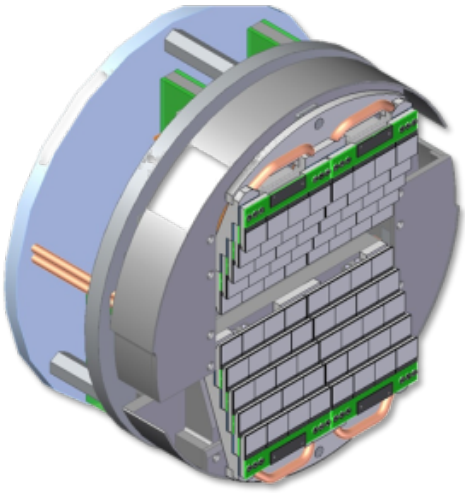
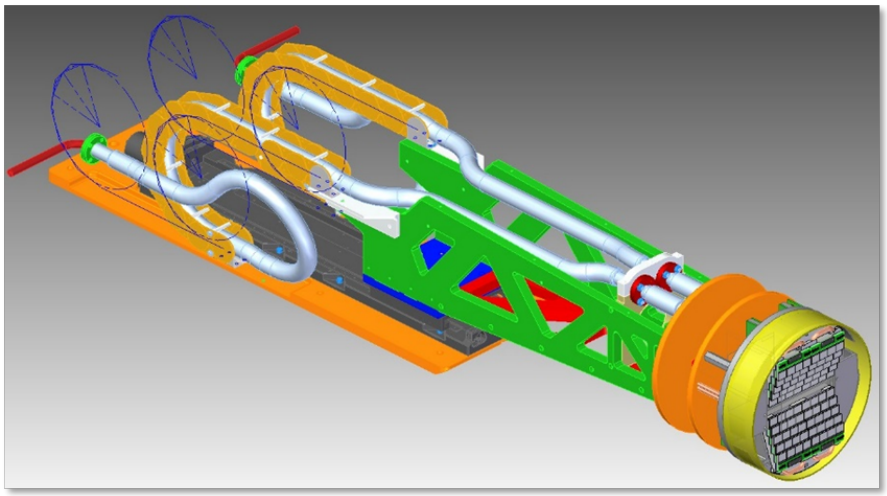
EpixHR

Overview

Our first large area detector (2Mpx, 5kHz). Total data volume 2Mpx*5kHz*2bytes/px=20GB/s. 5 tiles in each quad. Each tile is 4 asics. Each ASICs is 144*192*4 (4 is number of asics).





Some slides from Matt describing how to run epixHR prototype in MFX with LCLS-I timing: https://docs.google.com/presentation/d/1gauhczW2oMYa72_jbCMgenUQj7HGwxyCLvFPvMVpK84/edit?usp=sharing

The GitHub for the epixhr 2x2 prototype is https://github.com/slaclab/epix-hr-single-10k

Issues

Possible fiber configurations (with uniform distribution of tiles across nodes):

- 4 nodes: 5 tiles (1 quad) per node (5GB/s on one node, 21Hz per core (full-image equivalent))
- 10 nodes: 2 tiles per node (2GB/s on one node, 8Hz per core(full-image equivalent))
- 20 nodes: 1 tile per node (1GB/s on one node, 4Hz per core(full-image equivalent))

Is a segment a tile? or a set of tiles (e.g. 2 or 3)?

Two ideas: segments, and "unit cell". Should those ideas be identical?

Would make my brain hurt the least if we made a tile a segment (that's been our previous approach). Also allows us to change the fiber configuration as time goes on without defining a new segment type.

Perhaps we should mock-up the epixhr segment idea in advance using the kcuSim detector?

The configuration is only done on one node. But we can make sure that the configuration object is present on all the relevant nodes (and in the associated xtc file).

Another concern is the node that has the 6 KCU cards for the multimode to single mode conversion of 48 fiber pairs (perhaps minus 4 since in principle the timing can be sent with the local txi xpm using multimode?).

Do we also need to support det.image in the DRP?

Multimode to singlemode conversion:

- Need to buy a big node (like hsd node) to hold 6 kcu cards
- dionisio says this mtp24 is only available in multimode: https://www.mouser.com/ProductDetail/Amphenol-FSI/10124588-310?qs=81r%252BiQLm7BSk9m%252B2puhj7Q%3D%3D.
- see also https://www.amphenol-cs.com/product-series/leap-on-board-transceiver.html
- see also https://www.amphenol-cs.com/optics/transceivers.html
- amphenol tells us to contact our local sales rep: Gaurav.Singh@amphenol-fci.com
- feedthroughs may also be a problem. Kaz is getting part number RETA-RIB-CF40-24-GI50-025-S70-MTPF (24 fibers) and RETA-RIB-CF16-12-GI50-025-S48-MTPF (12 fibers) from sedi-ati.com. These two parts are similar to RETA-STANF21.208 (different length) and RETA-LBNL19.141 (different flange). Looks like feedthroughs are available in single-mode!
- transceiver outputs MPO24 (12 fiber pairs). the vacuum feedthroughs will also be MPO24
    - timing comes from XPM. configuration/monitoring/data fibers connect to DRP. we will use LC plus cassettes to get to MPO24 (and MPO8 going to the multimode to single mode converters).
- need to have/order:
    - a node to hold all the multimode-to-single-mode converter KCUs
    - LC to MPO24 cassettes
    - MPO8 breakout cables

Amphenol says these are "variants" of the transceiver that they support:

blocked URL

SZ Compression

Chuck writes about roibinsz:

> # Convert to ms/image/core
> Given 4M pixels x float32 (4 bytes) = 32 MB/image,
> 89.89GB/32MB => 2809 images/s on 400 cores => 7 images/s/core => 142ms/image/core
> For a 2M detector, this would take half the time => 71ms/image/core

This 71ms number + 20ms for calibration feels tight on 10 nodes. Implies 10Hz per core, or 500 cores (8 nodes)

This paper compares CPU/GPU SZ compression: https://arxiv.org/abs/2007.09625. It states 370x improvement on GPU vs 1 CPU core. If we have 60 usable cores on a DRP node, that makes a GPU-SZ node the equivalent of 6x faster than a CPU-SZ node.

Schedule

(on Dec. 19, 2022) Sometime in February/March a prototype with 3 tiles will be released (could generate fake data for missing panels for 1 quad). Full detector: summer?

Calibration Time

Running this script on a psffb node (twice so we get all the data in cache so it behaves more like the daq). Do we also need to support det.image in the drp?

```
import time
from psana import *
ds = DataSource('exp=mfxx1005021:run=340')
det = Detector('epix10k2M')
for nevt,evt in enumerate(ds.events()):
    calib = det.calib(evt,cmpars=[0,0,0,0]) # disable common-mode with cmpars
    if nevt>=20: break
    if nevt==0: tstart=time.time() # start the timer after first event (which is slow)
print('time per evt:',(time.time()-tstart)/nevt,'nevt:',nevt)

(ana-4.0.48-py3) drp-srcf-eb004:lcls2$ python ~/junk.py
time per evt: 0.019545722007751464 nevt: 20
(ana-4.0.48-py3) drp-srcf-eb004:lcls2$ python ~/junk.py
time per evt: 0.019071054458618165 nevt: 20
```

So 20ms/core, corresponding to ~50Hz. For the record, with common mode (the default) the det.calib time increases to ~140ms, and det.image is ~170ms. That implies for 5kHz we need 100 cores, or ~2 nodes. Hopefully we don't need common-mode in the DRP. The algorithms Mikhail runs are listed here: Method det.calib algorithms#Detectordependentalgorithms

Comparing to a simple C++ calibration algorithm:

```
#include <stdint.h>
#include <stdio.h>
#include <string.h>

#define RAW_SIZE 2000000
#define NIMG 200
uint16_t raw[NIMG][RAW_SIZE];
float result[RAW_SIZE];
uint16_t peds[RAW_SIZE];
uint8_t mask[RAW_SIZE];
float gains[7][RAW_SIZE];

int main() {
    memset(mask,0,RAW_SIZE);

    for (unsigned count=0; count<NIMG; count++) {
        for (unsigned i=0; i<RAW_SIZE; i++) {
            unsigned val = raw[count][i];
            unsigned range = val&0x7000;
            result[i] = mask[i] ? 0 : ((val&0xfff)-peds[i])*gains[range][i];
        }
    }
}
```

With this we see about 7ms per image:

```
(ana-4.0.48-py3) drp-srcf-eb004:lcls2$ g++ -o junk junk.cc
(ana-4.0.48-py3) drp-srcf-eb004:lcls2$ time ./junk

real        0m1.432s
user        0m1.423s
sys        0m0.006s
```

Fiber Connections

A slack conversation with Dionisio and Matt:

Hi Dionisio, I believe TXI is getting an epixhr 2Mpx and we have to purchase daq equipment for it now.. Can you tell me how many fibers that detector will have? Thanks!

10:36 AM

That detector will use the 12 lane transceiver
10:36
and there will be 4 fiber bundles per 2M camera
10:36
one fiber per quad similar to the analog 10k camera

10:50 AM

I assume you mean “one fiber ***bundle*** per quad” above? And 12 fibers per bundle, so a total of 48?

10:51 AM

exactly

10:51 AM

10:55
Sorry, one more question: each fiber will be 6Gbps or something higher? (edited)

10:56 AM

48 fiber pairs, I assume.

10:57 AM

all lanes are bi-dir
10:57
so yes, 48 pairs

11:01 AM

6gbps?

11:02 AM

these are the fibers I am planning to use here for testing
image.png

11:03
most likely 12gbps

11:03 AM

We need single-mode.
11:03
Those look like multi-mode.

11:10 AM

these are the fibers that can connect to the transceiver. MM to SM conversion will be needed for this camera and we need to plan of which option we will use for that (same as it was done for ePix HR? or something else)

11:35 AM

Perhaps we discussed before and I forgot. Doing the MM to SM conversion for 96 fibers feels unsustainable.

11:43 AM

We discussed before. The transceiver on the detector end only exists for mm now.

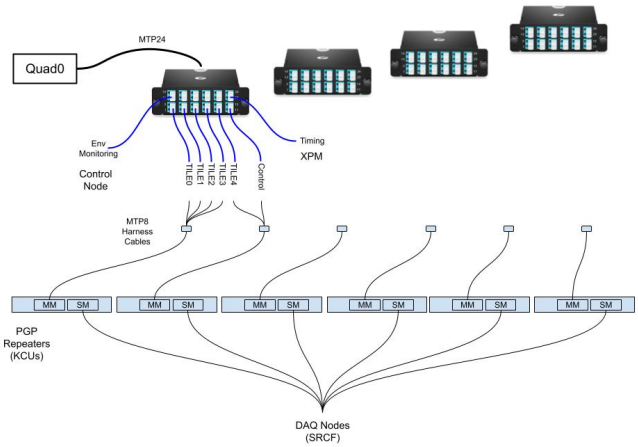
12:15 PM

Do I understand correctly we would need 48/8=6 kcu cards to to convert all of them? That will require a larger node.
12:18
I’m also worried because I had significant problems with lane-locking with the converter-card for the epix100. I had to mix-and-match lanes to find one that locked. Optical powers looked good. If we really need to do this conversion we should probably use that converter card to understand the lane locking problems. Maybe I needed to clean all the fibers somehow, even though the optical powers looked good?

Fiber Lane Assignments

Preliminary, from Dionisio

```
Lane[0].VC[0] = Data[0]
Lane[1].VC[0] = Data[1]
Lane[2].VC[0] = Data[2]
Lane[3].VC[0] = Data[3]
Lane[4].VC[0] = Data[4]
Lane[5].VC[0] = SRPv3
Lane[5].VC[1] = XVC
Lane[6].VC[3:0] = slow monitoring[3:0]
Lane[7].VC[3:0] = o-scope[3:0]
Lane[10:8] = Reserved for edgeML
Lane[11] = LCLS-II Timing
```



Can split the fibers onto different nodes.

Data is not currently in a natural order in the fiber, but plan on doing this in the tile electronics.

Calibration Issues Related to Data Reduction

Focusing on SAXS/WAXS in TXI for now, needed for data-reduction techniques: angular integration, cube, libpressio/libsz.

- offsets (use phil's max/min technique) run offline (at least at the start) perhaps on prescale events
    - ideally **needs work** (automation)
- pedestals (dark)
- gains (use charge injection or flat fields or design numbers)
- bad pixels
    - used dark runs for this in the past, but Mikhail is adding a new set of calibration constants so that other data (e.g. flat-field) can contribute to bad pixel masks
- geometry:
    - beam center (use silke's canny)
        - **needs some work,** e.g. integrating with LCLS2, fit-stability (e.g. with blobby rings)
        - Cong also has software for fitting silver-behenate rings (using LCLS1 data)
    - panel positions (epixHR2M on movable jaws).
        - **needs work**
        - not a flat detector
        - mikhail has software that projects onto a plane (will have to project the result back
        - do we have 3D metrology measurements for this detector? phil thinks yes (see below for response from Chris Kenney and Phil Hart about this). ideally would have metrology as a function of jaw separation.
        - we think TXI will have fuzzy silver behenate rings (see below for response from Andy Aquila about this)
            - Chuck can tell us about LCLS1 silver behenate data
        - could we use motor positions for the panel positions? didn't really
- timetool+delaystage
    - delaystage is relatively easy (e.g. gets changed in a step scan)
        - either a mechanical stage (uses speed of light) or an electronic delay stage which are also well understood
    - for the timetool camera edge-finding, want to convert the edge-position to picoseconds
        - this has been done routinely at XPP but not automated way
        - harder in XPP than CXI because you have to filter out bad events
            - filtered events: bad camera edge-fits, or width of the edge is too large, or amplitude of the edge, or the pulse-energy of the beam
        - ideally we would have a psana detector with calibration constants

            - the calibration constants exist for the opal. Ric has done some work porting the opal code to the piranha, but not complete.
    - **needs work** testing the calibration procedure and piranha edge-finding

From Chris Kenney about metrology measurements:

> Yes, we'll provide 3D metrology for each module including the Z positions. Each half should be mechanically stable internally.
>
> The two halves are moved with a motor which is specified to be accurate enough for the dynamic geometry. In addition there is a linear gage with fine accuracy that will provide a real time measurement of the gap.

From Phil Hart about metrology measurements:

> I don't know if this has been discussed and found impractical, or what, but I would think we might well be able to repeat what Kaz and I did with Georg's metrology group on the pnCCD to measure the relative x/y/(z?) positions of the two planes and check the motor encoder reporting using an optical telescope with the camera mounted in the chamber.

From Andy Aquila about silver behenate:

> This is correct Silver Behenate will be our go to for Z, centering and the two halves location. We will also use some lysozyme for a more accurate geometry during commissioning (this will be to determine the shingle locations).

epixhr2M Emulation

Brainstorming mtg on April 11, 2023: We are going to try to setup an as-complete-as-possible emulation of the epixhr2M in the real drp using the tdet simulated detector to bring together many complex moving parts.

existing epixhr panel data looked like this but only asic 4 worked:

1 2
3 4

new emulation data for a panel should look like this, and Ric will replace all asic data with asic 4 above (read from a file, can be same for every event for now, can get fancier in future):

1 2 3 4

issues:

```
- cpo to clarify with dionisio how many "configuration processes" we have?
  one per quad? one per detector? one per panel?
- mikhail's projection onto the plane
  o need simulated geometry
- need fake serial id numbers: test fetching of constants for a few panels
- multiple-segments per drp node.  one executable per segment?
  one executable for two segments? (prefer the latter).  or redefine
  a segment to be e.g. two panels (breaks our previous model of one
  segment being one piece of silicon)
  o need to event-build pgp which is a big change?
  o label each drp detector with one segment: epixhr_0. can we have
    two segments inside the dgram? (e.g. 0,1)
  o Ric points out that if we have multiple segments per drp exe then
    we could conceivably take advantage of that for things like veto
    computations. But we would have to distribute the panel data in
    an intelligent way.
- run drp-python
- run libsz compression
- run det.calib()
- eventually: veto using Ric's teb.  currently we have python-decision
  (parallelized for multiple nodes with multiple teb's).  but for the
  generation of the trigger data (e.g. "number of photons in my panels")
  feels like we should use drp-python. but this needs development.
- eventually: inject more realistic (simulated?) data.  use skopi for this?
```

EpixHR2x2

Fiber Lane Assignments

The fiber lane assignments for the 2x2 boards is

Lane[0] = Data
Lane[1] = Data
Lane[2] = ?
Lane[3] = Timing

ASIC Versions

The v2 ASIC requires descrambling of the data and the matrix readback does not work. The v4 ASIC should correct these issues.

Beam Test Trigger Setup

Detector tests run in the MFX beam line with hard x-ray beam from the LCLS NC accelerator. Thus, the 120 Hz beam is aligned with particular phases of the 60Hz AC power line, and the time between beam pulses varies with the power line sampling. In order to emulate high rate run triggers but still achieve alignment with the beam, an event code generating sequence is run in the XPM that restarts a train of triggers each 120 Hz cycle. For example, to simulate a trigger rate of 5 kHz, 41 triggers are generated with 200 us spacing every 120 Hz cycle (thus, 4920 kHz). The LCLS NC timing event codes provide enough time (~834us) to fire a few of these triggers before the beam arrives. The sequence would then generate new event codes at the following time intervals after receiving event code 40 from the accelerator:

event code	name	description	usec after EC 40
0	run trigger	used by detector to latch beam response	34, 234, 434, 634, 834, ..., 8034
1	daq trigger	used to readout detector	834 + subset of run trigger
2	parent trigger	used to trigger daq parent group	0 + all daq trigger
3	target	marks the daq event in-time with beam	834

The XPM is programmed using the seq_epixhr script. It's usage is:

> *(ps-4.6.3) bash-4.2$ seq_epixhr -h*
> *usage: seq_epixhr [-h] [--rate RATE] [--tgt TGT] [--daq DAQ] [--full] [--f360] [--minAC MINAC] [--pv PV] [--test] [--plot] [--verbose]*
>
> *Sequence for EpixHR DAQ/Run triggering*
>
> *optional arguments:*
> *-h, --help show this help message and exit*
> *--rate RATE Run trigger rate (Hz)*
> *--tgt TGT DAQ trigger time (sec)*
> *--daq DAQ DAQ trigger at RUN trigger rate/N*
> *--full DAQ trigger at RUN trigger rate (same as --daq 1)*
> *--f360 DAQ trigger at 360 Hz*
> *--minAC MINAC Minimum 120H interval in 119MHz clocks*
> *--pv PV XPM pv base*
> *--test Calculate only*
> *--plot Plot sequence*
> *--verbose Verbose*

The pv argument programs the sequence to a particular engine within the XPM, thus assuming the event codes that appear. Lastly, the a readout group must be driven by the run trigger event code. Since the DAQ makes no use of this readout group, it has to be done manually with groupca.

Running devGui

From Alex Batyuk:

```
cd /cds/home/w/weaver/epix-hr-new/software
source ./setup_l2si.sh
python scripts/ePixHr10kTDaqLCLSII.py --start_viewer True --start_gui True
```