Trigger Primitive Generation - Firmware

Konstantinos Manolopoulos

DUNE Upstream DAQ – Readout Technology Evaluation

July 29th 2020

Science & Technology Facilities Council
Rutherford Appleton Laboratory

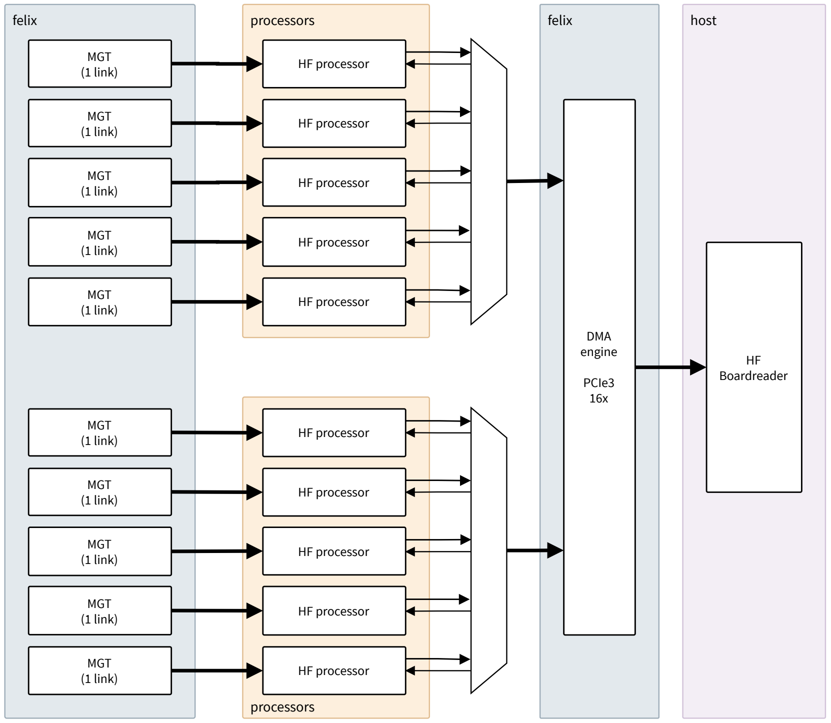
Outline

- Processing framework & Hit Finder algorithm
- Development platforms and Testing strategy
- Software emulation and Validation
- FELIX integration
- ProtoDUNE demonstration & Results
- Future work
- Summary
- Criteria analysis

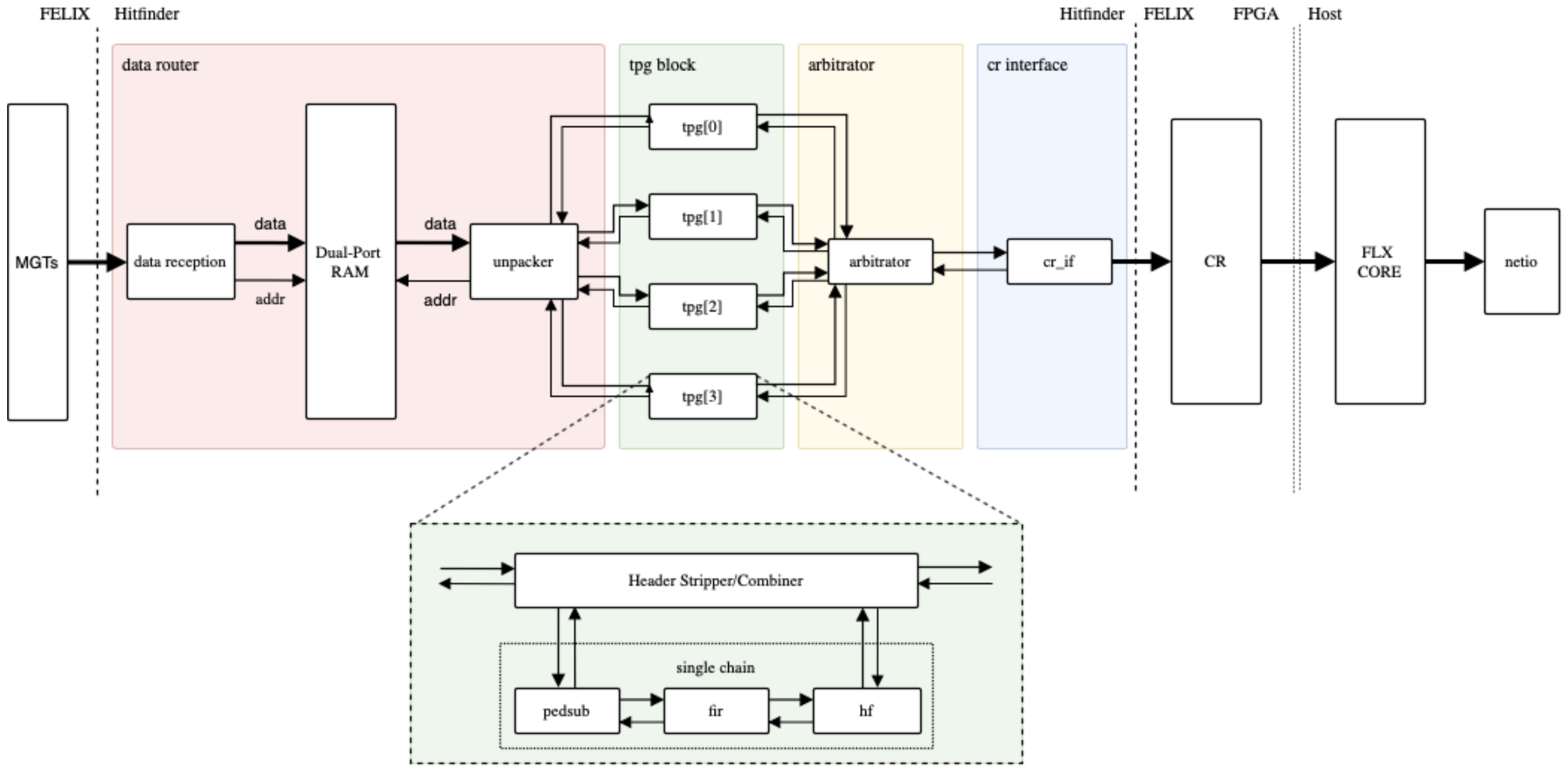
Processing framework & Hit Finder algorithm

Reminder

- 10 fibres per APA @9.6 Gbps
  - 12-bit ADC samples @2MSamples/s
  - Effectively 10 independent data streams
- Each fibre corresponds to 256 wires
  - Induction & collection wires on the same fibre
- 10 "Hit Finder" processors operating in parallel (1 per fibre) @250MHz
- Each Hit Finder produces hits also by processing ADC values in parallel

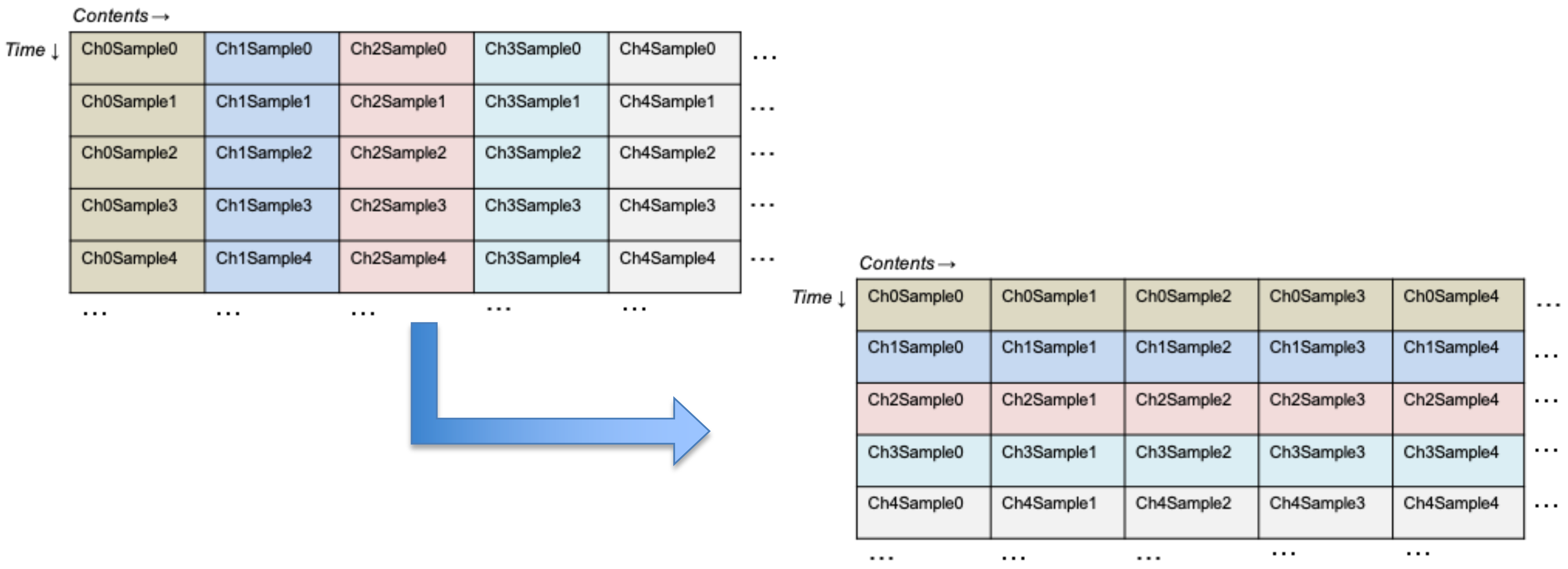


Hit Finder Firmware Architecture



Data Router

- Consists of 3 major blocks: **Data Reception, Data Storage** and **Unpacker**
- **Data Router** turns WIB packets (multiple channels, single time-tick) to processing packets (single channel, multiple time-ticks)
  - Data arrives in packets containing all channels for a single time-tick
  - Rearrange into packets with multiple time ticks for a single channel



Contents →

Time ↓						
	Ch0Sample0	Ch1Sample0	Ch2Sample0	Ch3Sample0	Ch4Sample0	…
	Ch0Sample1	Ch1Sample1	Ch2Sample1	Ch3Sample1	Ch4Sample1	…
	Ch0Sample2	Ch1Sample2	Ch2Sample2	Ch3Sample2	Ch4Sample2	…
	Ch0Sample3	Ch1Sample3	Ch2Sample3	Ch3Sample3	Ch4Sample3	…
	Ch0Sample4	Ch1Sample4	Ch2Sample4	Ch3Sample4	Ch4Sample4	…
	…	…	…	…	…	

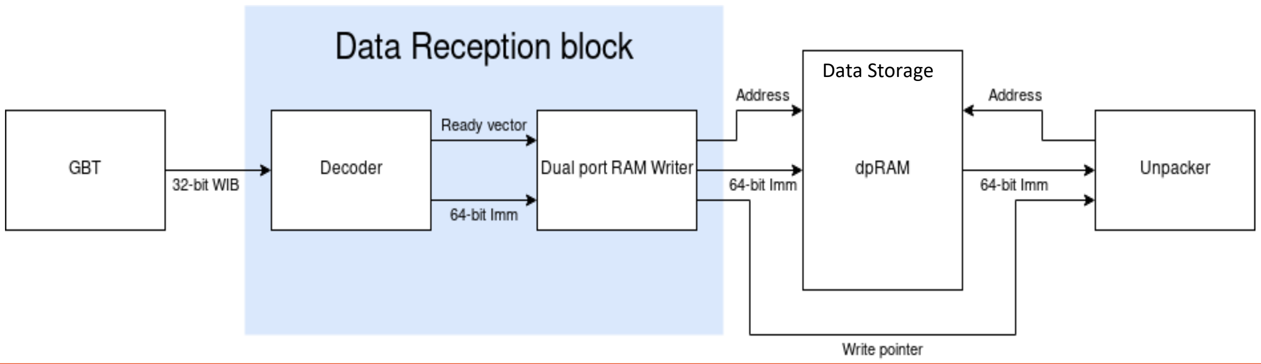
Contents →

Time ↓						
	Ch0Sample0	Ch0Sample1	Ch0Sample2	Ch0Sample3	Ch0Sample4	…
	Ch1Sample0	Ch1Sample1	Ch1Sample2	Ch1Sample3	Ch1Sample4	…
	Ch2Sample0	Ch2Sample1	Ch2Sample2	Ch2Sample3	Ch2Sample4	…
	Ch3Sample0	Ch3Sample1	Ch3Sample2	Ch3Sample3	Ch3Sample4	…
	Ch4Sample0	Ch4Sample1	Ch4Sample2	Ch4Sample3	Ch4Sample4	…
	…	…	…	…	…	

Data Router

- **Data Reception block** receives data from the GBTs and converts incoming packets from 32-bit WIB to Nx16-bit (N=4) format
  - Converted packet written to a circular buffer (**Data Storage** - dpRAM block implementation)
  - Recognizes & exploits repeating patterns in incoming data, allowing smarter buffering of smaller parts of the frame

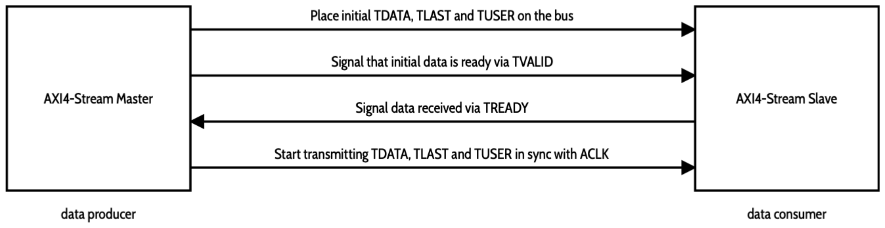
- **Unpacker:** unpacks channel data from Data Storage (64-bit dpRAM), repacks them N=4 channels at a time, and sends the packets to the TPG blocks over axi4s bus



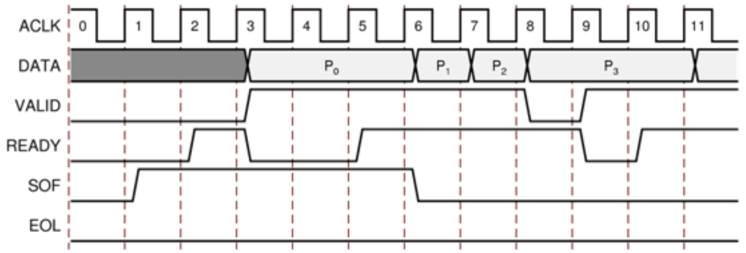
Data Format used in TPG blocks

- ADC samples are processed in packets
  - Each packet consists of 64 ADC samples (data frame) and a Header that contains a time-stamp, channel no. and flags (Header frame)

- AXI4-Stream protocol after unpacking & reordering of data
- AXI4-Stream: AMBA protocol designed to transport arbitrary unidirectional data streams
  - tvalid denotes valid data on the bus
  - *tuser* flag denotes end of each frame,
  - *tlast* denotes end of packet
  - *tready* used to apply backpressure

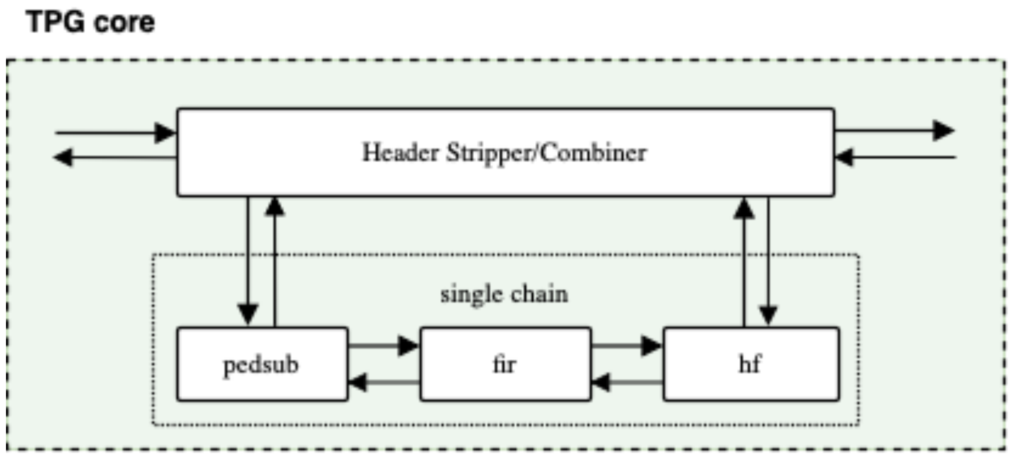

Place initial TDATA, TLAST and TUSER on the bus
Signal that initial data is ready via TVALID
Signal data received via TREADY
Start transmitting TDATA, TLAST and TUSER in sync with ACLK
AXI4-Stream Master
AXI4-Stream Slave
data producer
data consumer

AXI4-Stream handshake



TPG block - Header Stripper/Combiner

- Receives packets from Data Router
  - Upon arrival of a packet signal DR to hold transmission
- Strips Header and sends it to the output
- Sends the 64 ADC values to the processing chain (PedSub, FIR, HF)
- Receive hits from Hit Finder and forwards them to the output
- Signals Data Router to send next packet



TPG block - Pedestal Subtraction

- Firmware implementation based on Phil's algorithm for Pedestal Subtraction

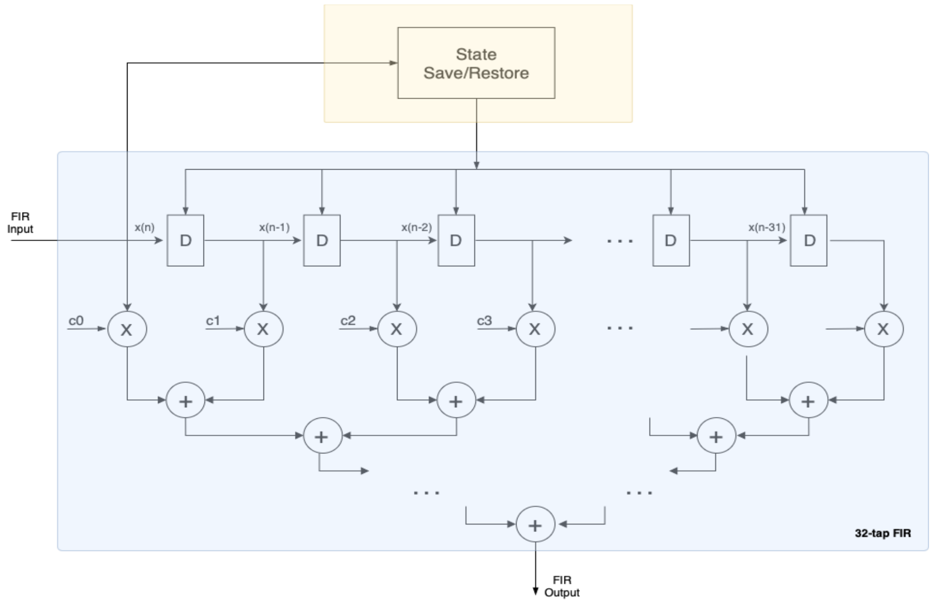
- Algorithm:
  - Start with *accumulator=0* and an estimate of the *pedestal*
  - If *ADCvalue* >(<)*pedestal* => *accumulator* +(-) 1
  - If *accumulator* = +(-) 10 => *pedestal* +(-) 1 and reset *accumulator* to 0

- State Save/Restore mechanism
  - At the end of each packet store the pedestal & accumulator values and retrieve them when it's time to process a packet from that same channel again.
  - Use of distributed RAM to save BRAM resources

- *For validation purposes we currently also send to the output the pedestal and accumulator values at the beginning of each data packet, if that packet produces a hit.*

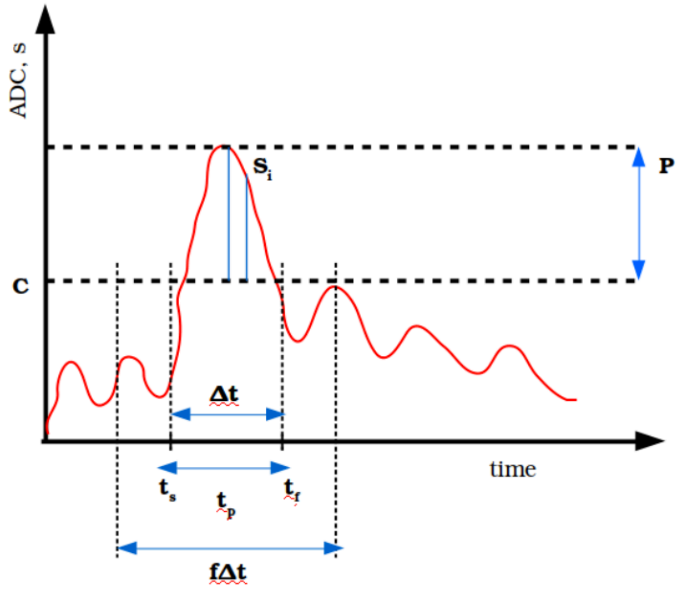
TPG block – Filtering

- Simple low-pass, 32-tap FIR filter
  - Custom FIR implementation
- State Save/Restore mechanism
  - Store the last 32 ADC values of an incoming packet for each channel
  - Retrieve the values and preload them into the FIR when it's time to process a new packet from that same channel
- Coefficient set currently hardwired
  - Can easily be changed to accept different sets of coefficients



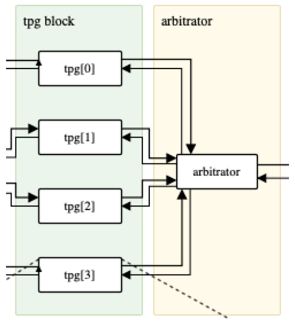
TPG block - Hit Finder

- Processes pedestal-subtracted & filtered ADC values
- A hit starts when 2 consecutive ADC values (s_i) are above threshold and lasts until the values drop below threshold
- Quantities output by Hit Finder:
  - Start/End time of a hit
  - Peak time, peak amplitude
  - Sum of all ADC values above threshold
  - Detects if a hit expands over end of packet and raises a Hit Continue flag
  - Hit Finder is memory-less (no State S/R)



- Threshold can be changed via IPBus

Arbitrator & CR-interface

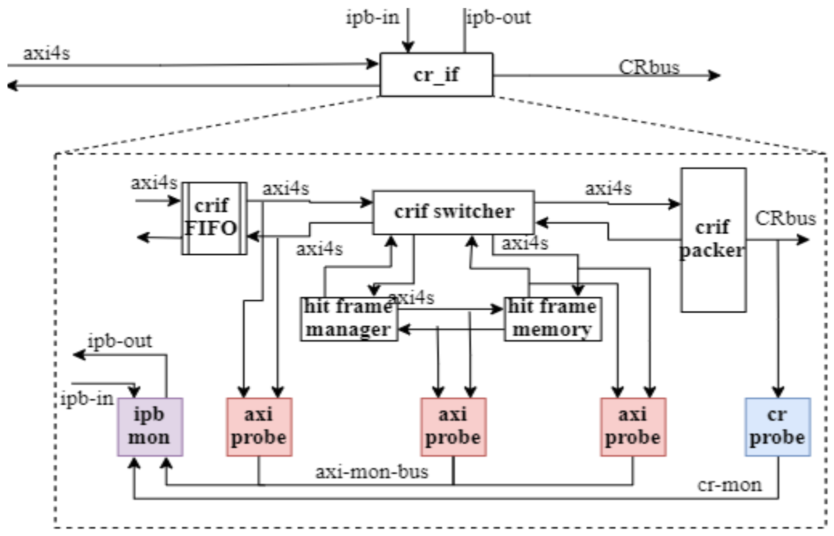


Arbitrator: multiplexes the 4 TPG blocks into a single output

- FIFOs used to buffer inputs in case of multiple incoming packets
- Waits for end-of-packet signal to output data

CR-interface: used as interface between Hit Finder and Central Router

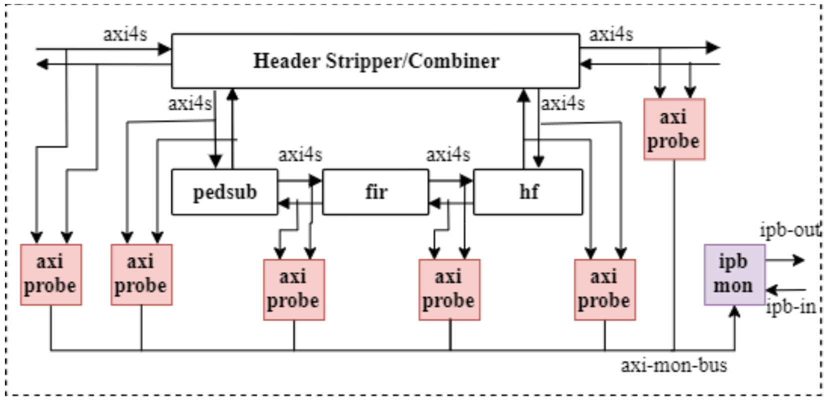
- Consists of 3 main blocks:
  - FIFO wrapper: prevents backpressure events
  - Hit Packet Filter: filters out hit empty packets, detects & removes corrupted hit packets
  - CRIF Packer: is the CR-if core and translates each hit packet into a CR packet format

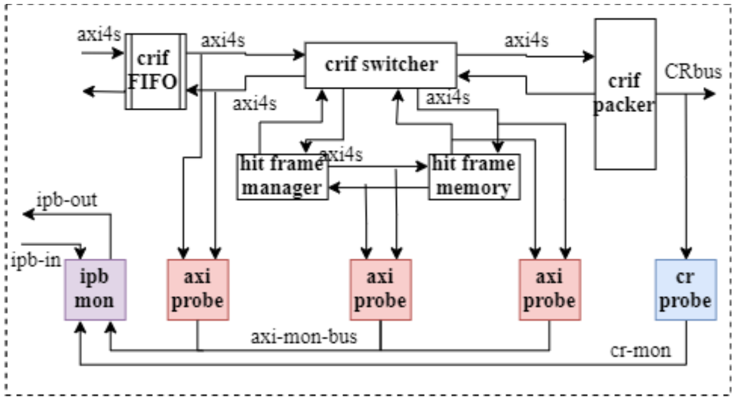


Development Platforms & Testing strategy

Development Platforms

- ZCU102 eval. boards used for development (ZU9EG-2)
  - Testing sites at Bristol, RAL & Sussex
    ➔ allowed parallel testing of different parts of HF chain
  - Use of Source & Sink buffers to inject and capture data
  - Use of monitor probes plugged at the interface between blocks
    - Monitor traffic on a bus counting flowing packets,
    - Monitor AXI4 & CR protocol errors





Testing strategy

- Each individual fw block comes with it's own testbench and has been tested in fw simulation in standalone mode
  - Separate tbs for Data Reception, Unpacker, PedSub, FIR, HF block etc

- Another set of testbenches used to test groups of fw blocks:
  - Data Router (Data Reception + Unpacker)
  - TPG block (Header S/C + PedSub + FIR + HF)

- Full HF chain testbench, including file-based Sink & Source, also available

Testing sequence:

- *Step1*: Sw generated input/output files used for firmware simulation
- *Step2*: Fw testing with on ZCUs using same input files
- *Step3*: Integrate a single HF link with FELIX and test on FELIX coldbox (no connections to ProtoDUNE) with same input files
- *Step4*: Integrate 2 HF links (or more) with FELIX and test with real PD data

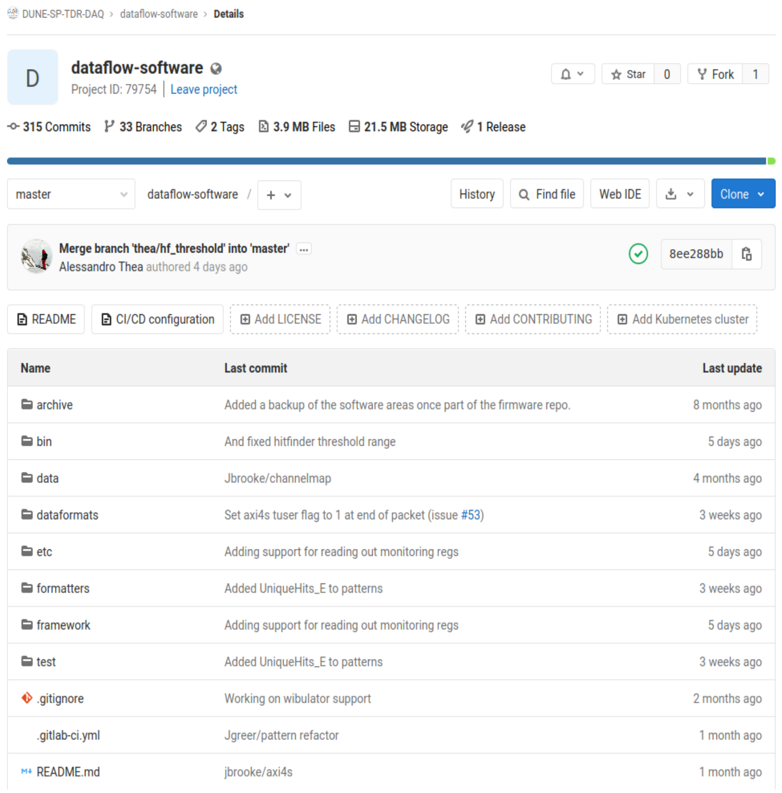
Software emulation & Validation

TPG Simulation: Purpose and Scope

- Provide software tools for testing, debugging and bit-accurate verification during the fw development lifecycle

- TPG implementation in Python aims to emulate firmware behaviour
  - Pedestal subtraction (with State S/R feature)
  - FIR filtering (with State S/R feature)
  - Hit Finding

- Software package in gitlab
  - Public repository, collaborative effort
  - Documentation, CI



DUNE-SP-TDR-DAQ › dataflow-software › Details

dataflow-software
Project ID: 79754 | Leave project

315 Commits · 33 Branches · 2 Tags · 3.9 MB Files · 21.5 MB Storage · 1 Release

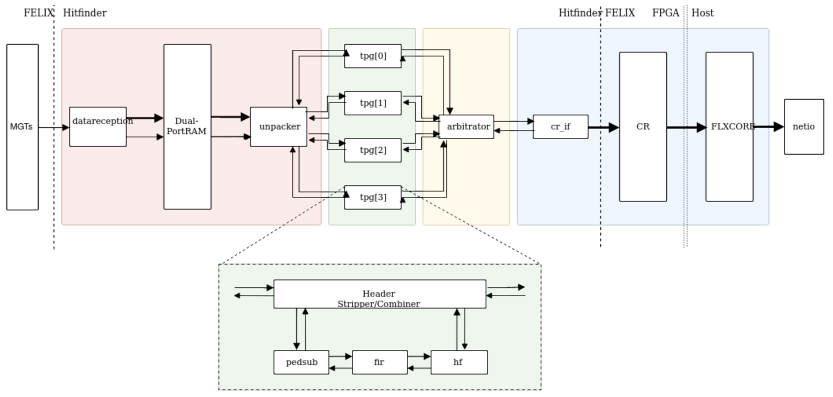
master · dataflow-software /

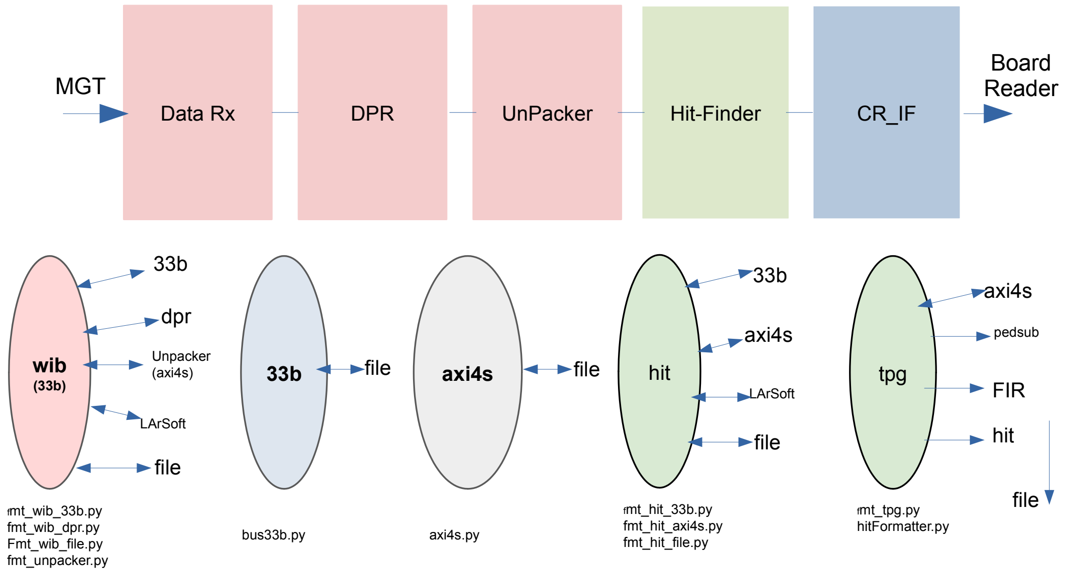
Merge branch 'thea/hf_threshold' into 'master'
Alessandro Thea authored 4 days ago · 8ee288bb

Name	Last commit	Last update
archive	Added a backup of the software areas once part of the firmware repo.	8 months ago
bin	And fixed hitfinder threshold range	5 days ago
data	Jbrooke/channelmap	4 months ago
dataformats	Set axi4s tuser flag to 1 at end of packet (issue #53)	3 weeks ago
etc	Adding support for reading out monitoring regs	5 days ago
formatters	Added UniqueHits_E to patterns	3 weeks ago
framework	Adding support for reading out monitoring regs	5 days ago
test	Added UniqueHits_E to patterns	3 weeks ago
.gitignore	Working on wibulator support	2 months ago
.gitlab-ci.yml	Jgreer/pattern refactor	1 month ago
README.md	jbrooke/axi4s	1 month ago

Data-flow and data-formats

- Dataflow in sw replicates the fw components and interfaces
- Python classes are used to represent data at different stages of the chain
- Data printed to files, passed among classes and converted to any format
- Flexible pattern generation for firmware debugging





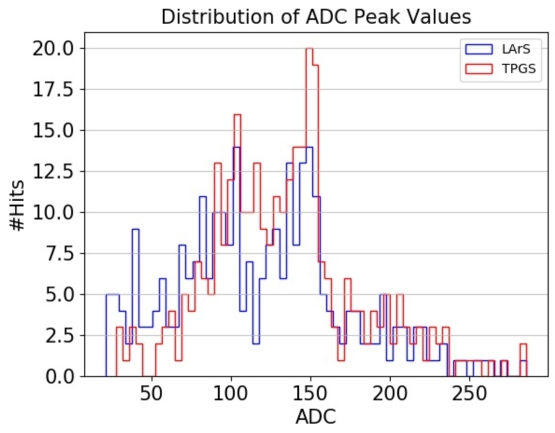
TPG Simulation: Validation

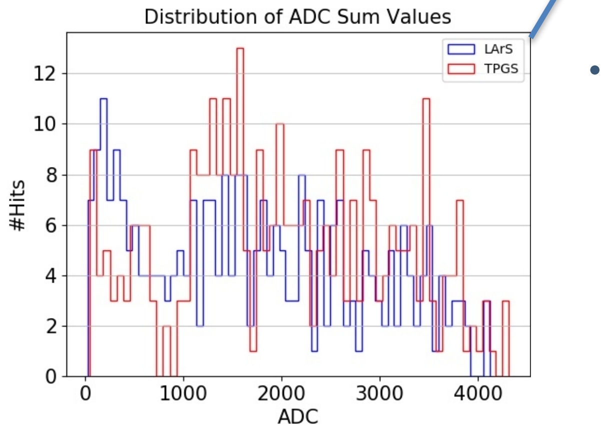
- Emulation of the TPG algorithms was initially done using a LArSoft implementation
  - Used to validate pre-packetized versions of the firmware
  - Extended to emulate packetized TPG block, but lacks flexibility required to ‘look inside’ the firmware
  - Well understood: a ‘gold standard’ to debug python TPG against
- New TPG-sim compared with the LarSoft implementation to validate it
- Provide support to the TPG-Firmware as it is being developed.
- Able to use dataflow-software tools to look at the hits found by the firmware
  - Both the standard version and the validation version of firmware (the one including the PedSub values) were run in the protoDUNE.

TPG Sim: Validation with LArSoft

- Since the firmware and software validation were based on the main LArSoft code, before diverging, it is necessary step to verify dataflow-software against it.
- We have generated few patterns, to validate the TPG-sim, and have processed them through LArSoft to compare the various quantities. Here are few results:

Patterns	No.of Hits TPG-sim	No.of Hits LArSoft	Match %
FixedHits_A	256	256	100%
FixedHits_D	256	256	100%
UniqueHits_C	233	225	79.6%
UniqueHits_D	322	286	58.04%





- There is 100% agreement in trivial patterns
- There are still a lot of cases (as can be seen in this pattern), where out TPG sw does not agree with LArSoft standard
- We have few hints and understanding about the disagreement. Currently investigating and fixing these bugs

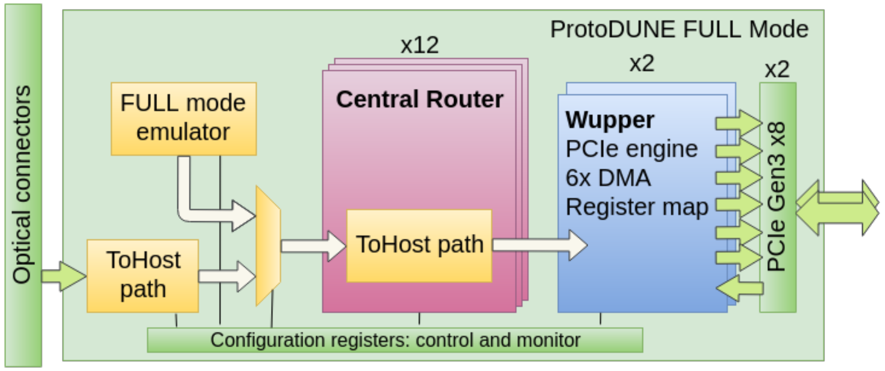
FELIX integration

FELIX Firmware - ProtoDUNE

- Generic bi-directional i/f between frontend and pc
- 2 link protocols for data transfer
  - GBT/Versatile link
  - FULLmode protocol (single wide data stream)

- FLX-712 (16 PCIe3 lanes), FULLmode 12-link configuration

- Jumbo Block Super Chunk (JBSC)
  - *Super Chunk: packed together Chunks to minimize memory-copy effort*
  - Transfer data to host with up to 4k DMA blocks
  - Reliable 100 Gbps transfer rate
  - Extensively tested in data-taking (2018-2019)
  - Loosely aligned with master
    (Last common ancestor ~June 2019)



FELIX firmware - Hitfinder integration challenge

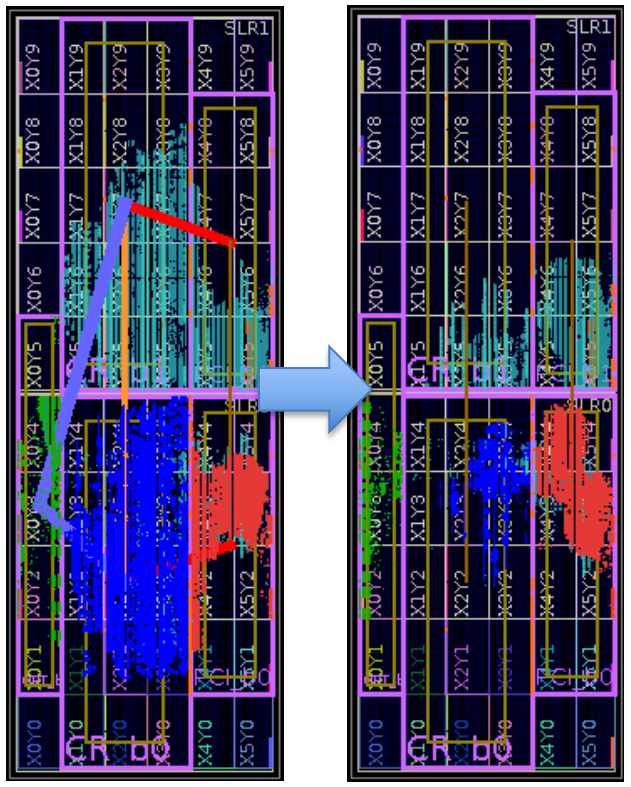
- Not conceived to accommodate “user” firmware
  - Lack of area separation between core firmware and user extensions
  - Timing closure challenging, even in the JBSC vanilla version
  - Long build times, even for bare-bone designs (wupper-only builds)
  - Monolithic approach to register definition (1-size-fits-all register map)
    - No hierarchical address map where one can reserve a user address range ➔ adding new registers means modifying top address table & regenerating everything
  - Monolithic firmware repository
  - Very limited availability on evaluation platforms

- We should keep in mind however that:
  - FELIX fw was a snapshot version of a system still in development
  - Has since undergone an important redesign, which makes customizability a much more supported feature (lot of feedback from PDUNE has been taken into account)

Making space

- Optimization of JBSC chFIFOs (64kB)
  - shrunk back to 16kB (as in master)
- Removal of FULLmode “toHost” logic
  - Un-used FPGA resources in ProtoDUNE
- Significant improvement in timing closure stability

FLX712 - KCU115 resources

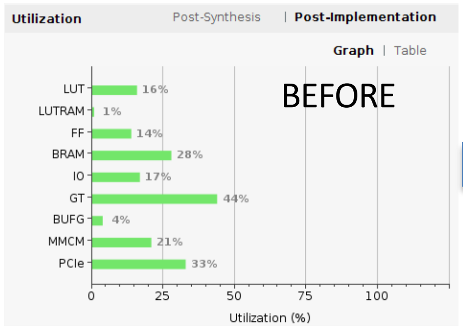


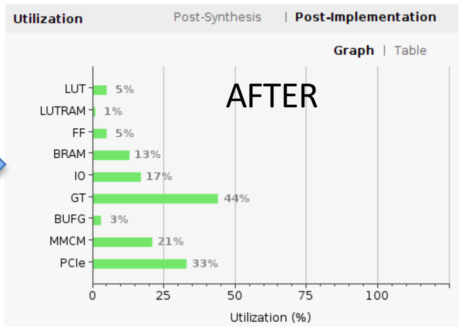
BEFORE AFTER

GBT

Central Router

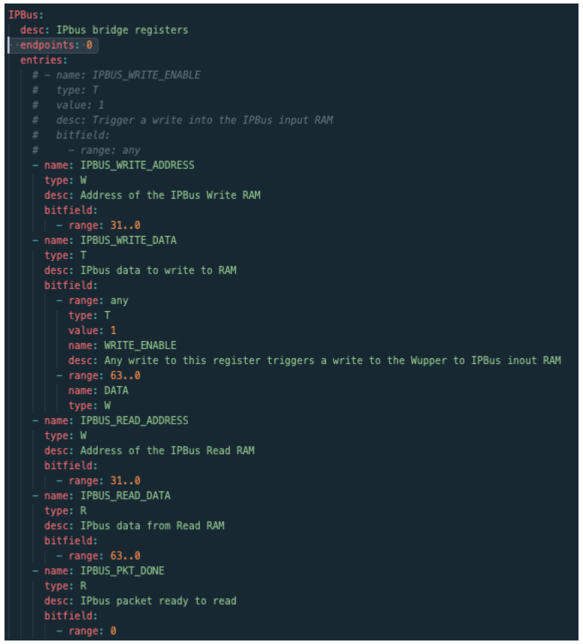
Wupper/DMA





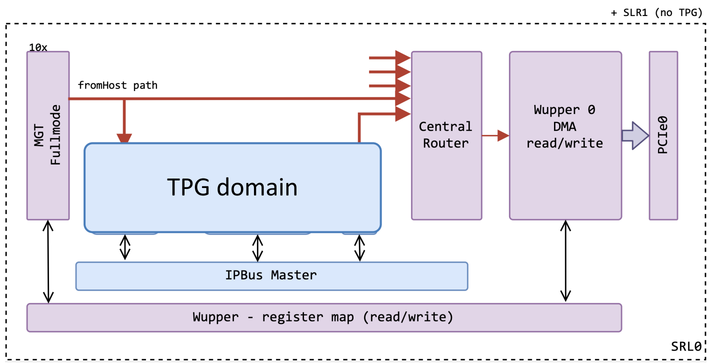
Integration strategy

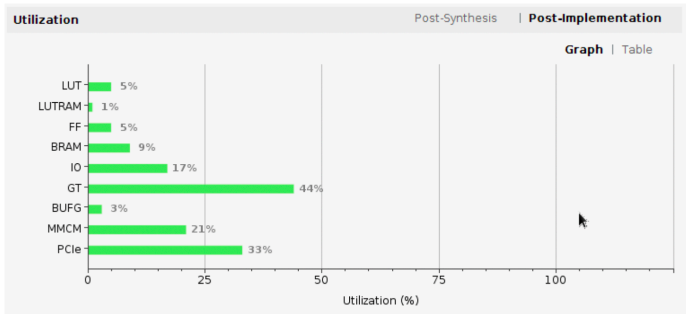
- Keep FELIX and TPG firmware domains separate
  - TPG as a “pod” inside the FELIX infrastructure
- Added an IPbus-Wupper bridge block to use **unmodified** TPG firmware and software tools
- **Advantages**
  - Very thin layer of custom fw/sw (the bridge)
  - **Full re-use of existing FELIX and TPG tools**
  - **Seamless switch between TPG dev boards and Felix**
  - Minimal resource impact
    - IPBus master footprint negligible
    - IPbus-Wupper bridge relies on 5 register-map registers

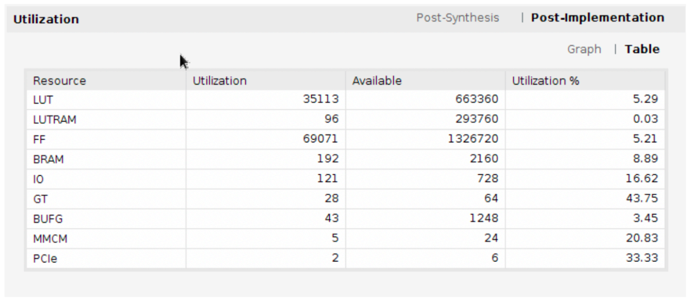


```
IPBus:
  desc: IPbus bridge registers
  endpoints: 0
  entries:
    # - name: IPBUS_WRITE_ENABLE
    #   type: T
    #   value: 1
    #   desc: Trigger a write into the IPBus input RAM
    #   bitfield:
    #     - range: any
    - name: IPBUS_WRITE_ADDRESS
      type: W
      desc: Address of the IPBus Write RAM
      bitfield:
        - range: 31..0
    - name: IPBUS_WRITE_DATA
      type: T
      desc: IPbus data to write to RAM
      bitfield:
        - range: any
          type: T
          value: 1
          name: WRITE_ENABLE
          desc: Any write to this register triggers a write to the Wupper to IPBus inout RAM
        - range: 63..0
          name: DATA
          type: W
    - name: IPBUS_READ_ADDRESS
      type: W
      desc: Address of the IPBus Read RAM
      bitfield:
        - range: 31..0
    - name: IPBUS_READ_DATA
      type: R
      desc: IPbus data from Read RAM
      bitfield:
        - range: 63..0
    - name: IPBUS_PKT_DONE
      type: R
      desc: IPbus packet ready to read
      bitfield:
        - range: 0
```

FELIX and TPG fw domains and floorplanning





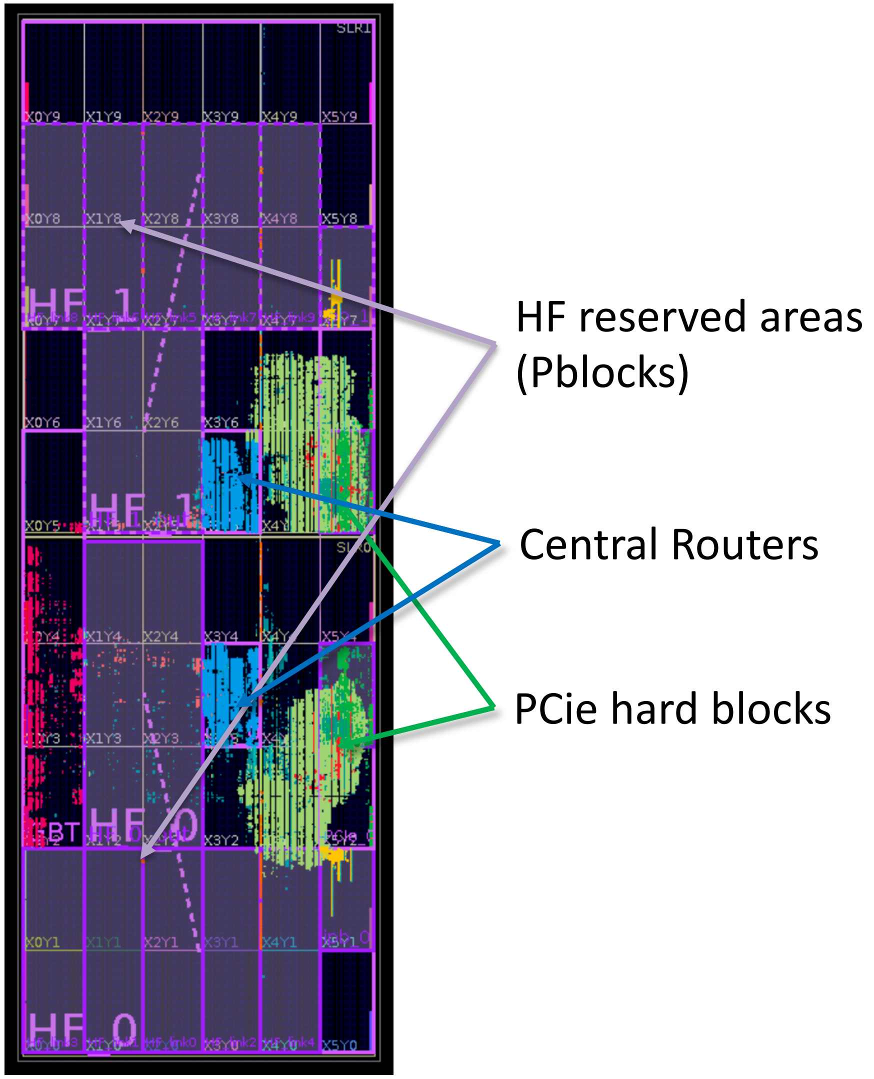


Utilization

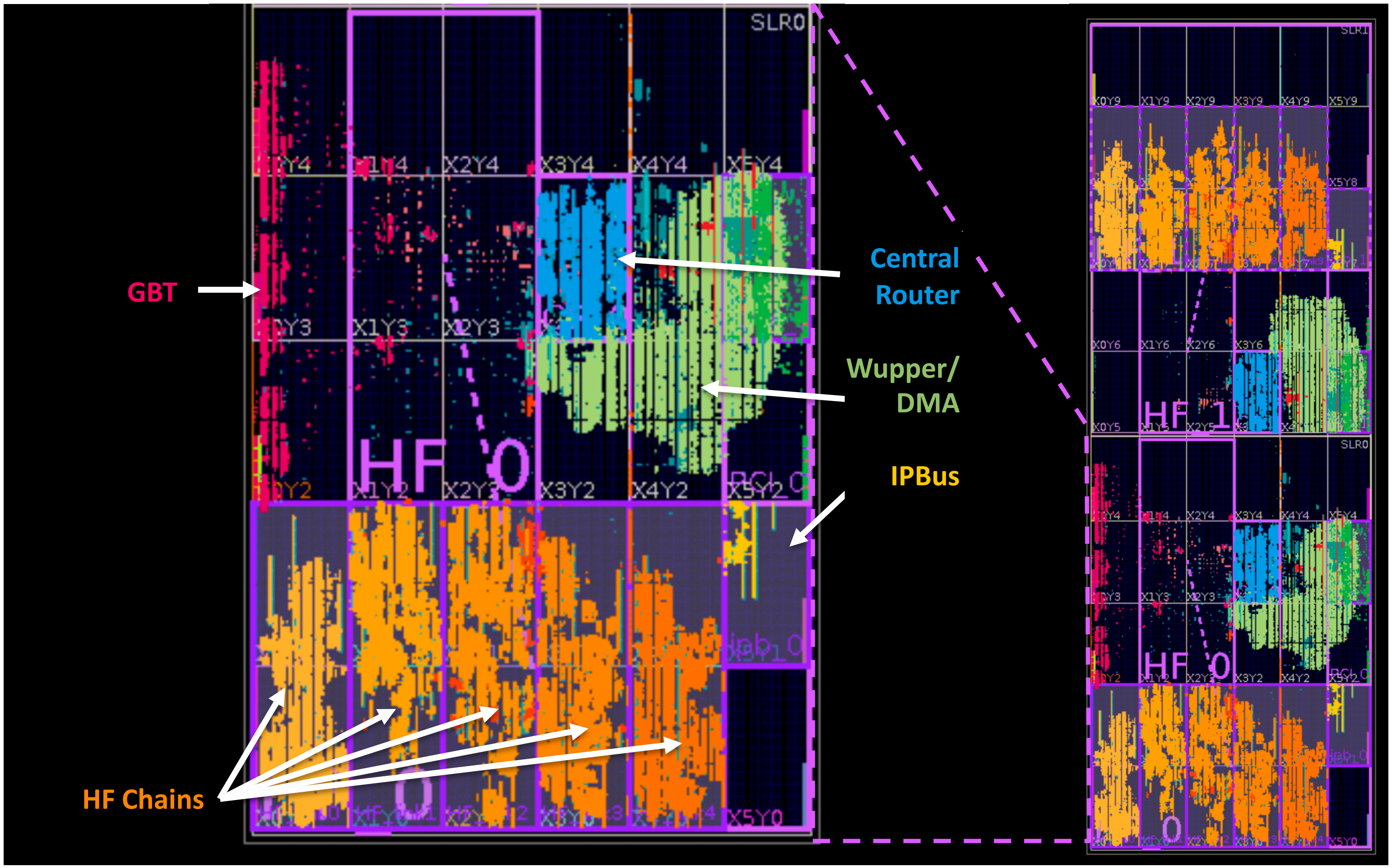
Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	35113	663360	5.29
LUTRAM	96	293760	0.03
FF	69071	1326720	5.21
BRAM	192	2160	8.89
IO	121	728	16.62
GT	28	64	43.75
BUFG	43	1248	3.45
MMCM	5	24	20.83
PCIe	2	6	33.33

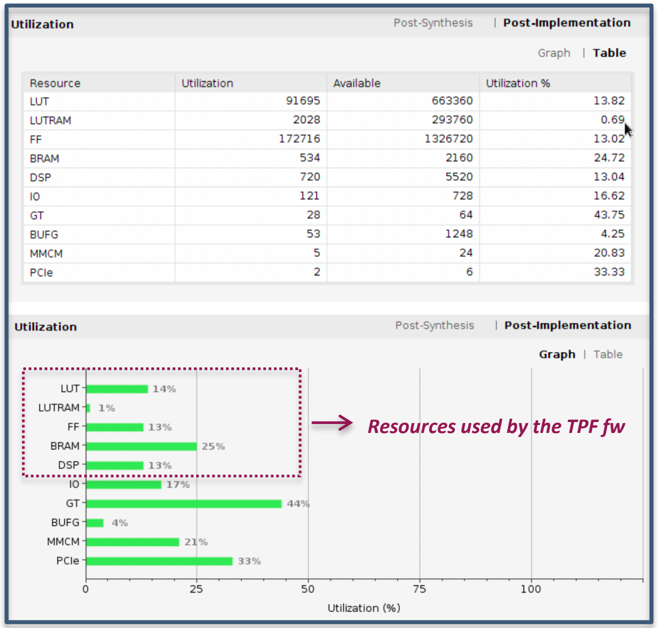


10-links design floorplan



10-links design resources

Total Resources 10xHF + FELIX



Utilization — Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	91695	663360	13.82
LUTRAM	2028	293760	0.69
FF	172716	1326720	13.02
BRAM	534	2160	24.72
DSP	720	5520	13.04
IO	121	728	16.62
GT	28	64	43.75
BUFG	53	1248	4.25
MMCM	5	24	20.83
PCIe	2	6	33.33

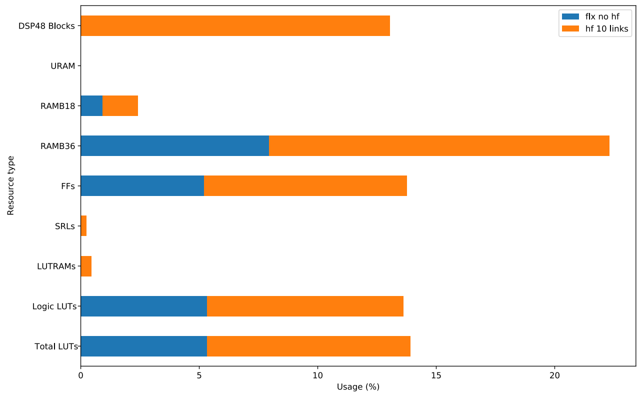
Utilization — Post-Synthesis | **Post-Implementation**

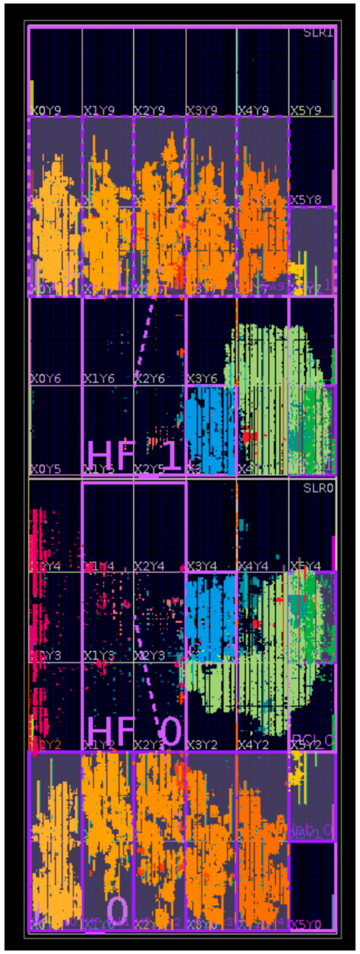
Graph | Table

Resource	Utilization (%)
LUT	14%
LUTRAM	1%
FF	13%
BRAM	25%
DSP	13%
IO	17%
GT	44%
BUFG	4%
MMCM	21%
PCIe	33%

Resources used by the TPF fw

Total Resources with/without 10xHF





Acronyms:

- LUT: LookUp-Tables
- LUTRAM : LookUp Table RAM
- FF: Flip-Flops
- BRAM: block of Random Access Memory
- DSP: Digital Signal Processing
- IO: Input/Output blocks
- GT: Gigabit Tranceivers
- BUFG: global buffers
- MMCM: Mixed-Mode Clock Manager

Science & Technology Facilities Council Rutherford Appleton Laboratory

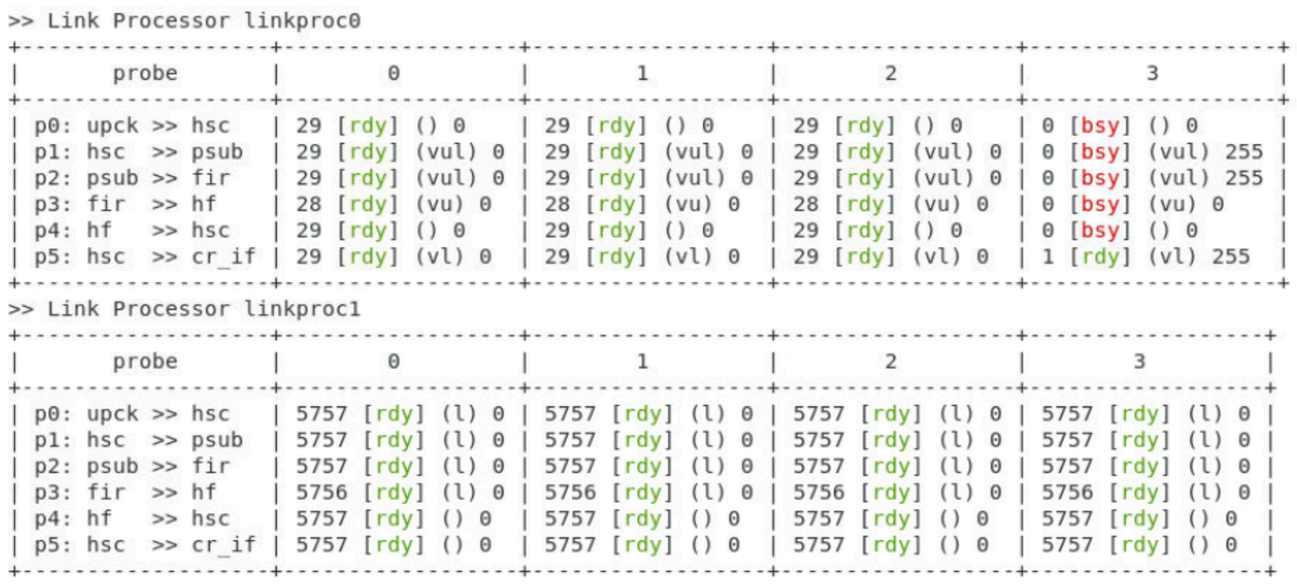
ProtoDUNE Demonstration & Results

ProtoDUNE-SP Testing

- ProtoDUNE-SP FELIX testing was split between the FELIX cards in server np04-srv-030 and np04-srv-029
- np04-srv-030: Readout connected to ColdBox that had no APA inside. Used to test initial firmware builds without requiring detector time
- np04-srv-029: FELIX connected to APA6 which allows reading of raw ADC links from the APA

FELIX control

- Makes use of **FELIX software suite** for card initialisation, logical link (elink) configuration and firmware flashing
- **Dataflow-software** package was developed and used to specifically configure, control and monitor aspects of the Hit Finder links over IPBus
- **Ipb-o-flx** package was required for interfacing with FELIX card Hit Finder modules using IPBus

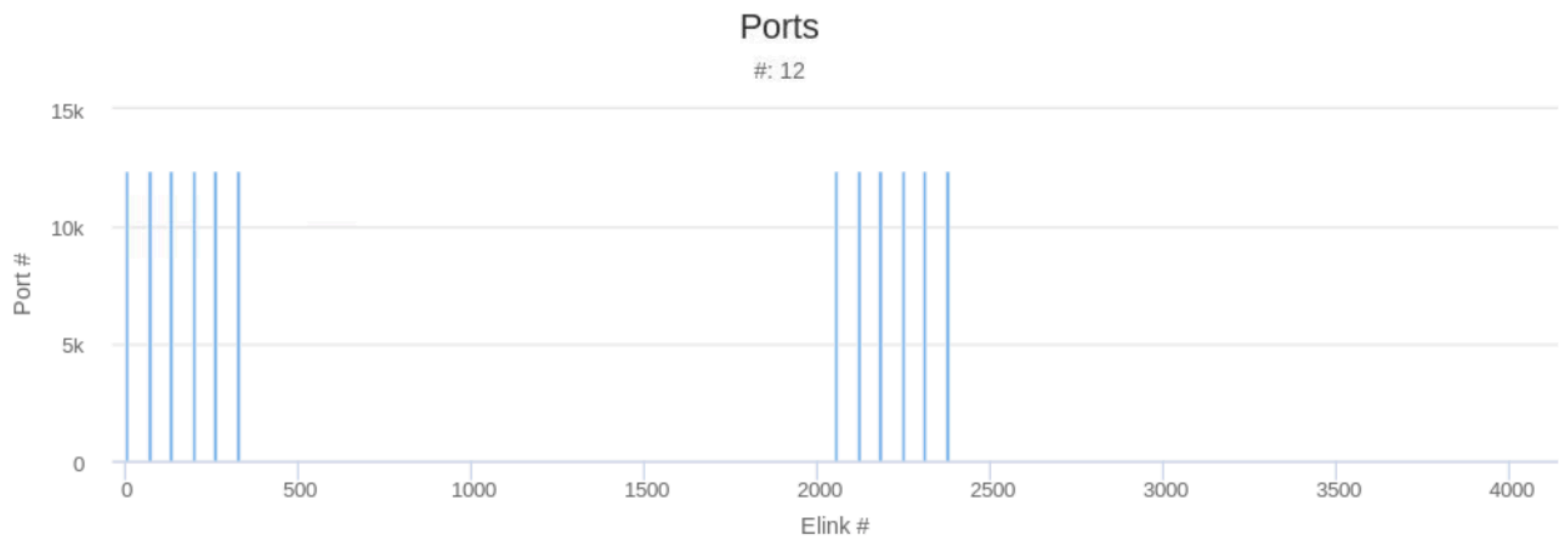


```
>> Link Processor linkproc0
+-------------------+-------------------+-------------------+-------------------+-------------------+
|       probe       |         0         |         1         |         2         |         3         |
+-------------------+-------------------+-------------------+-------------------+-------------------+
| p0: upck >> hsc   | 29 [rdy] () 0     | 29 [rdy] () 0     | 29 [rdy] () 0     | 0 [bsy] () 0      |
| p1: hsc  >> psub  | 29 [rdy] (vul) 0  | 29 [rdy] (vul) 0  | 29 [rdy] (vul) 0  | 0 [bsy] (vul) 255 |
| p2: psub >> fir   | 29 [rdy] (vul) 0  | 29 [rdy] (vul) 0  | 29 [rdy] (vul) 0  | 0 [bsy] (vul) 255 |
| p3: fir  >> hf    | 28 [rdy] (vu) 0   | 28 [rdy] (vu) 0   | 28 [rdy] (vu) 0   | 0 [bsy] (vu) 0    |
| p4: hf   >> hsc   | 29 [rdy] () 0     | 29 [rdy] () 0     | 29 [rdy] () 0     | 0 [bsy] () 0      |
| p5: hsc  >> cr_if | 29 [rdy] (vl) 0   | 29 [rdy] (vl) 0   | 29 [rdy] (vl) 0   | 1 [rdy] (vl) 255  |
+-------------------+-------------------+-------------------+-------------------+-------------------+
>> Link Processor linkproc1
+-------------------+-------------------+-------------------+-------------------+-------------------+
|       probe       |         0         |         1         |         2         |         3         |
+-------------------+-------------------+-------------------+-------------------+-------------------+
| p0: upck >> hsc   | 5757 [rdy] (l) 0  | 5757 [rdy] (l) 0  | 5757 [rdy] (l) 0  | 5757 [rdy] (l) 0  |
| p1: hsc  >> psub  | 5757 [rdy] (l) 0  | 5757 [rdy] (l) 0  | 5757 [rdy] (l) 0  | 5757 [rdy] (l) 0  |
| p2: psub >> fir   | 5757 [rdy] (l) 0  | 5757 [rdy] (l) 0  | 5757 [rdy] (l) 0  | 5757 [rdy] (l) 0  |
| p3: fir  >> hf    | 5756 [rdy] (l) 0  | 5756 [rdy] (l) 0  | 5756 [rdy] (l) 0  | 5756 [rdy] (l) 0  |
| p4: hf   >> hsc   | 5757 [rdy] () 0   | 5757 [rdy] () 0   | 5757 [rdy] () 0   | 5757 [rdy] () 0   |
| p5: hsc  >> cr_if | 5757 [rdy] () 0   | 5757 [rdy] () 0   | 5757 [rdy] () 0   | 5757 [rdy] () 0   |
+-------------------+-------------------+-------------------+-------------------+-------------------+
```

Link configuration

- Configuration enables 12 FULLMODE links
  - 10 being the raw data streams and 2 aggregating data hits

- Using the felixcore application from FELIX software suite the data from individual e-links are published on the network.



*** E-link:** variable-width logical link on top of the GBT protocol. Can be used to logically separate different streams on a single physical link.

Readout tools

- **Fdaq:** Used to continuously read data from specified links.
  - Not suited for sustained readouts and writing of data to disk at the same time due to the high payload (WIB frame) rate.

- **Netio-cat:** Subscribes to a link published by felixcore and prints user payloads (chunks).
  - Useful for integrity checks and low-level diagnostics of the system.

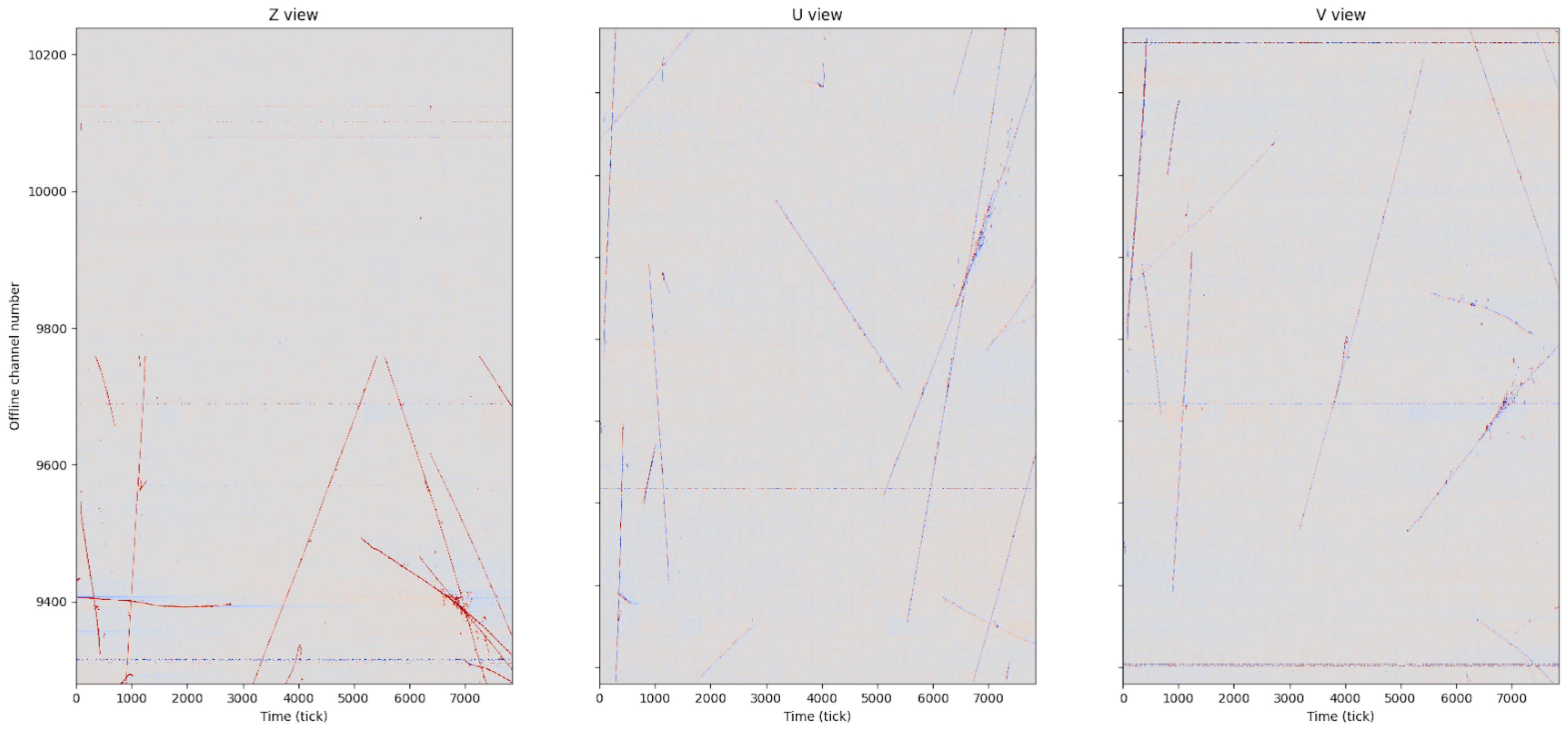
- **OnHost BoardReader variant:** A standalone version of the ProtoDUNE-SP FELIX OnHost BoardReader application.
  - Serializes the data from the DMA buffer into software defined latency buffers.
  - Since this is based on the same sw foundation as the current ProtoDUNE-SP FELIX OnHost BoardReader used in software HitFinding algorithms, it demonstrates the ability of the system to integrate into ProtoDUNE-SP for production

Data capture procedure

- **hfButler** tool is Hit Finder specific software used to control and configure the Hit Finder firmware blocks.
- Configure OnHost BR to be ready to buffer up data when the links are enabled
- Run hfButler configuration commands to configure Hit Finder links for test. Configuration depends on test objective (number of links enabled, Hit Finder threshold setting, channels masked)
- Issue start command to OnHost BR to start serializing DMA data stream.
- When enough data is collected, issue a stop command to the BR.

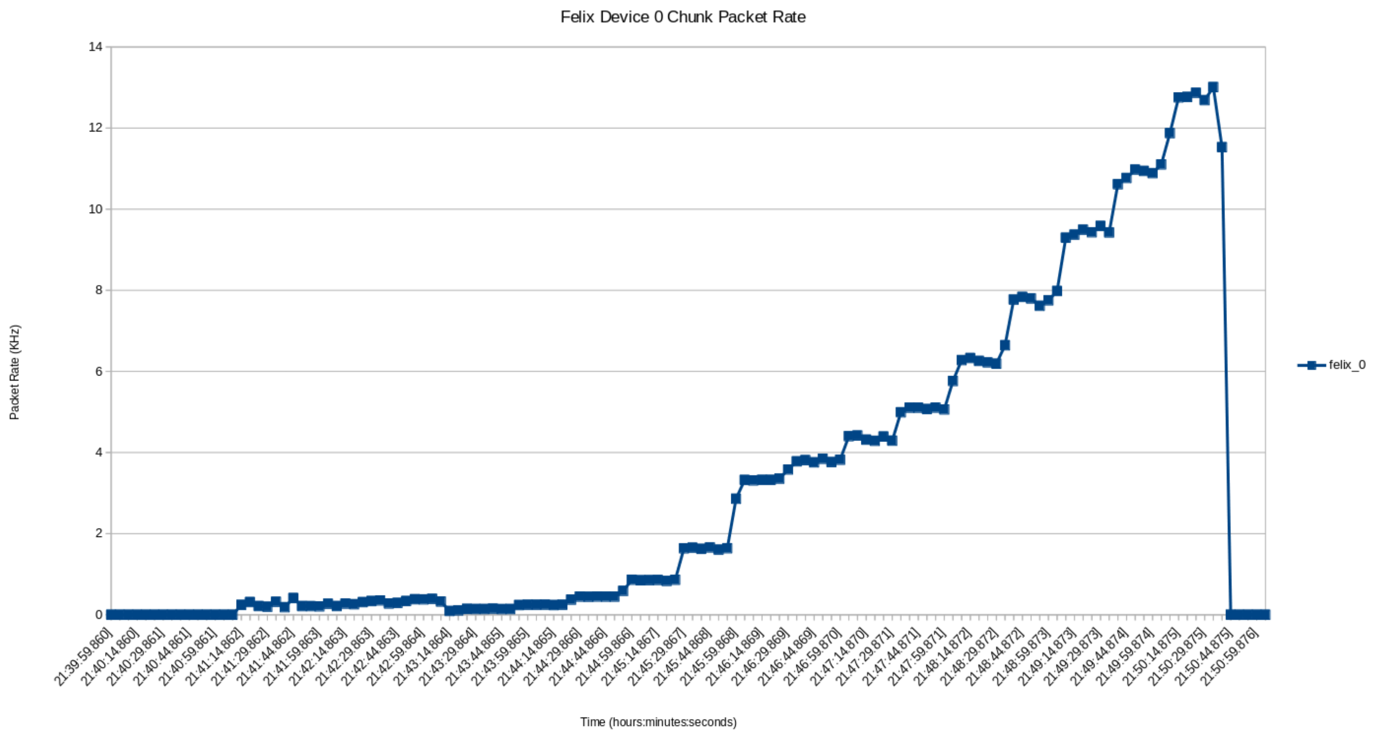
Capture analysis

- Captured ADC link binary data processed and visualised using **Felix-long-readout** tool and **waveform-tools**



Threshold scan/stability test

- OnHost board reader also provides useful monitoring information such as chunk packet rates. This is used in the graph below to measure the effect of adjusting the threshold value of 5 Hit Finder links on the number of hit packets generated

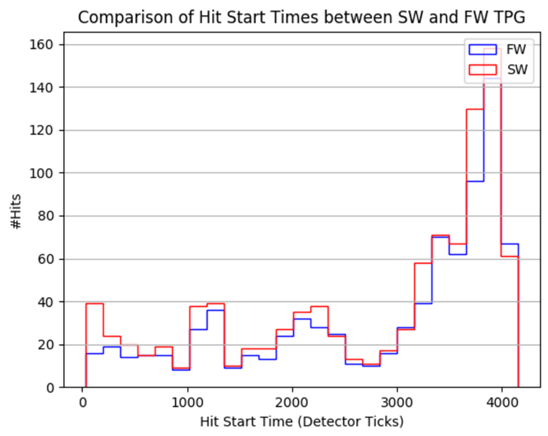


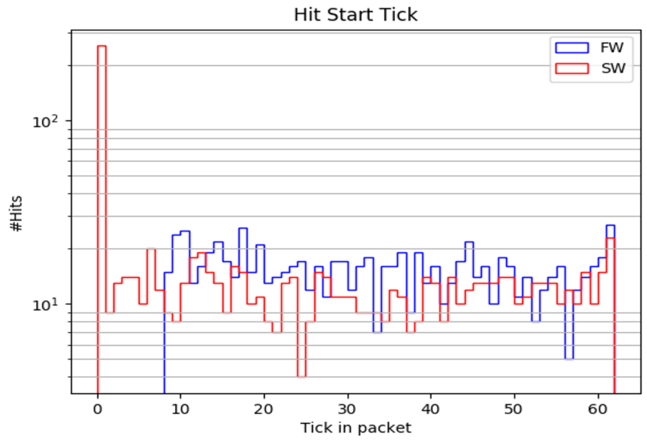
Performed Tests

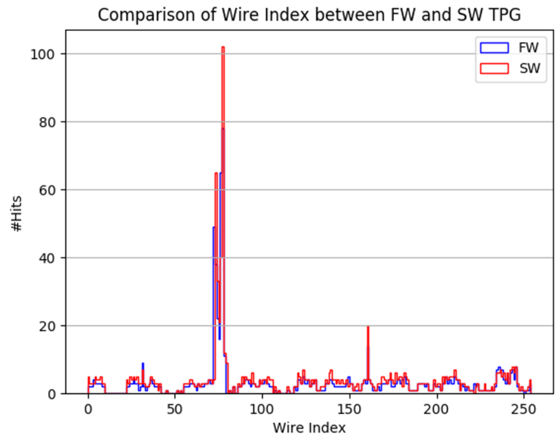
- Testing with ProtoDUNE-SP tracked via e-log and gitlab issue tracker.
  - 06/02/20 - Initial testing on np04-srv-029 with 2 Hit Finders
  - 06/05/20 - Demonstrated 12 link FULLMODE capture with 2 Hit Finders. Lack of pedestal Save State Restore causes all 64 tick packets to record as a hit
  - 06/06/20 - Tested firmware with State-SR functionality for pedestal subtraction
  - 06/24/20 - 12 link capture with State-SR enabled pedestal and filter
  - 06/27/20 - 12 link fdaq acquisition run for 12 hours demonstrating no block or chunk errors recorded
  - 07/10/20 - Tested and demonstrated 12 link capture with 10 Hit Finders. Demonstrated link control and gathered information on chunk rates
  - 07/16/20 - 12 e-link captures with 2 Hit Finders. Channel masking tested and blocking of multiple hit packets into larger chunks to relieve Board Reader of high chunk copy rates
  - 07/17/20 - 12 e-link captures with 10 Hit Finders. Tested threshold adjustment

ProtoDUNE fw vs TPG Sim

- Looking at the hits from the fw TPG in ProtoDUNE vs TPG-sim predictions
  - TPG-sim still under debugging but finds almost the same number of hits on each wire at the same time.
  - Some indications of edge case bug in fw TPG, such as start tick.
  - There are captures include pedestal values which allow the validation process to be rigorous.
  - Debugging and feature addition will go hand in hand as the system matures.

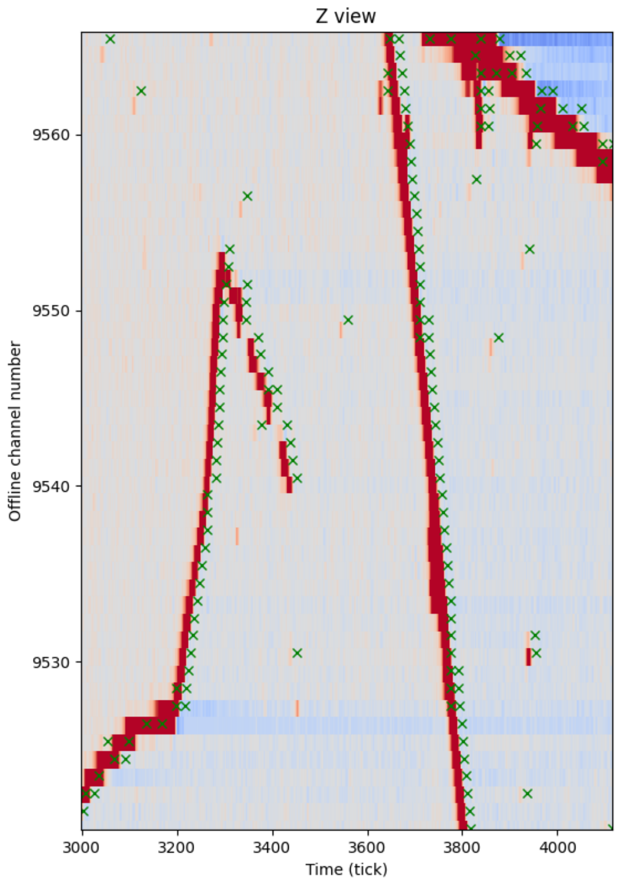


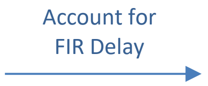


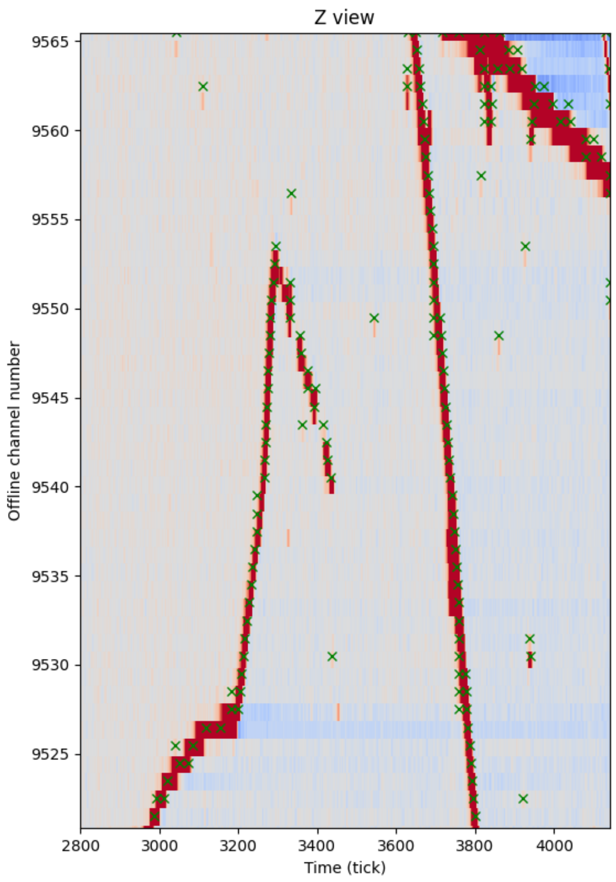


ProtoDUNE fw Hits

- TPG Firmware appears to be functioning well despite any remaining bugs. Looking at the collection wires (unipolar signal):



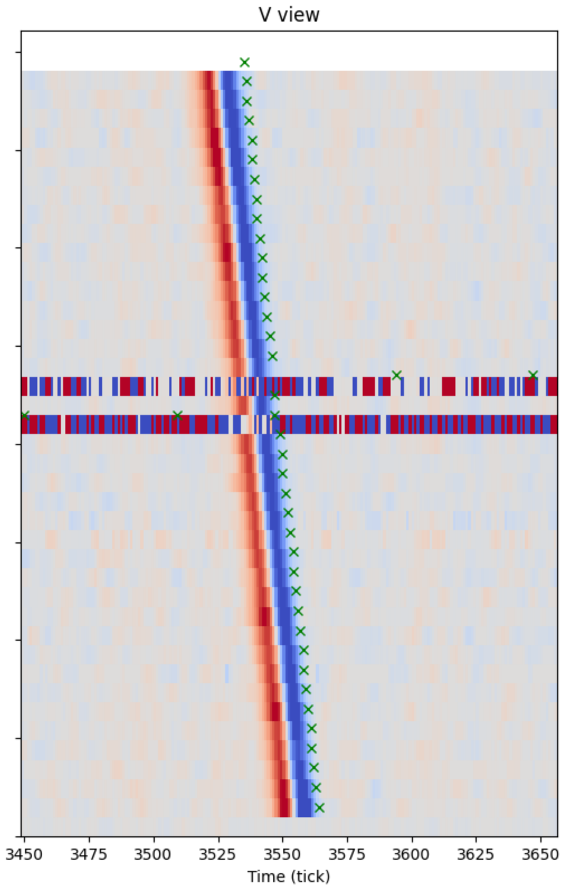


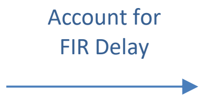


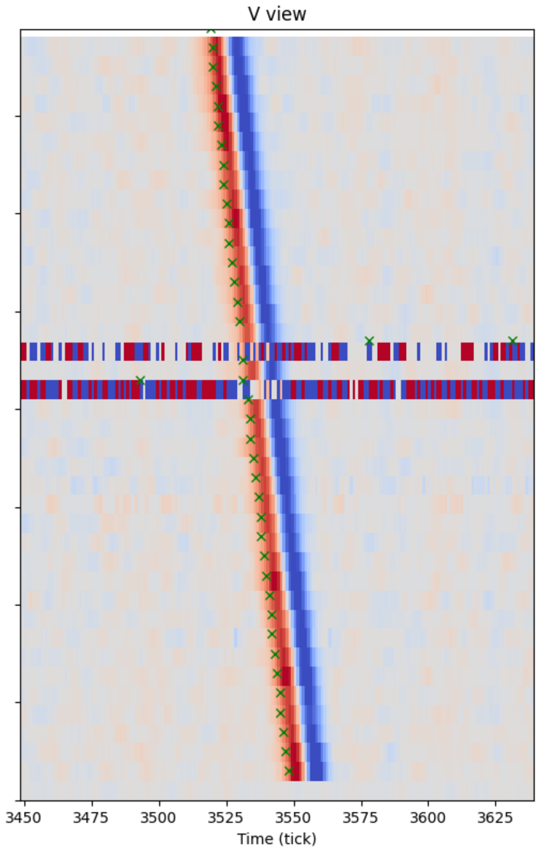
Science & Technology Facilities Council
Rutherford Appleton Laboratory

ProtoDUNE fw Hits

- TPG firmware appears to be functioning well despite any remaining bugs. Looking at the induction wires (antisymmetric bipolar signal):







Future work & Summary

Future work & improvements

- Continue debugging python TPG
  - Improve agreement with LArSoft
- Redo testing sequence and verify fw vs sw
- Use captured raw data for hw tests

- Evaluate the newest developments in the FELIX fw area to see if and how things have improved also for DUNE
- Work on integration with FELIX daq for PDII
- Improve/simplify testing & data taking procedure
  - Unify existing ipbus scripts

- Make entire system robust & production ready

Summary

- We have implemented in firmware a Hit Finder chain that processes the ADC values of a fibre and produce hits
- 10 Hit Finders have been successfully integrated on a FELIX board, thus proving we are able to process a full APA on a single FPGA
- Performed initial tests on PDUNE with real data
- New python TPG-sim is being debugged against LArSoft
- Have identified the areas where we need to concentrate our effort in order to deliver a production-ready system

Criteria Analysis

Criteria Analysis

- **Features**
  - ***Which features have existing implementations and/or demonstrators?***

    Ans: ProtoDUNE data demonstrator for processing a full APA with a10-link HF integrated with FELIX

  - ***What features are missing and how much further development is required?***

    Ans: A complete implementation of the data processing chain and the hit finder algorithm on collection wires has been done.
    Further development for the sw DAQ tools includes the following tasks:
    i) Consolidation of the TPG configuration sequence,
    ii) Porting of the control python scripts to C++ and integration with the DUNE FELIX application,
    iii) Configuration management (from files or other),
    iv) Detailed monitoring.

  - ***How does the solution interface to other components of the Upstream DAQ, and wider DAQ? (e.g. Dataflow, CCM, Data Selection interfaces)***

    Ans: The FPGA-based TPG plugs-in on ProtoDUNE FELIX firmware, exploiting available resources and interfaces without modifying any core function. In a similar fashion, the “Hit Finder board reader” sw is a seamless extension of the on-host FELIX board-reader.

    The FPGA-based TPG will interface with CCM, DF and DS through what today would be called the "HitFinder board reader" (not implemented yet). The HF-BR will:
    - Receive control commands and configuration data from the CCM and apply them to the TPG firmware, as well as monitor the status of the same and publish the information to the monitoring system
    - Publish hits towards the DS system in the agreed format and towards DF, whenever requested
    - Respond to DF event fragment request.

Criteria Analysis

- **Adaptability**
  - ***Were any components that rely on this technology integrated in existing DAQ systems? (e.g.: within ProtoDUNE-SP DAQ)***

    Ans: i) Integration of 10 link Hit Finder with FELIX, ii) OnHost BoardReader variant for Hit Finding was based on the same sw foundation as the current ProtoDUNE-SP FELIX OnHost BoardReader.

  - ***How would the solution adapt to potential new requirements of the DUNE DAQ? (for example, different TP algorithms, RoI-based readout)***

    Ans: Current firmware implementation is flexible enough to be able to adapt to new requirements, provided they are compatible, up to a point, with the way data is currently being handled (i.e. parallel processing of optical links, wires processed in a round-robin fashion, data packets).
    New algorithms can be implemented within (or replacing) the existing firmware, baring always in mind that in a firmware implementation and there is an upper limit on available resources.

  - ***Is it possible to tune and align the solution to support new ideas (e.g.: additional interfaces) or are there intrinsic structures that constrain potential extensions/modifications?***

    Ans: The current resource utilisation for a 10-link HF is low enough to ensure a certain amount of flexibility to be able to support new ideas. However, as stated previously, there is the constraint imposed by the available resources on an FPGA.

Criteria Analysis

- **Reliability**
  - ***Can you ensure data integrity using this technology?***
  - *Ans: Data integrity checks can mostly be run in the Data Reception block and in various checkpoints of the firmware.*
  - ***How do you handle and mitigate the known failure scenarios (above) and other errors in order to avoid their propagation to other components?***
  - *Ans: Data Reception includes basic checks for data corruption and the CR-if* detects and removes corrupted hit packets. *We also know that continuous applied backpressure will cause the Data Router to drop incoming packets and we plan to make the system more robust against this.*

Criteria Analysis

- **Long term support & maintenance**
  - ***What is the balance between in-house development and COTS components?***

    Ans: Current fw implementation uses the Xilinx Vivado tool and the IP cores that are included in its license. We also make use of off-the-shelf developments boards for testing purposes. Finally, the IPBUS sw that is used will be maintained through CMS.

  - ***Does the solution require specific hardware products? (E.g.: only works with a specific SSD variant, or with any kind? Requires specific FPGA/CPU models or features? Are there implications for spares?)***

    Ans: No. TPG firmware implementation does not depend on the use of a specific Xilinx FPGA.

  - ***Can this solution profit from research and development outside DUNE (eg. by manufacturers, other experiments, etc.)?***

    Ans: The FELIX board (and firmware) is mainly developed for the ATLAS experiment. As a result, the FELIX integration allows us to benefit from the R&D of another experiment.

  - ***How strong and connected is the community of users and partners around the technology?***

    Ans: Strengthening ties with the FELIX community will benefit our future work.

  - ***Do we have sufficient engineers and developers with the necessary expertise to support this solution?***

    Ans: There are still a number of things that need to be completed, which require maintaining approx. the currently available manpower. Leveraging on the effort made to define solid fw/sw validation procedures, we expect that the effort required to maintain the system in long term will be relatively limited.

Criteria Analysis

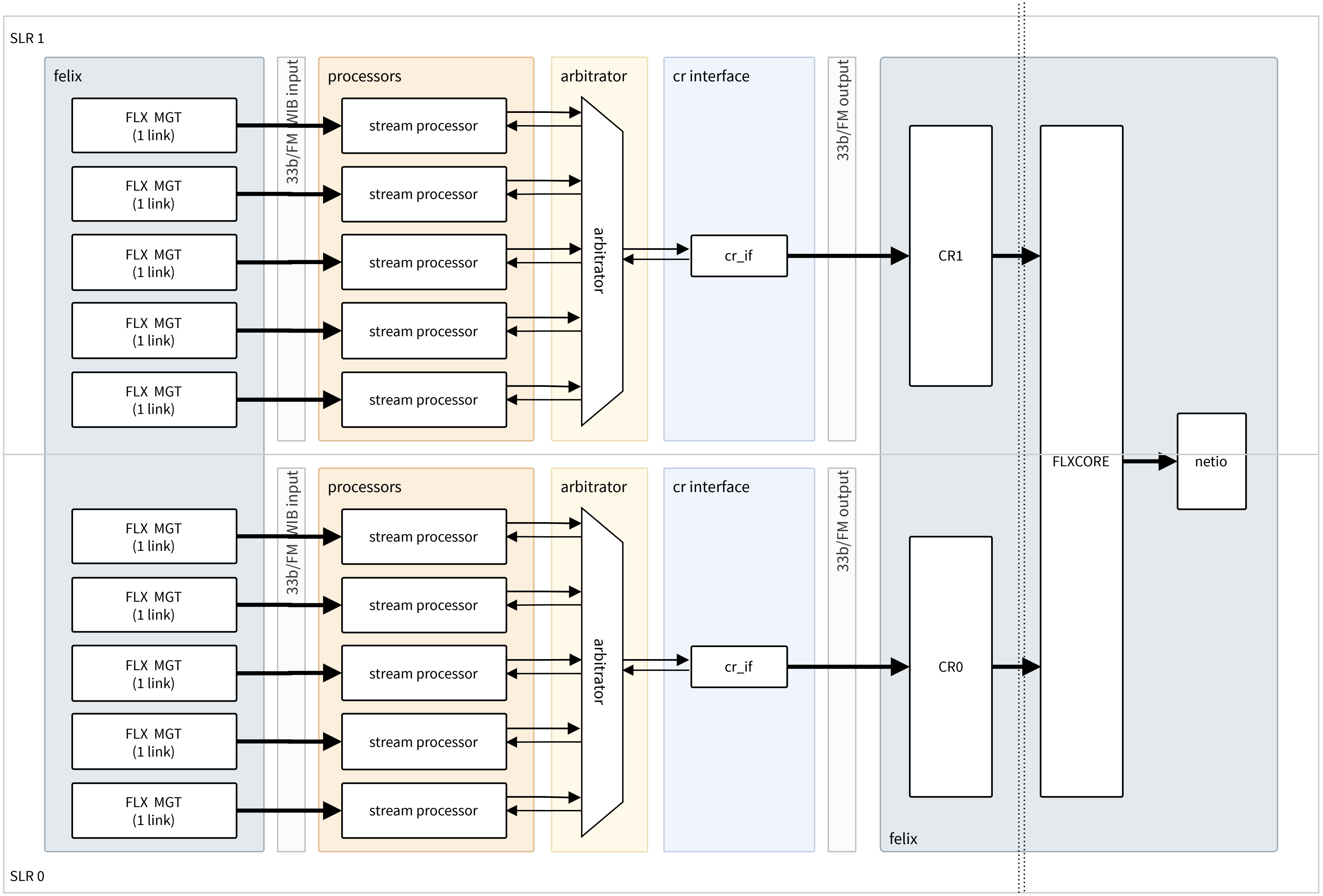
- **Resource Requirements**
  - ***What are the resources (FPGA, cpu-cycles, memory) required by the solution as it stands now, per APA?***

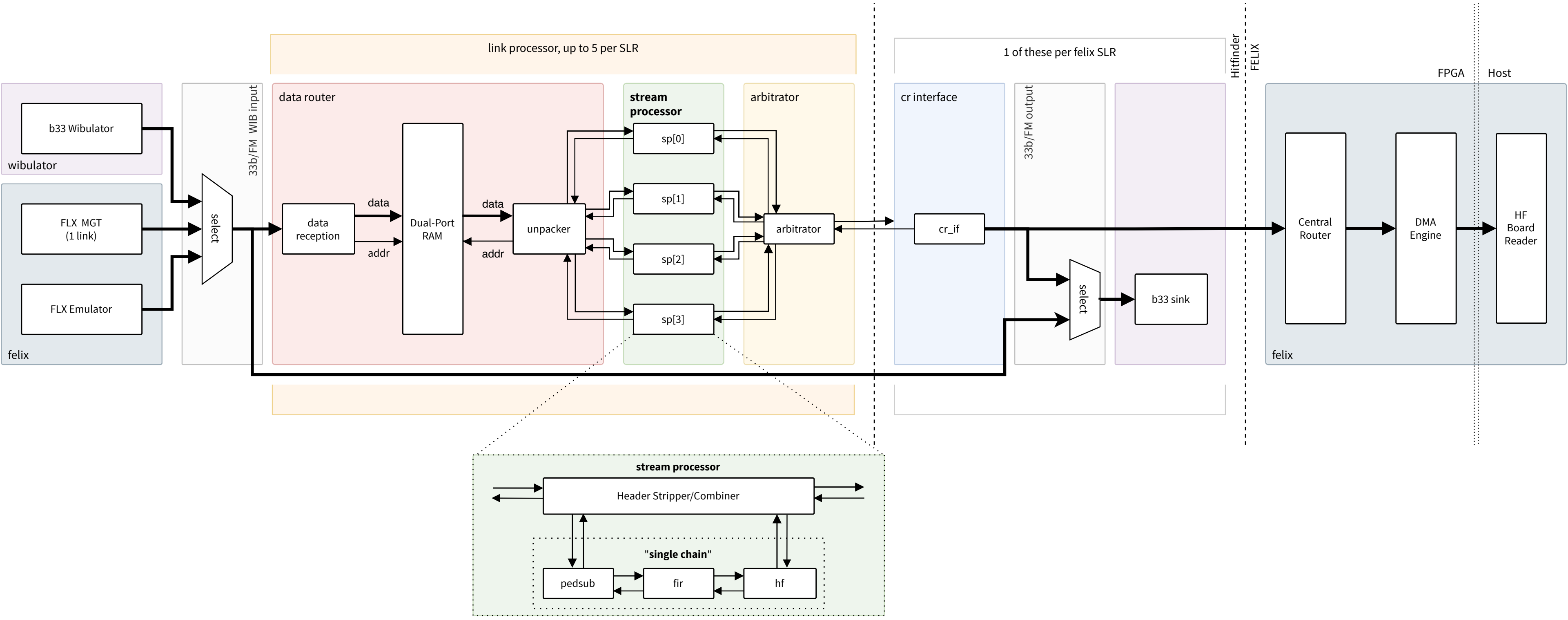
    Ans: FPGA resources covered in detail in previous section. Currently all optical links for each APA can be processed in a single FELIX-712 (KU115). Any future choice of FPGA that can accommodate required number of links for 1 APA, will most likely come with enough resources for a TPG fw implementation.

  - ***What are the future prospects for reducing resource use?***

    Ans: We will revisit our fw code and try to identify parts that could be optimized. Experience shows that resource utilization should be expected to rise, since new ideas for improving the HF approach will come and will be implemented.

Backup slides


SLR 1
felix
FLX MGT (1 link)
FLX MGT (1 link)
FLX MGT (1 link)
FLX MGT (1 link)
FLX MGT (1 link)
33b/FM WIB input
processors
stream processor
stream processor
stream processor
stream processor
stream processor
arbitrator
arbitrator
cr interface
cr_if
33b/FM output
CR1
FLXCORE
netio
FLX MGT (1 link)
FLX MGT (1 link)
FLX MGT (1 link)
FLX MGT (1 link)
FLX MGT (1 link)
33b/FM WIB input
processors
stream processor
stream processor
stream processor
stream processor
stream processor
arbitrator
arbitrator
cr interface
cr_if
33b/FM output
CR0
felix
SLR 0



link processor, up to 5 per SLR
1 of these per felix SLR
Hitfinder
FELIX
FPGA
Host
b33 Wibulator
wibulator
FLX MGT (1 link)
FLX Emulator
felix
33b/FM WIB input
select
data router
data reception
data
addr
Dual-Port RAM
data
addr
unpacker
stream processor
sp[0]
sp[1]
sp[2]
sp[3]
arbitrator
arbitrator
cr interface
cr_if
33b/FM output
select
b33 sink
Central Router
DMA Engine
HF Board Reader
felix
stream processor
Header Stripper/Combiner
"single chain"
pedsub
fir
hf