A "FELIX-lite" Streaming DAQ Readout

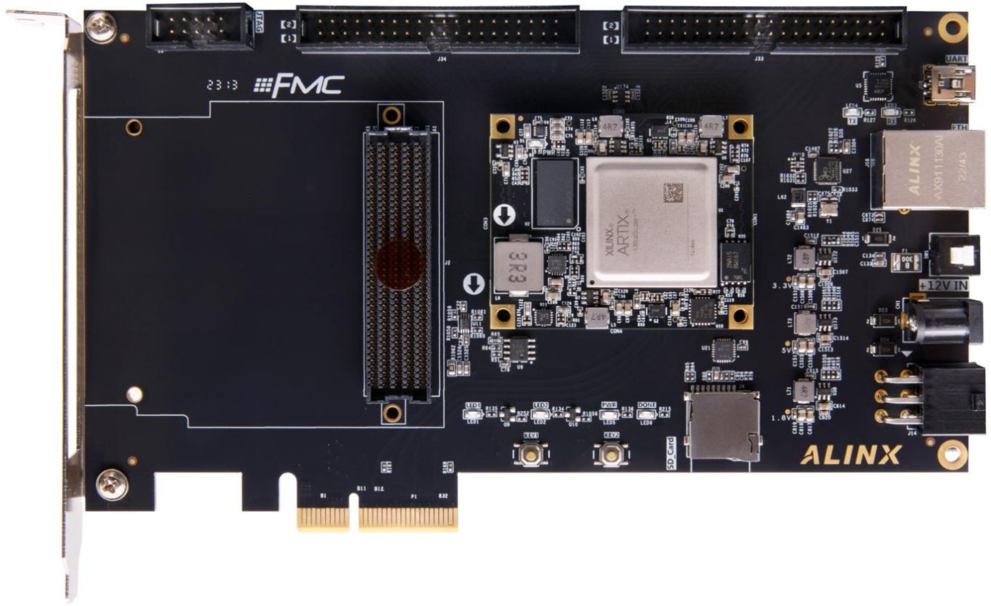
For Simple and Cheap
Streaming Readout Tests,
RDO Testing &
ASIC Testing

Tonko Ljubicic, Rice University
15 May, 2025

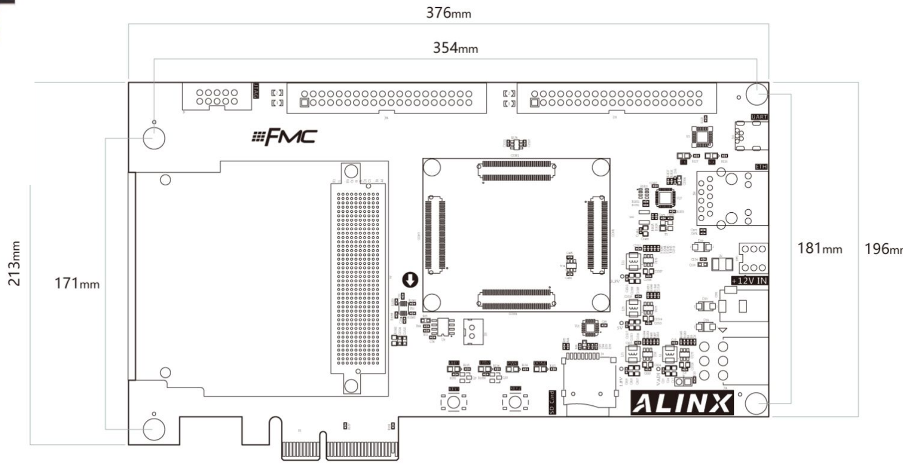
Guiding Principles/Requirements

- Create a small, cheap and compact DAQ system which includes full streaming primitives
  - It's a **"DAM-lite"** board, instead of the much bigger, cumbersome, FELIX board
  - FTOF is its first client, BTOF will become its second while other AC-LGAD or lpGBT based detectors could also use it
    - but others too…
  - we chose the Alinx AXAU15 commercial development kit as our **"DAM-Lite"** board
    - Xilinx Artix Ultrascale+ FPGA (xcau15p)
      - Xilinx AXI4 (ARM) based internal bus
      - Xilinx microBlaze CPU soft core for testing and debugging via a USB console port
        - very quick development cycle, done in "C", with the board on the bench (no need for a PCIe host)
    - FMC connector for extension cards
      - fiber/SFP module if we want to connect to e.g. lpGBT or an "RDO"
        - we like the **Trenz TEF0008 FMC SFP+ carrier** [used in STAR]
        - 4 available fibers (but hear William's ideas for a custom fiber interface!)
      - direct connection to ASIC Test Boards via their FMC interface
        - hopefully starting with EICROC1
    - PCIe x4 connector for actual streaming data once installed in a host computer
- Firmware development cycle must be quick and painless
  - heavy use of GHDL (a free VHDL simulator running under Linux/Win) dramatically quickens the development cycle (from days to hours)
  - GHDL also avoids constant use of Xilinx Vivado tools with all their bugs and slowness

Alinx AXAU15



- **small, compact**
- Artix xcau15p FPGA (& associated PROM)
- FMC HPC (for fiber or direct ASIC Test Board)
- PCIe x4
- Mini USB UART Console (connected to the soft CPU core in the FPGA)
- Gb Ethernet (unused, anyone interested?)
- JTAG port
- 40 pin Expansion Ports (I only use them for debugging with a scope but TBD)
- External Power Supply connector (when on the bench)
- PC Power Supply connector (when installed in the computer)
- Micro SD slot (unused)



Top View

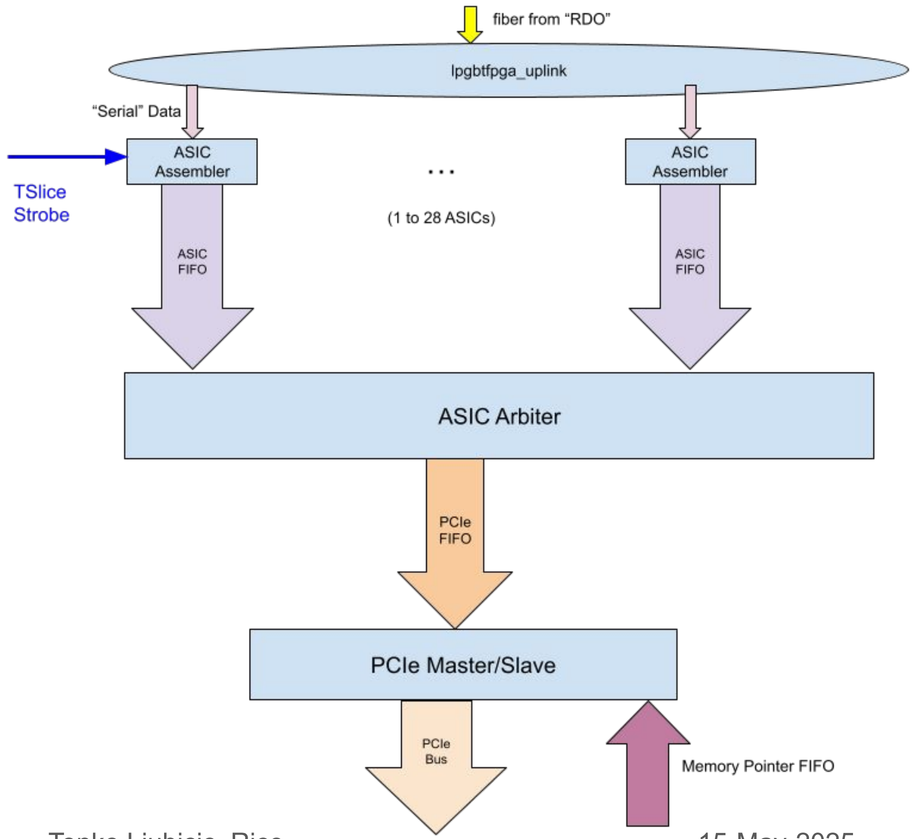
Hardware Steps (Related to AC-LGAD Detectors)

1. Obtain an AXAU15 with fiber FMC and SFP+ ✅
    a. encode the framework firmware ✅
    b. encode the fiber interface (GTH) ✅
        i. currently 1 fiber only but can be extended to all 4
        ii. coded for lpGBT 10.22/FEC5 using the “lpgbtfpga” package from CERN
    c. encode the PCIe master ✅
        i. currently PCIe x4 @ 5 Gbs
    d. use a realistic ASIC simulator to check the data path with streaming ✅
        i. 1-28 ASICs/fiber; each ASIC generates 30 kHz of random noise and generates hits corresponding to 500 kHz collision rate ✅
        ii. numbers come from TOF tests
    e. implement streaming using ~1ms long timeslices with emulated GTU interface ✅
2. Connect it to the CMS ETL Readout Board at Rice (which is very similar to FTOF RBv1!) [1-2 weeks]
    a. encode lpGBT register reads/writes
3. Connect the CMS ETL ETROC ASIC Module to the CMS ETL Readout Board at Rice [2-3 weeks]
    a. encode I2C ASIC configuration protocols
4. Do a readout of ETROC (we don’t care about the data) using an internal trigger (ETROC is not a streaming chip but we’ll make it look like one) [4-6 weeks]
5. Once the FTOF Readout Board RBv1 arrives (depends on PED funds, hoped for ~Jul 2025) repeat 2, 3 & 4
    a. also using our Power Board PBv1 (also depends on PED funds…)
    b. this vets the RBv1, PBv1 and the Readout Chain
6. Once we can get our hands on an EICROC1 Test Board we can connect it to our RBv1 and read it out (Fall 2025?)
7. Eventually connect the EICROC1 to an AC-LGAD sensor and take it all for a full spin (end of 2025?)

Software Steps (in parallel to hardware)

- Linux device driver for streaming timeslice buffers ✅
- Control/Status device driver for the AXAU15 registers ✅
- lpGBT & ASIC Configuration [waiting for hardware…]
- DAQ threads for readout □
- Data Coherency checkers □
- Mini Event Builder to assemble data from multiple fibers based upon the GTU timestamp ☑️
- File formation using "A Common Data Format" [waiting for it…]
  - in the meantime use STAR's SFS Data Format and DAQ Readers
- Logging (ERROR, NOTICE, etc) □
- Monitoring (HTTP and Websockets) [thinking about it, setting up…]
- Online cluster-finding (TOF-specific) or other online analysis procedures to make the data volume smaller and/or remove the noise [thinking about it…]
- ⇒ generally, make the system usable for non-expert lab testing
  - while also learning how to control it all

Data scheme (same for each fiber)



- lpgbtfpga_uplink unpacks the raw fiber format into 28x8 bit partially serialized chunks of ASIC words
    - I am connected to an lpGBT on the far end thus *"lpgbtfpga"* but it can be any custom RDO protocol
- ASIC Assembler assembles the chunks into 32bit ASIC words
    - it knows about the IDLE words *and suppresses* them
    - it accepts the TSlice Strobe pulse from the GTU and acts accordingly (see box below)
    - the ASIC Packet (header,data,trailer) is sent to the ASIC FIFO
    - ASIC FIFO must be large enough to accept an entire "black" event worth of channels (1026 words for FTOF)
- ASIC Arbiter decides which ASIC FIFO to read
    - fair round-robin algorithm; fast
    - ASIC FIFO is read in complete ASIC Packets and the packets are sent to the PCIe FIFO
        - note that it maintains the data packet boundaries
- PCIe Master reads the PCIe FIFO and strobes the data in 16-word bursts into the computer's preallocated memory
    - Memory Pointer FIFO contains the physical memory address of available buffers in the computer's memory
        - typically 64-256 such buffers, typically 1 MB long
        - filled by the DAQ threads running in the PC
    - Timeslice data starts with a 16 word Header, followed by ASIC data and ends with a 16 word Trailer (more below)

- Upon receipt of the TSlice Strobe the ASIC Assembler keeps pushing data into the ASIC FIFO until an ASIC Trailer is seen → this is to make sure we don't break up the ASIC Packet between timeslices
- After which it writes a Sentinel Word to the ASIC FIFO which informs the ASIC Arbiter that a timeslice is complete for this ASIC
- The ASIC Arbiter temporarily suspends reading from an ASIC which sent the Sentinel and waits until <u>all</u> ASICs sent their Sentinels thus completing the timeslice.
- 16 word Trailer is sent over the PCIe bus
- 16 word Final Header is sent over the PCIe bus

Timeslice Raw Data Format

Timeslice Header (16x 32bit words)

0	0x**ABCD**0001	0x0001 is the Format Version
1	Geo Id (Det:PC:DAM:fiber)	e.g. 0x0601011F
2	Words in this Timeslice	
3	Status/Error	0 is all OK
4	GTU Timeslice Xing (lo)	
5	GTU Timeslice Xing (hi)	
6	DAM Xing (lo)	
7	DAM Xing (hi)	
8	Timeslice count	Since Run Start
9	Event Type	Assists the Data Unpacker
10-14	Reserved/Debugging	
15	0x**FEED**0001	0x0001 is the Format Version

- **Trailer** is similar but with more reserved/debugging words
- Format Version (currently set to 0x0001) *determines the format of the Header and the Trailer*, not the data itself
- Geo Id determines which board this is, unique to the entire DAQ
  - 8 bits for the Detector Id ("6" for FTOF?)
  - 8 bits for the PC of this detector (1 to many, e.g. 4 for FTOF)
  - 8 bits for the DAM board in this PC (typically just "1")
  - 8 bits for the fiber of this DAM board (1 to 4 for AXAU15, 1 to xx for FELIX)
- **GTU Timeslice Xing** (or "Command") is the main datum which is used to assemble timeslices over the entire EPIC DAQ
  - it is sent to all EPIC DAMs at the same time, with the same value
  - BTW, I assumed here that the GTU Command is 64 bits
- DAM Xing also counts at the same clock as the GTU ⇒ it is used to compare the delta with the GTU as the run progresses to make sure we didn't skip a clock
- Event Type assists the Data Unpacker tools and provides guidance on the specific style of the ASIC data or flavor

Final Thoughts

- I like this board. I’ve been working with it for ~month and it looks solid. Quiet fan. Precisely done. ⭐⭐⭐⭐⭐
- Artix 15 is not that large, not that much FIFO memory available
  - 4 fibers **AND** 28 ASICs on 1 fiber **AND** 1024 channels in each ASICs will not be possible due to lack of memory
    - either less fibers, less ASICs or smaller ASICs…
  - technical limit is the number of available BRAM for the ASIC FIFOs
- PCIe Core is large and takes Vivado a looong time to synthesise and elaborate → impractical during fast debugging/commissioning turnarounds
  - which is why I have 2 flavors of the firmware: *with* and *without* PCIe
    - they are exactly the same except the “nopcie” has the entire “PCI Express Bridge” simply omitted
  - Without PCIe, the soft-core CPU can do exactly the same over USB albeit much, much slower
    - still, it can readout about 32 kB of data before the FIFOs overfill → enough for TOF applications or low data rate readout
    - E.g. I use it to commission lpGBT interaction & configuration
      - also ASIC I2C configuration and data format commissioning
- Final Note: William et al (JLab) are planning on making an FMC card compatible with AXAU15 which will have 4 data fibers <u>and</u> the <u>complete</u> GTU⇔DAM data path and protocols
  - Great idea! I would like to get one ASAP :-)