AstroPix BIC wafer/module/stave meeting, May 11 (2026) / C. Kim (PNU)

- **AstroPix-v3 study**
  - Adapter card production
    a. Reminder: interconnection card between the Gecco board and the probe card
    b. Recent modification: power module has been added to the design enabling I-V scan
    c. Status: Dr. Kim of NOTICE currently design the circuit (expect design completion next week);
    * Expected delivery date by May 20
    d. Test plan (tentative):
    d-1. (FPGA + Gecco) ↔ adapter card ↔ probe card test at NOTICE (late May)
    d-2. Onsite test with full apparatus at C-ON tech (late May to early June)
  - QA framework development
    a. Repository: https://github.com/Chongk/astropix_v3_QA
    b. Purpose: mass chip QA framework for existing v3 chip, considering migration to the future v5 chip
    c. Structure:
    b-1. Uses A-STEP firmware and data format
    b-2. Python-based control/DAQ/QA and C++/ROOT-based decoding and analysis
    d. DAQ and readout: manual IRQ-owned readout in software level

QA framework Structure (1/3)

- **Python-based control/DAQ/QA**
  - bootstrap.py: opens the board connection from XML settings and loads YAML chip configurations
  - config.py: provides the *V3Config* object (single source of truth for v3 chip state: e.g., enable pixel, etc.)
  - protocol.py: converts configuration state into A-STEP-compatible SPI bitstreams and frames
  - transport.py: hardware adapter over the existing A-STEP board driver –
    read firmware ID, configure clocks, arm/disarm readout, SPI select/deselect, read buffer, etc.
  - controller.py: chip control on top of transport and protocol (e.g., chip programming, threshold/injection, etc.)
  - daq.py: manual IRQ-owned DAQ (details later)
  - threshold.py: threshold control (internal threshold path or external Gecco voltage-card path)
  - qa.py: QA modules (* enforces the manual IRQ-owned readout),
    currently prepared for smoke (overall DAQ chain sanity), 5 pixels injection,
    and threshold scan vs. noise
  - v3_qa_run.py: QA runner script

QA framework Structure (2/3)

- **C++/ROOT-based decoding and analysis**
  - decode.h / decode.cpp:
    a. C++ decoder for A-STEP binary output
    b. Parses raw binary (*.bin) files into V3Hit objects
    (i.e., header, chip id, row/col, timestamp, tot, and FPGA_timestamp)
    c. The decoding statistics also accessible
    (i.e., decoded hit count, skipped bytes, total raw bytes, and malformed fraction)
    d. Currently does NOT generate human-readable output (e.g., txt or csv) to save disk space, but it can be added easily if necessary
  - qa_ana.cpp: ROOT-based QA analysis modules

QA framework Structure (3/3)

- **Workflow of current framework**
  - Initialization (bootstrap):
    a. Read XML setting and open the board
    b. Load YAML configuration file(s)
    c. Build runtime objects
  - Configuration and programming (config, protocol, and controller):
    a. Stores the intended chip state (*V3Config*)
    b. Converts the state into routing and SR frames (*V3Protocol*)
    c. Applies the state to the board/chip (*V3Controllder*)
  - DAQ (daq):
    prepares the run, flushes stale data, arms readout, waits for IRQ, and performs manual burst acquisition (*V3DAQ*)
  - QA execution (qa): perform QA routines (*V3QA*)
  - Output storage: stores raw binary data (if produced), run summary, and readout-index info
  - Post analysis: decode raw binary data and perform ROOT-based analysis

QA framework Dealing with A-STEP data (1/2)

- **Decoding A-STEP based binary**
  - A-STEP data format: 11 bytes per packet
    a. Mapping (by following order):
       a-1. A-STEP header (1 byte)
       a-2. Layer ID (1 byte)
       a-3. AstroPix-v3 hit payload (5 bytes): hit header, row/col, timestamp, ToT MSB/LSB
       a-4. FPGA timestamp (4 bytes)
    b. A valid A-STEP header:
       b-1. The decoder scans byte stream and accept 11 bytes only when a location start with:
            byte “i” is 0x0A, and byte “i+1” is 0x01, 0x02, or 0x03
       b-2. Otherwise, it advances by 1 byte and counts such bytes as skipped (or trailing)
  - A-STEP data format does NOT record readout index
    a. QA framework writes a separate supplementary JSON file (sidecar) maps each DAQ burst to a byte range
    b. In other words,
       b-1. Raw binary remains untouched
       b-2. Readout index (DAQ burst index, NOT event) is stored independently
       b-3. Merging happens only at decode/analysis

QA framework Dealing with A-STEP data (2/2)

- **Quick crosscheck with official decoder**

offset	readout	layer	chip	payload	location	isCol	ts	tot	fpga_ts
12	0	1	0	4	5	0	108	673	1814228666
23	0	1	0	4	6	0	230	1571	1814228666
34	0	1	0	4	7	0	84	1879	1814228666
45	0	1	0	4	8	0	183	2200	1814228666
56	0	1	0	4	10	0	220	844	1814228666
67	0	1	0	4	14	0	56	1759	1814228666
78	0	1	0	4	16	0	78	2378	1814228666
89	0	1	0	4	19	0	85	458	1814228666
100	0	1	0	4	23	0	173	2727	1814228666
111	0	1	0	4	25	0	220	1212	1814228666
122	0	1	0	4	27	0	32	1423	1814228666
133	0	1	0	4	28	0	255	139	1814228666
144	0	1	0	4	29	0	29	1999	1814228666
155	0	1	0	4	6	0	45	840	1814228666
166	0	1	0	4	8	0	6	1	1814228666
177	0	1	0	4	14	0	60	3577	1814228666
188	0	1	0	4	16	0	179	3417	1814228666
199	0	1	0	4	18	0	67	1150	1814228666
210	0	1	0	4	19	0	202	359	1814228666
221	0	1	0	4	23	0	143	1810	1814228666

```
dec_ord,readout,layer,chipID,payload,location,isCol,timestamp,tot_m
0,0,1,0,4,5,0,108,2,161,673,6.73,1814228666
0,0,1,0,4,6,0,230,6,35,1571,15.71,1814228666
0,0,1,0,4,7,0,84,7,87,1879,18.79,1814228666
0,0,1,0,4,8,0,183,8,152,2200,22.0,1814228666
0,0,1,0,4,10,0,220,3,76,844,8.44,1814228666
0,0,1,0,4,14,0,56,6,223,1759,17.59,1814228666
0,0,1,0,4,16,0,78,9,74,2378,23.78,1814228666
0,0,1,0,4,19,0,85,1,202,458,4.58,1814228666
0,0,1,0,4,23,0,173,10,167,2727,27.27,1814228666
0,0,1,0,4,25,0,220,4,188,1212,12.12,1814228666
0,0,1,0,4,27,0,32,5,143,1423,14.23,1814228666
0,0,1,0,4,28,0,255,0,139,139,1.39,1814228666
0,0,1,0,4,29,0,29,7,207,1999,19.99,1814228666
0,0,1,0,4,6,0,45,3,72,840,8.4,1814228666
0,0,1,0,4,8,0,6,0,1,1,0.01,1814228666
0,0,1,0,4,14,0,60,13,249,3577,35.77,1814228666
0,0,1,0,4,16,0,179,13,89,3417,34.17,1814228666
0,0,1,0,4,18,0,67,4,126,1150,11.5,1814228666
0,0,1,0,4,19,0,202,1,103,359,3.59,1814228666
0,0,1,0,4,23,0,143,7,18,1810,18.1,1814228666
```

offset	readout	layer	chip	payload	location	isCol	ts	tot	fpga_ts
0	0	1	0	4	10	0	140	3473	3659606485
11	0	1	0	4	4	1	140	3452	3995150805
22	1	1	0	4	20	0	85	2704	1467051989
33	1	1	0	4	7	1	85	2687	1802596309
44	2	1	0	4	12	0	21	3430	3434270933
55	2	1	0	4	6	1	21	3412	3769815253
66	3	1	0	4	31	0	181	2305	1519755734
77	3	1	0	4	30	1	181	2284	1855300054

```
dec_ord,readout,layer,chipID,payload,location,isCol,timestamp,tot_
0,0,1,0,4,10,0,140,13,145,3473,34.73,3659606485
0,0,1,0,4,4,1,140,13,124,3452,34.52,3995150805
0,0,1,0,4,20,0,85,10,144,2704,27.04,1467051989
0,0,1,0,4,7,1,85,10,127,2687,26.87,1802596309
0,0,1,0,4,12,0,21,13,102,3430,34.3,3434270933
0,0,1,0,4,6,1,21,13,84,3412,34.12,3769815253
0,0,1,0,4,31,0,181,9,1,2305,23.05,1519755734
0,0,1,0,4,30,1,181,8,236,2284,22.84,1855300054
```

QA framework DAQ and readout behavior (1/3)

- **AstroPix-v3 data acquisition**
  - Simplified data acquisition cycle **(chip)**

    * WARNING: this is a software level implementation being used in the current QA framework –

    i.e., Python controls readout sequence and FPGA/board API executes it

    a. The chip asserts IRQ (interrupt requested) when data is available

    b. CSN (chip select not) is asserted low by the reader

    c. Dummy bytes are sent to provide SPI clock

    d. Data burst is drained while valid data continue to arrive

    e. The reader can enter tail-drain phase even after the IRQ deasserts

    f. The burst ends when:

    f-1. IRQ is no longer asserted, and

    f-2. A configured # of idle rounds reached

    g. CSN is deasserted high,

    at the end of the software burst

FE <-> APS Readout Sequence Overview

- Timestamp Clock and ToT clocks active
- Communication using SPI
- Send Routing Byte through DaisyChain
- Send Shift Register Config Command to configure Pixel Matrixes
- Wait for Interrupt with SPI deactivated (Digital Block Off)
- When Interrupt is asserted, Send dummy bytes to start readout
  - Keep sending dummy bytes until the interrupt is deasserted and no packets are detected for c.a 40 Bytes
- Deactivate SPI to 'sleep'

v3 readout sequence design (quote from v3 spec sheet)

QA framework DAQ and readout behavior (2/3)

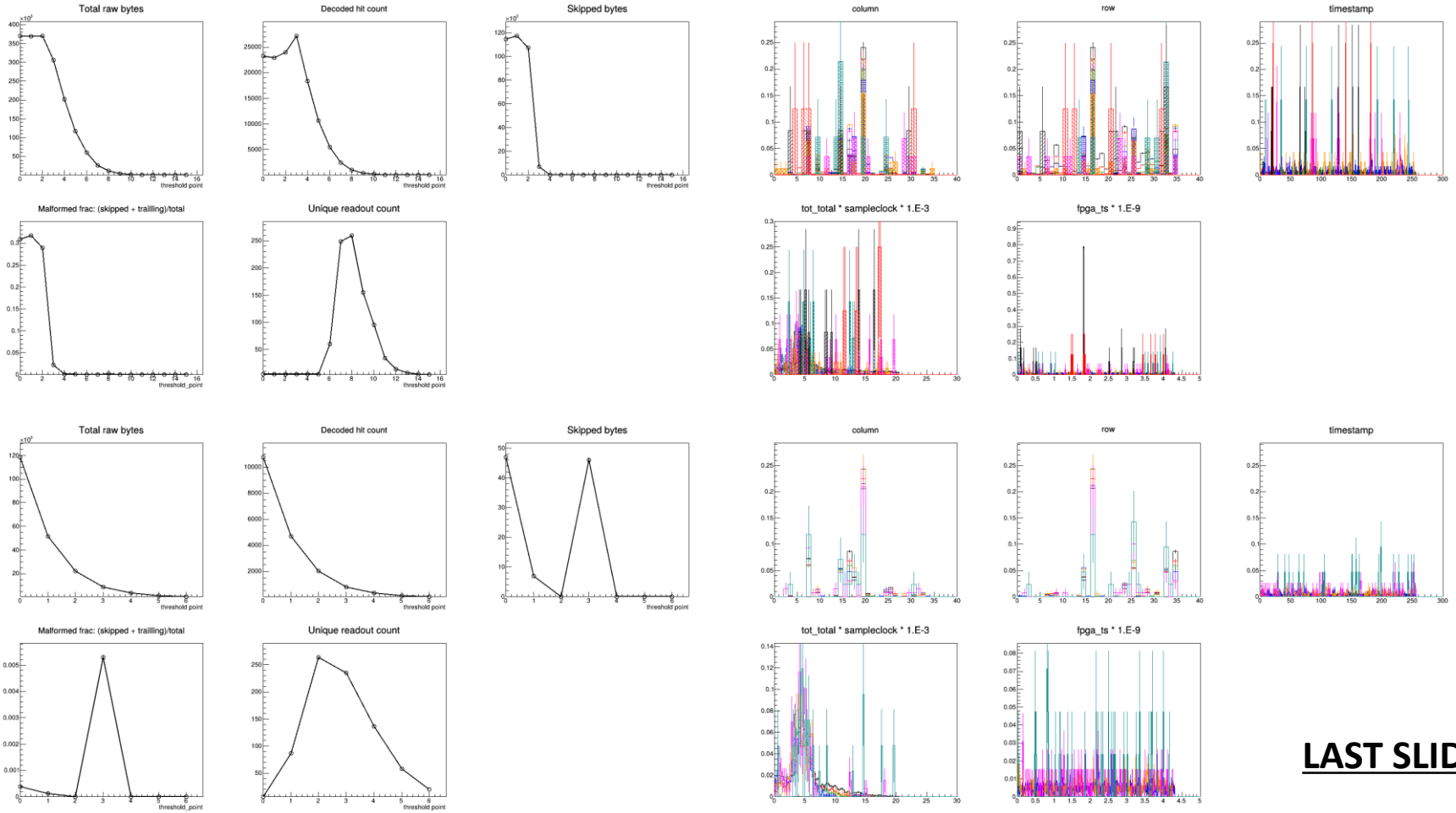
- **Data acquisition sequence of current QA framework**
  - Entire sequence (including previous chip level behavior):
    a. Reset and program the chip(s)
    b. Flush stale data
    c. Arm readout with `autoread=False`
    d. Poll IRQ in software
    e. When IRQ is seen, assert CS
    f. Send dummy bytes and read the FPGA buffer
    g. Continue until the software termination condition is met
    h. Drain any remaining FPGA buffer content
    i. Deassert CS
    j. Store the acquired bytes as one DAQ chunk with metadata such as:
       # of rounds, bytes written as dummy, if the readout frame is truncated or stopped
  - WARNING: the QA framework does NOT rely on firmware autoread (which is used in official codes):

    this makes the framework is slower and inefficient than a pure firmware-owned control,

    but it is more transparent for QA since the burst process is visible and analyzable at byte level

QA framework DAQ and readout behavior (3/3)

- **Example: threshold scan vs. noise @ bias -150V**
  - 1st scan: [150, 300]; 2nd scan: [200, 260]; Stepwidth of 10 mV



LAST SLIDE