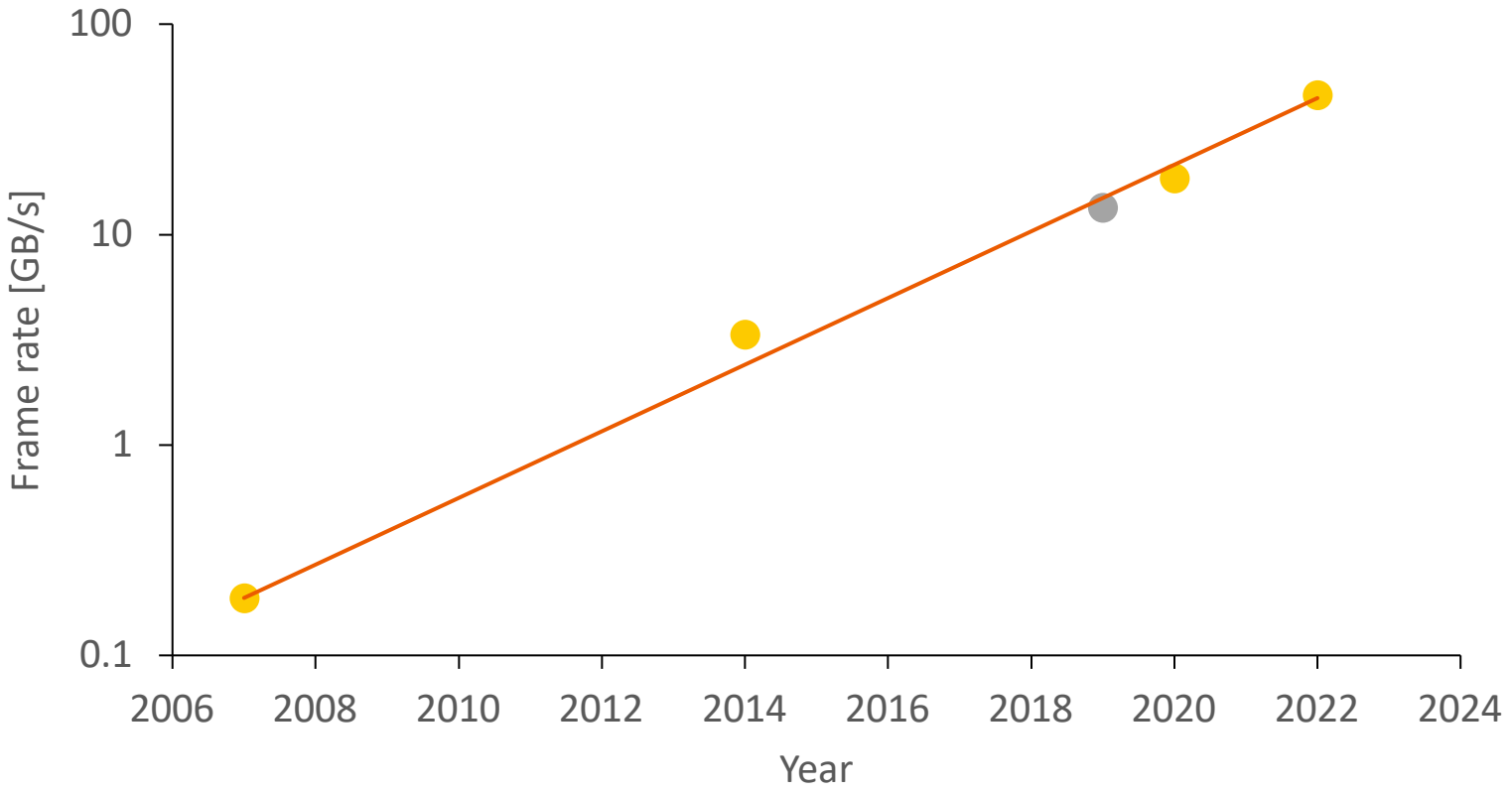
- FPGA based detector data acquisition
- Compression for diffraction images

PAUL SCHERRER INSTITUT

PSI

FPGA based detector data acquisition

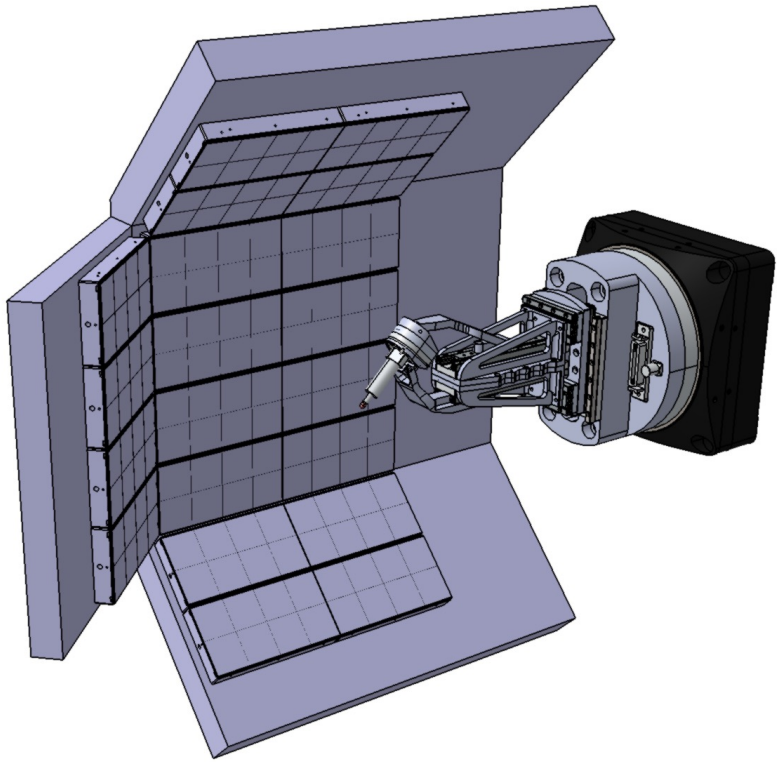
MX detector data rates double every 2 years



2007	PSI PILATUS	6 Mpixel	12.5 Hz	**0.2 GB/s**
2014	Dectris EIGER	16 Mpixel	133 Hz	**3.4 GB/s**
2019	Dectris EIGER 2 XE	16 Mpixel	400 Hz	**13.5 GB/s**
2020	PSI JUNGFRAU	4 Mpixel	2200 Hz	**18.4 GB/s**
2022	PSI JUNGFRAU	10 Mpixel	2200 Hz	**46.1 GB/s**

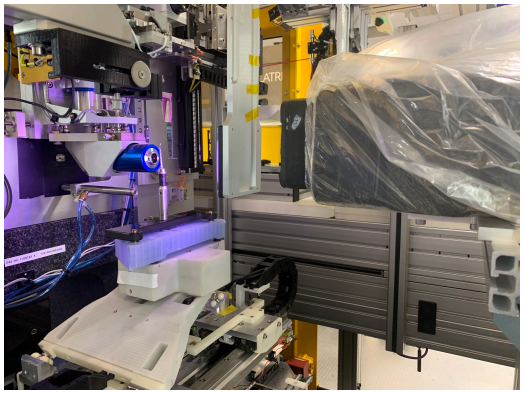
Task: JUNGFRAU 10 Mpixel in 2022

- Beamline PXIII was selected as a pilot project for SLS 2.0 (Phase 0) – it will get major refurbishment in 2022
- One of developments is tiled JUNGFRAU 10 Mpixel detector for native-SAD applications – producing up to 46 GB/s
- Need smooth integration of the detector into beamline
- 4 Mpixel system planned for installation this year



Task: JUNGFRAU integration

- What is needed to operate JUNGFRAU detector?
  - UDP receiver
  - Conversion from raw to energy/photons
  - Peak finder for fast experimental feedback
  - Bitshuffle prefilter
  - Write to memory
- JUNGFRAU at a synchrotron must be internally operated at ≥ 1 kHz frame rates for continuous measurement at synchrotron
- Each of these task is difficult at 46 GB/s
- All tasks have low computational complexity and simple logic, but data throughput is high

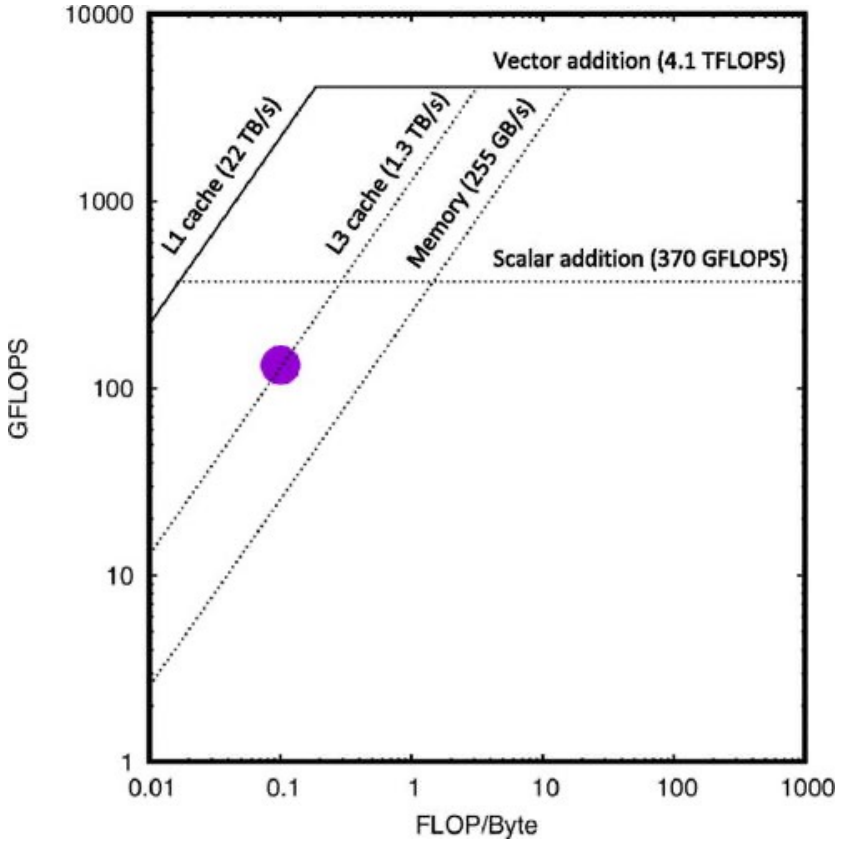


VMXi DLS
JUNGFRAU 1M tested at multilayer monochromator high-flux beamline

JUNGFRAU 4M tested at **X06DA Swiss Light Source** (CH) – special box to allow to cool detector to – 15°C

Conventional hardware results

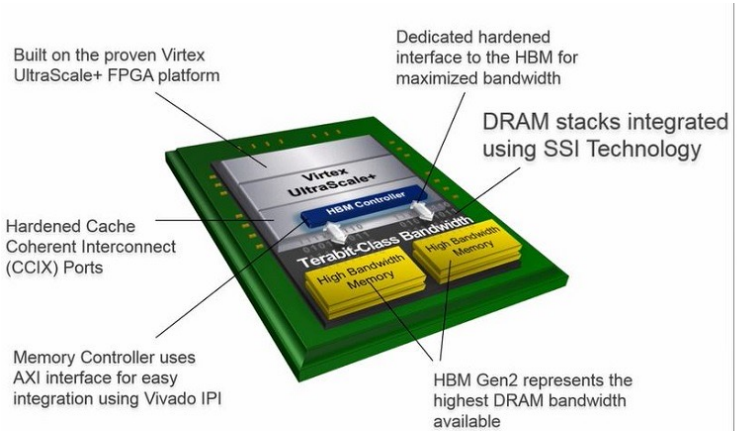
- Raw to photon conversion example
- The most powerful single server available from PSI vendor
  - 4 CPU socket, 1.5 TB RAM, NVMe SSD drives and Mellanox fiber ethernet network cards
- Profiling and performance tuning
- Outcome: the server can handle JUNGFRAU 4Mpixel at 550 Hz, but not more
- Bottleneck: **memory bandwidth of CPU is too small** for conversion procedure



See: "JUNGFRAU detector for brighter x-ray sources: Solutions for IT and data science challenges in macromolecular crystallography" Leonarski et al. Structural Dynamics (2019) https://doi.org/10.1063/1.5143480

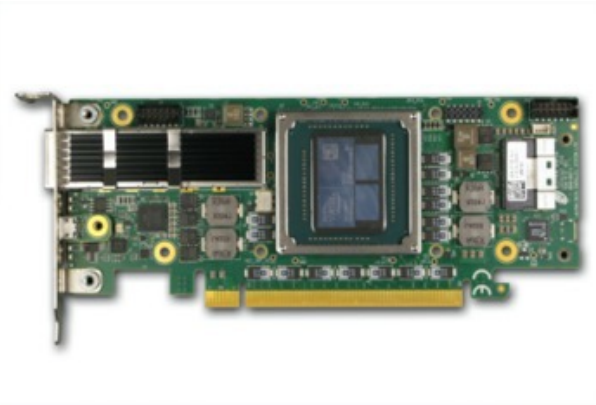
Field programmable gate arrays (FPGA)

- FPGAs
  - Real-time device:
    design guarantees throughput and latency
  - Fast memory bandwidth:
    up to 460 GB/s per chip with HBM2
  - Vast I/O options:
    100G ethernet, PCIe, ...



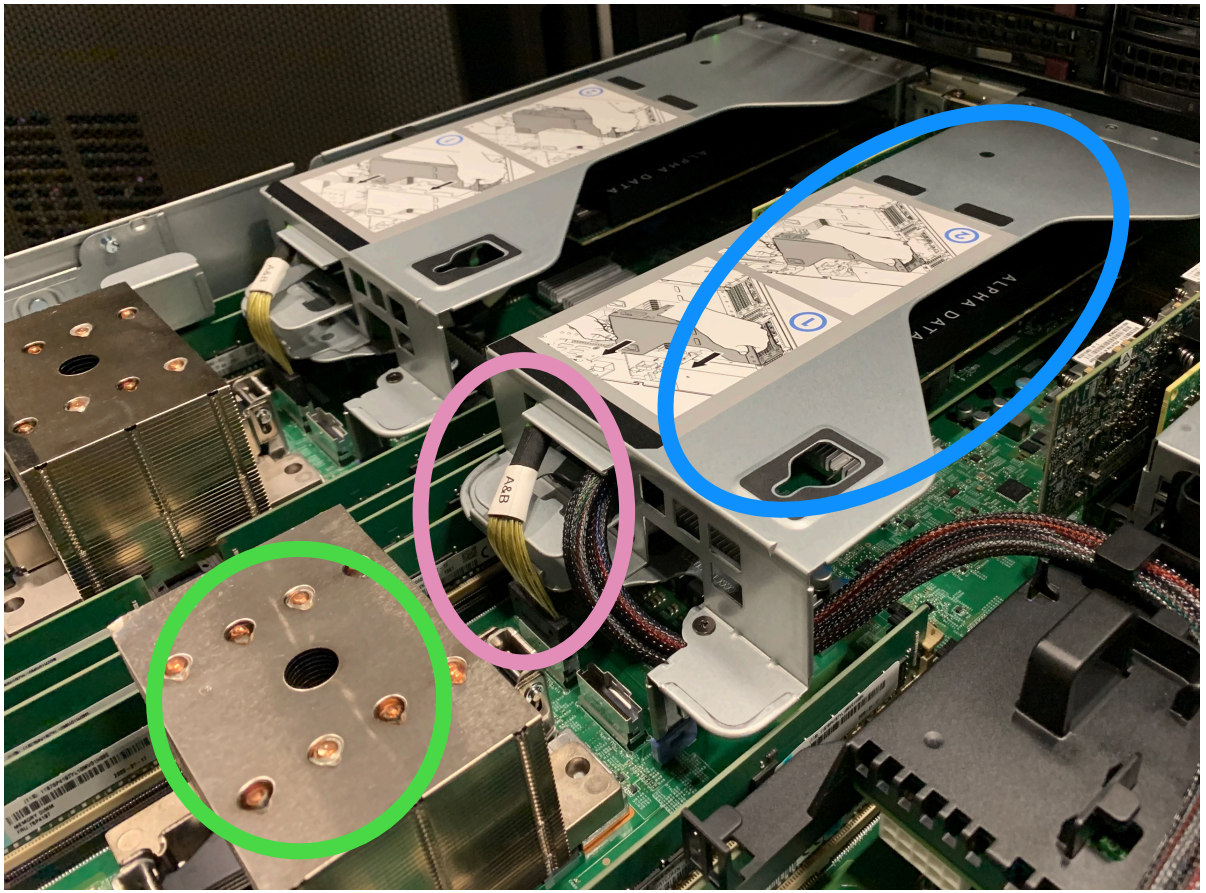
- Also considered GPUs, but lacked necessary functionality

- Very powerful, but significant effort in development, due to need of specific hardware design skills and hardware description languages



Alpha-Data 9H3 board

OpenCAPI



POWER9 CPU **OpenCAPI cable** **FPGA board**

What difference brings OpenCAPI?

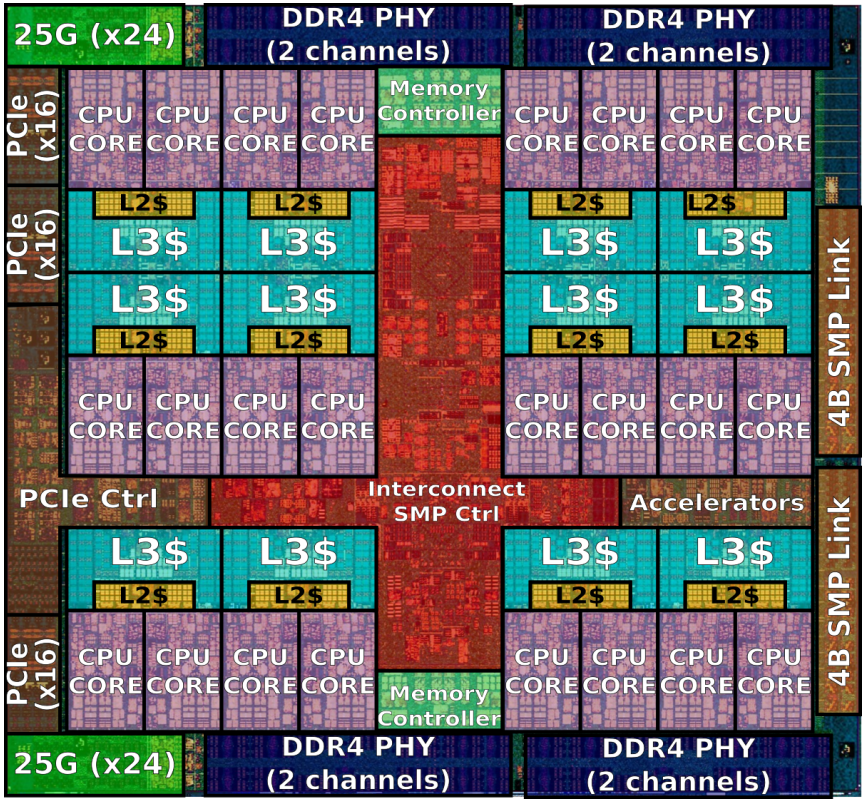
- Similar to **virtual mode** starting from 80286/80386 CPUs (as opposed to real mode), which hides complexity of system memory

- Each process has its own virtual address space
  - **Simplicity**: virtual address space is not dependent on system physical memory
  - **Safety**: there is no (easy) way to access memory of another process
- Translation virtual <-> physical memory is done by CPU electronics

Source: Wikipedia

What difference brings OpenCAPI?

- Standard HW interfaces, like PCIe, are not aware of virtual address space – developer need to address issue, increasing complexity
- OpenCAPI allows external accelerators (like FPGA) to attach directly to a virtual address space of the process
- Drastically reduced complexity of interconnect



POWER9
WikiChip

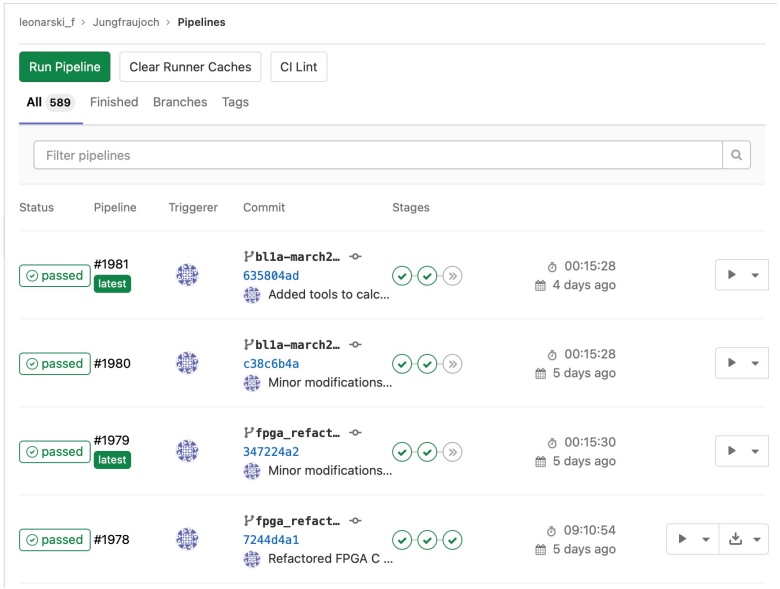
OpenCAPI 25G

FPGA as software development project

- Aim: (relatively) fast transformation of SW based algorithms to FPGA

- **High-level synthesis** -> compile C++ functions into hardware description language
  - Not as efficient as pure HDL
  - Easier to write
  - Easier to test and plug into CI pipeline

- **Memory coherent interconnect** (e.g. OpenCAPI, Intel CXL)
  - Virtual, not physical, addressing
  - CPU cache coherent
  - No need for OS/kernel expertise
  - Requires special hardware (at the moment)



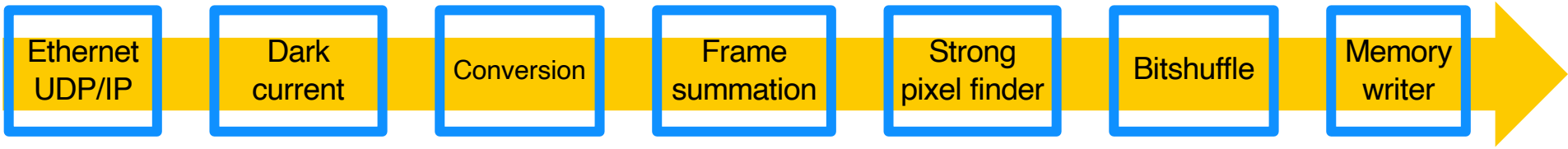
Both C++ verification and HDL simulation are part of CI:

C++ 8 minutes to cover 13 scenarios and 95% of code

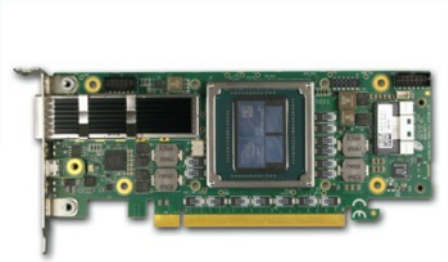
HDL 4 hours to cover single simple scanario

Jungfraujoch server

Up to **50 GB/s acquisition and data analysis** in a single 2U IBM POWER9 server with 1-4 FPGA boards



FPGA board with OpenCAPI interface



- Data acquisition
- Initial data analysis
- Pre-compression

(2.5 Mpixel/board for JF)

Jungfraujoch & Gold Standard

```
leonarski_f — leonarski_f@pc13660:/home/filip/Jungfraujoch/build/tests — ssh hop.psi.ch — 9...
[leonarski_f@pc13660 tests]$ ./HDF5DatasetWriteTest ../../tests/test_data/compression_benchmark.h5
JF4M with gaps detected (2068 x 2164)
Number of images in the dataset: 100
Images loaded
Done
[leonarski_f@pc13660 tests]$ /home/filip/cnxvalidate/build/nxvalidate -a NXmx -l /home/filip/defini
tions test_master.h5
test_master.h5 passed validation
[leonarski_f@pc13660 tests]$
```

IUCrJ
ISSN: 2052-2525
BIOLOGY | MEDICINE

Volume 7 | **Part 5** | September 2020 | Pages 784-792
https://doi.org/10.1107/S2052252520008672
OPEN ACCESS
Cited by 1

Gold Standard for macromolecular crystallography diffraction data

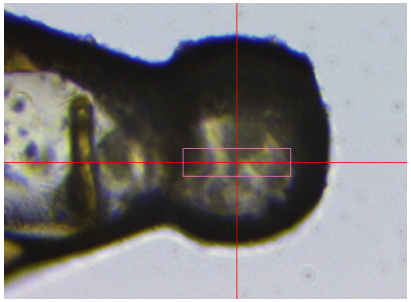
Herbert J. Bernstein,[a*] Andreas Förster,[b] Asmit Bhowmick,[c] Aaron S. Brewster,[c] Sandor Brockhauser,[d,e,f] Luca Gelisio,[g] David R. Hall,[h] Filip Leonarski,[i] Valerio Mariani,[g] Gianluca Santoni,[j] Clemens Vonrhein[k] and Graeme Winter[h]

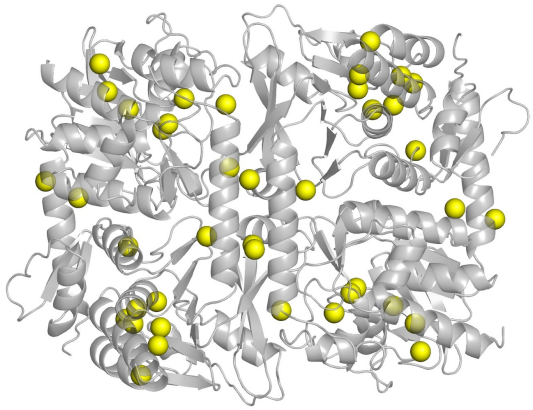
Dectris Neggia ✅ Dectris Albula ✅ DIALS ✅

Commissioning in KEK (Jan – May 2021)

- Detector and data acquisition system was sent in November for an experiment in Photon Factory, KEK
- More than 2,000 datasets collected for protein targets, few real-life native-SAD structures solved
- Due to pandemic, detector support and development (including deployment of new FPGA design) was done fully remotely from Switzerland

BL-1A Photon Factory
JUNGFRAU detector (up) tested in helium chamber for native-SAD measurements with 3.75 keV X-rays





Commissioning in KEK (Jan – May 2021)

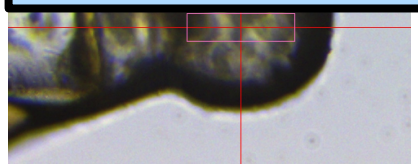
- Detector and data acquisition system was sent in November for an experiment in Photon Factory, KEK

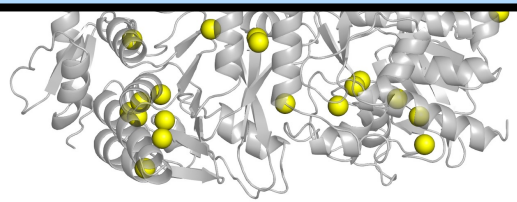
Results will be presented in another presentation:

The crystallomics pipeline, a shotgun approach on native proteomes to (re)discover the unsuspected

by **Sylvain Engilberge**

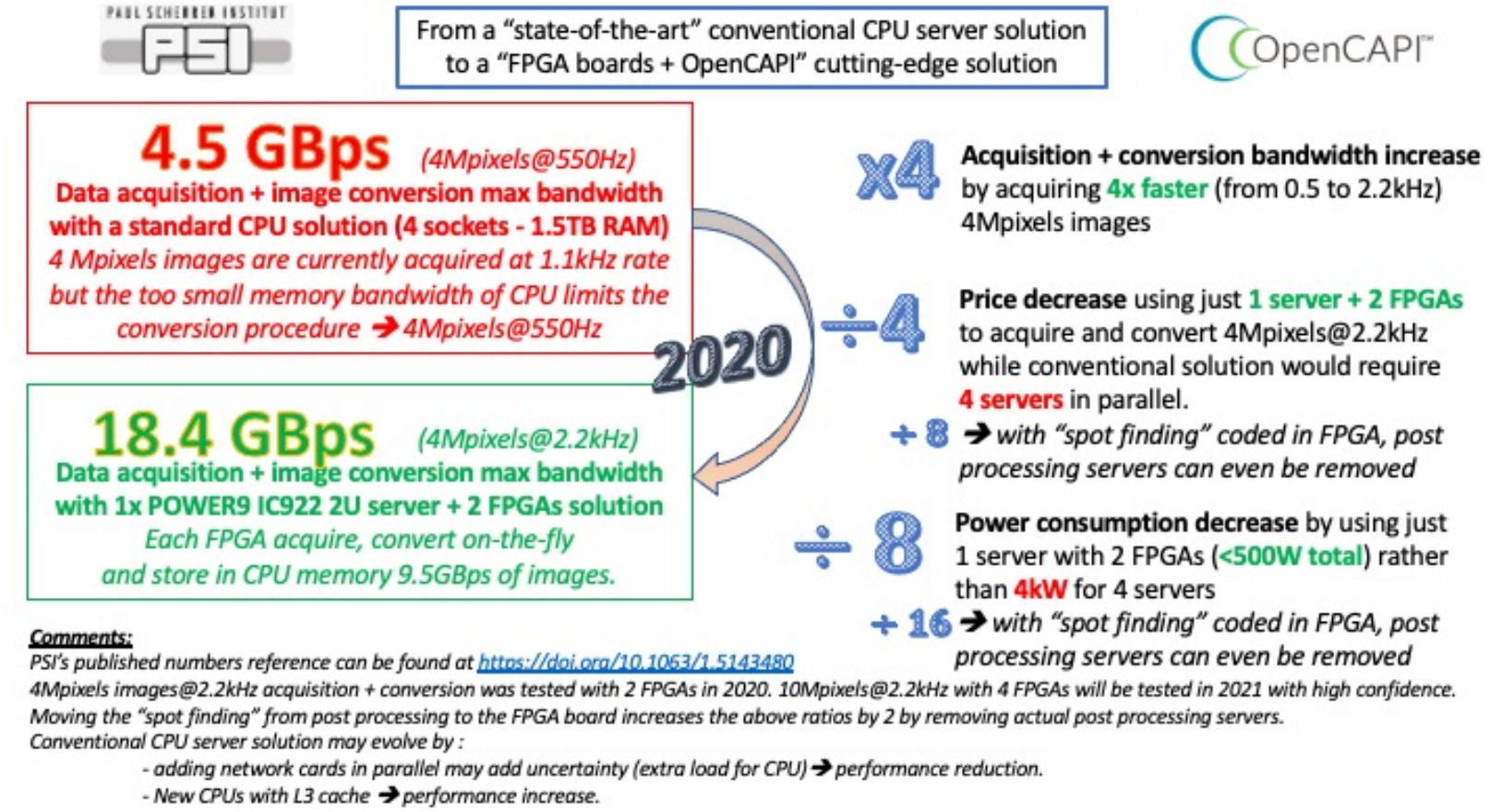
18.08 11:45





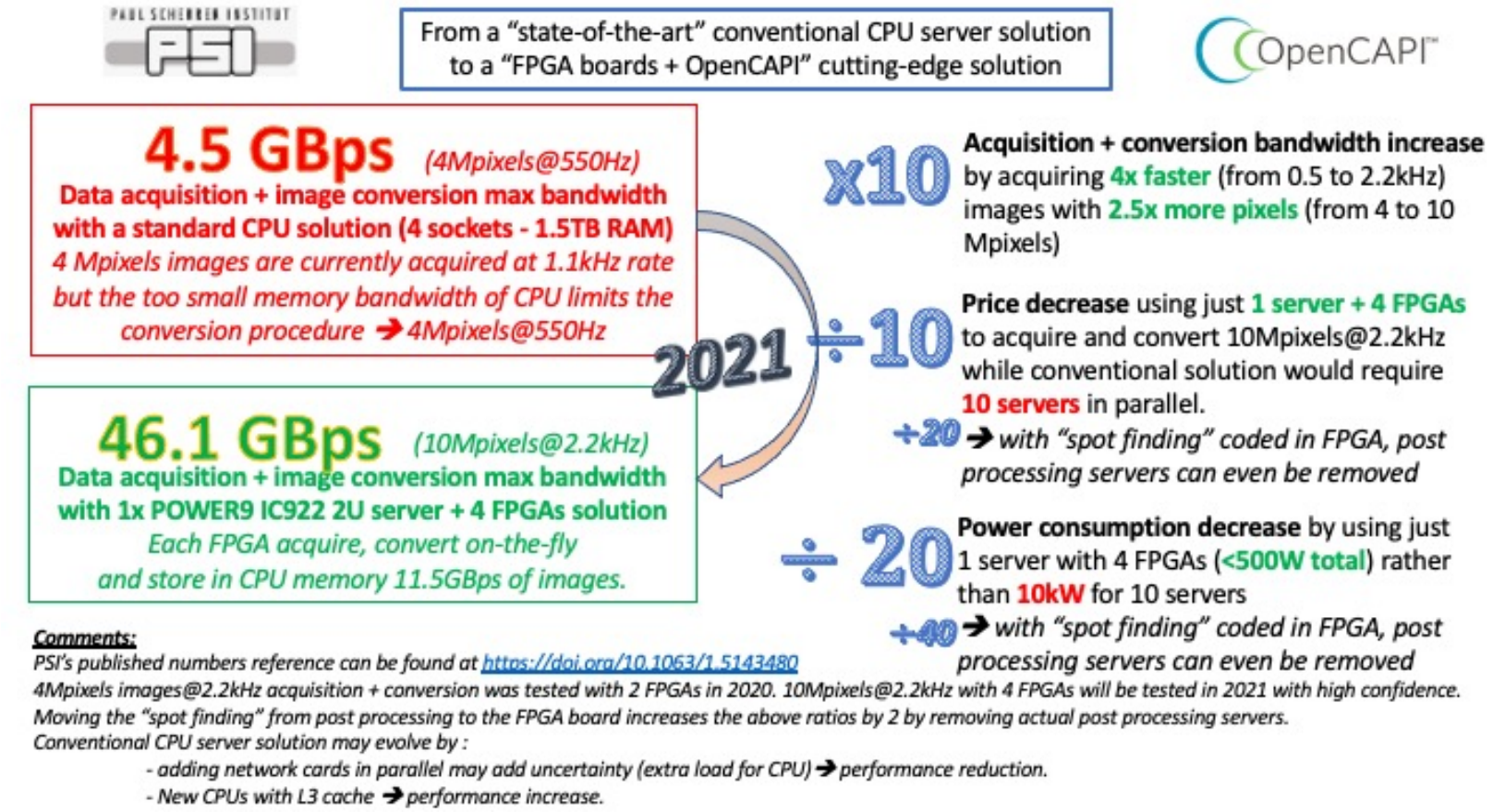
measurements with 3.75 keV X-rays

Possible gain from using FPGA based system



Courtesy: B. Mesnet (IBM)

Possible gain from using FPGA based system



Courtesy: B. Mesnet (IBM)

PAUL SCHERRER INSTITUT

PSI

Compression of diffraction images

Efficient compression should be complementary to HW development

- FPGA processing can help in early stages of the pipeline, however transfer and storage remain an issue - efficient compression can be a good answer

- Bitshuffle/LZ4 introduced with EIGER images is currently dominant

- There is lot of happening in the community:
  - Argonne: SZ lossy compression project for ExaFEL
    https://arxiv.org/pdf/2105.11730.pdf
  - Argonne: On-chip compression of X-ray data following counting statistics.
    https://doi.org/10.1107/S1600577520013326
  - RIKEN and FSU: TEZIP based on recurent neural networks
    https://ieeexplore.ieee.org/document/9499386

- Is HDR MX community ready to embrace these developments?
- Is there a way to tweak bitshuffle/LZ4?

How does current compression work?

Bitshuffle

- Precompression filter
- Makes compression easier, but there is no actual data reduction at this step

LZ4

- Popular lossless compressor
- Dictionary algorithm
- Byte oriented

Byte compression of pixels in LZ4

1	0	0	2	0	0	1	0

- Algorithm like LZ4 try to find repeatable sequences of bytes, which are replaced by some special tokens
- They focus on bytes, as texts are encoded as bytes
- The longer these sequences are – the more efficient is compression
- However, it is hardly expected to find repeatable byte sequences in diffraction images

Bitshuffle filter – input sequence

1	0	0	2	0	0	1	0

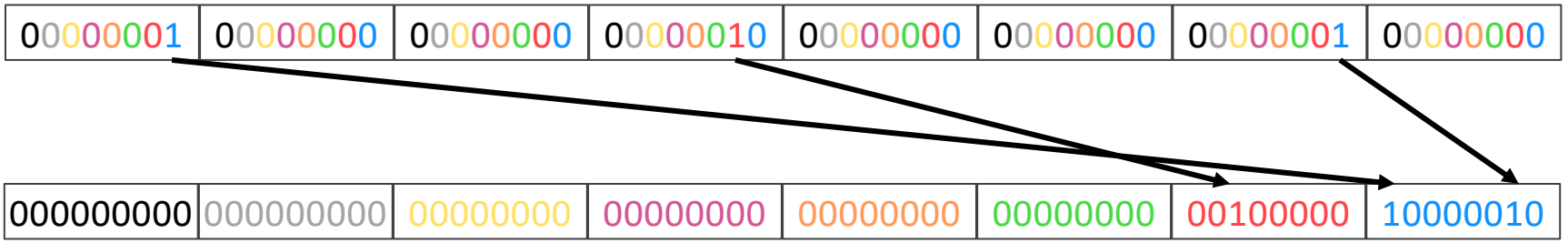
Bitshuffle filter – input sequence in binary

1	0	0	2	0	0	1	0

00000001	00000000	00000000	00000010	00000000	00000000	00000001	00000000

Bitshuffle filter – shuffling operation

1	0	0	2	0	0	1	0



Bitshuffle filter - outcome

1	0	0	2	0	0	1	0

00000001	00000000	00000000	00000010	00000000	00000000	00000001	00000000

000000000	000000000	00000000	00000000	00000000	00000000	00100000	10000010

0	0	0	0	0	0	32	130

Long sequence of zero – low entropy: trivial to compress

High entropy: unlikely to compress this part

Bitshuffle filter - outcome

1	0	0	2	0	0	1	0

00000001	00000000	00000000	00000010	00000000	00000000	00000001	00000000

000000000	000000000	00000000	00000000	00000000	00000000	00100000	10000010

0	0	0	0	0	0	32	130

Long sequence of zero – low entropy: trivial to compress

High entropy: unlikely to compress this part

MX diffraction images (esp. collected at high frame rates) contain mostly low counts (background) and small number of high counts (Bragg spots)

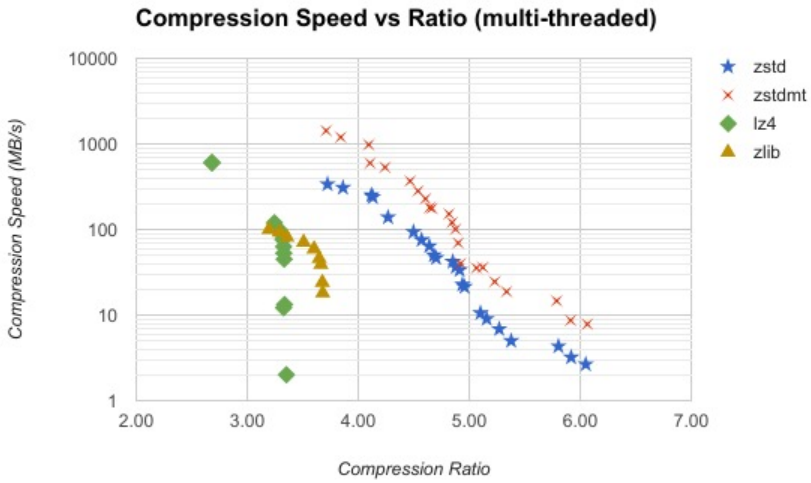
Bitshuffle implementation

- On CPU, bitshuffle is implemented as combination of some bit instructions – there is no perfect CPU instruction to do it
- Bitshuffle step is generally fast (GB/s), but it will be a bottleneck, if compression is also fast
- Doing bitshuffle on FPGA is trivial – just connecting wires properly between source and destination

LZ4 has excellent performance, but no flexibility

- LZ4 is made to be extremely fast to decompress, relatively simple in construction
- Y. Collet, author of LZ4, made new algorithm called Zstandard (at Facebook)
- github.com/facebook/zstd
- Zstandard is similar to LZ4, but has more options within the algorithm
- **By setting Zstdandard compression level, one can set tradeoff between performance and compression ratio**

https://gregoryszorc.com/blog/2017/03/07/better-compression-with-zstandard/



Zstandard for diffraction image

- Lysozyme dataset collected with JF4M

	Compression ratio	Throughput
Bshuf+LZ4	7.0x	2.5 GB/s
Bshuf+Zstandard (default)	8.0x	0.6 GB/s
Bshuf+Zstandard (+10)	8.3x	0.2 GB/s
Bshuf+Zstandard (-10)	6.8x	1.9 GB/s

- Zstandard provides flexibility, not available with LZ4
- In general Zstandard compression is slower than LZ4

All performance numbers from AMD Ryzen 5950X

Zstandard for diffraction image

- Lysozyme dataset collected with JF4M

	Compression ratio	Throughput
Bshuf+LZ4	7.0x	2.5 GB/s
Bshuf+Zstandard (default)	8.0x	0.6 GB/s
Bshuf+Zstandard (+10)	8.3x	0.2 GB/s
Bshuf+Zstandard (-10)	6.8x	1.9 GB/s

- Zstandard provides flexibility, not available with LZ4
- In general Zstandard compression is slower than LZ4

- **Zstandard can benefit from large compression blocks. With 1 MB block compression factor of 9.0x could be reached with 1.1 GB/s throughput.** LZ4 doesn't gain from larger block size.

Speed-up in Jungfraujoch

- Do bitshuffle in FPGA and transfer bitshuffled images to host memory (currently 8 kB block)

	Compression ratio	Throughput
LZ4 (bshuf on CPU)	7.0x	2.5 GB/s
LZ4 (bshuf on FPGA)	7.0x	4.0 GB/s

Speed-up in Jungfraujoch

- Do bitshuffle in FPGA and transfer bitshuffled images to host memory (currently 8 kB block)
- Use custom Zstandard compressor – only compress long sequences of zeroes (at least 5 bytes), no compression for other byte values; standard decompressor

	Compression ratio	Throughput
LZ4 (bshuf on CPU)	7.0x	2.5 GB/s
Zstandard (bshuf on CPU)	8.0x	0.6 GB/s
LZ4 (bshuf on FPGA)	7.0x	4.0 GB/s
Zstandard/RLE (bshuf on FPGA)	6.4x	**7.4 GB/s**

Speed-up in Jungfraujoch

- Do bitshuffle in FPGA and transfer bitshuffled images to host memory (currently 8 kB block)
- Use custom Zstandard compressor – only compress long sequences of zeros (and ones), don't do any other compression; standard decompressor

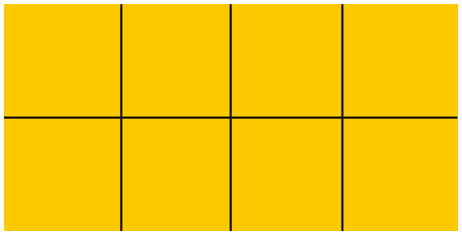
Just compressing zeros of bitshuffled data accounts from most of compression and can be done extremly efficient (incl. FPGA implementation).

To get beyond this, one needs significant effort.

Compression ratio	Throughput
7.0x	2.5 GB/s
8.0x	0.6 GB/s
7.0x	4.0 GB/s
6.4x	**7.4 GB/s**

There is one issue - multipixels

- EIGER and JUNGFRAU modules are both 1024 col x 512 lines
- However pixels on ASIC boundaries are larger, leading to double sized pixel
- In post-processing these pixels are replaced with 2 (edges) or 4 pixels (corners)
- Resulting 1030x514 pixel image better represents geometry, but has worse properties for splitting into packets and blocks (not power-of-two)
- It is relatively complex to add these multipixels in electronics (FPGA) + they add small to image size without extra information



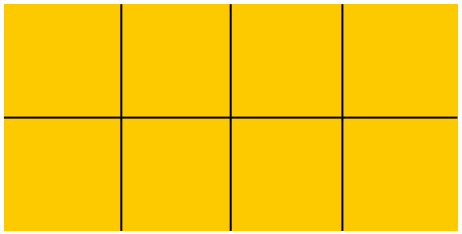
There is one issue - multipixels

- EIGE
  1024
- Howe
  to do
- In po
  (edge
- Resul
  geom
  packe
- It is r
  electr
  witho

Requirement to have multipixels precludes bitshuffling on FPGA at the moment.

The solution is simple – add these at decompression with special HDF5 plugin (in pipeline for development).

Is there a more elegant solution?



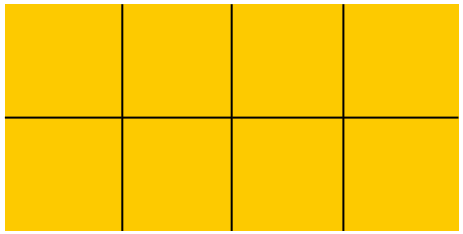
There is one issue - multipixels

- EIGER
  1024
- Howe
  to do
- In po
  (edge
- Resul
  geom
  packe
- It is r
  elect
  witho

Requirement to have multipixels precludes bitshuffling on FPGA at the moment.

The solution is simple – add these at decompression with special HDF5 plugin (in pipeline for development).

Is there a more elegant solution?



How about placeholders for module gaps? These are even bigger problem and even less useful.
But here GOLD STANDARD is an answer!

Summary

Jungfraujoch is a fully integrated DAQ solution for kilohertz frame rate JUNGFRAU detectors – deployed on X06DA beamline this summer for user operation

More focus is necessary for compression development – flexibility of Zstandard can come handy for future data rates

Acknowledgements

MX Group (PSI)
- Vincent Olieric
- Takashi Tomizaki
- Chia-Ying Huang
- Sylvain Engilberg
- Justyna Wojdyła
- Meitian Wang

Detector Group (PSI)
- Aldo Mozzanica
- Martin Brückner
- Carlos Lopez-Cuenca
- Bernd Schmitt

Science IT (PSI)
- Leonardo Sala

Controls (PSI)
- Andrej Babic
- Leonardo Hax-Damiani

SLS management (PSI)
- Oliver Bunk

Photon Factory, KEK
- Naohiro Matsugaki
- Yusuke Yamada
- Masahide Hikita

MAX IV
- Jie Nan
- Zdenek Matej

Uni Konstanz
- Kay Diederichs

LBL
- Aaron Brewster

DLS
- Graeme Winter

ESRF
- Jerome Kieffer

Dectris
- Stefan Brandstetter
- Andres Förster

IBM Systems (France)
- Alexandre Castellane
- Bruno Mesnet

InnoBoost SA
- Lionel Clavien