

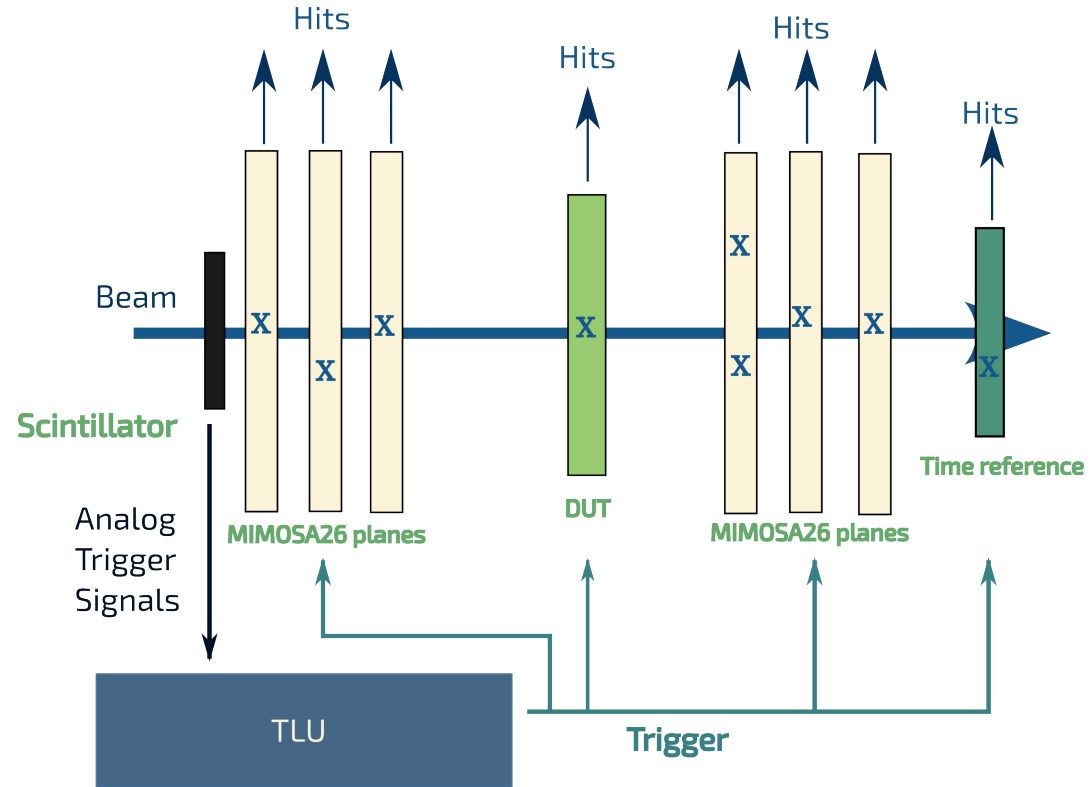
BTTB13 VALENCIA

A PYTHON-BASED CONTROL SOFTWARE FOR THE AIDA-2020 TRIGGER LOGIC UNIT AND TESTS IN HIGH-RATE TEST BEAM ENVIRONMENTS

Rasmus Partzsch, Christian Bepin, Yannick Dieter, Jochen Dingfelder, Fabian Hügging, Lars Schall

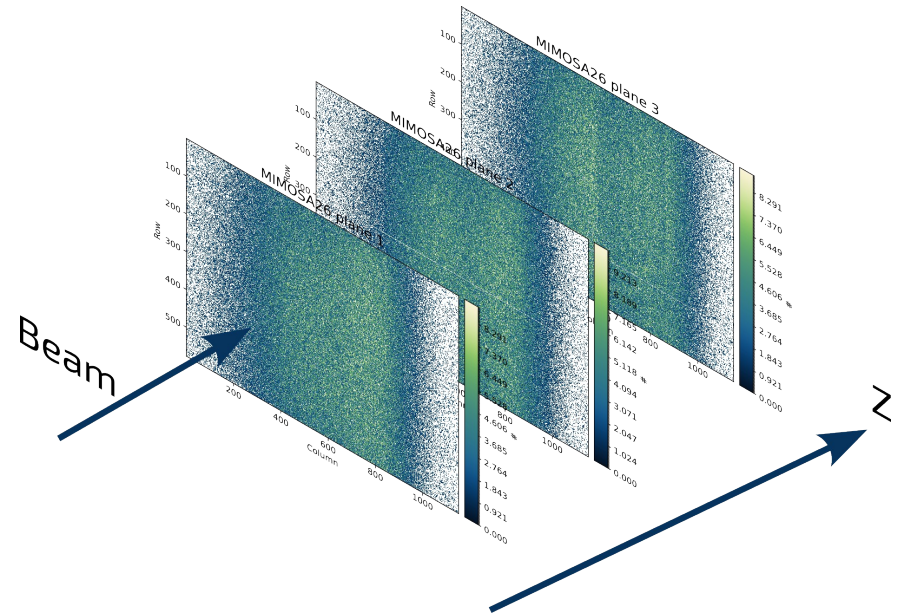
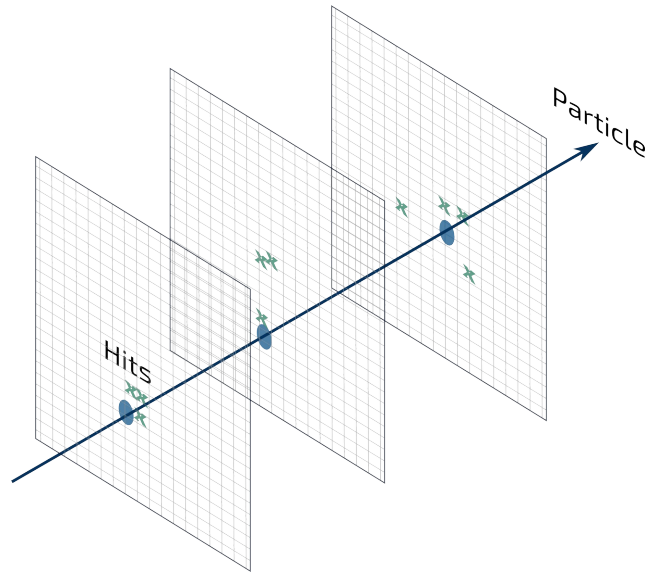
Introduction: A Test Beam Setup

- Example EUDET-type reference tracker
 - High spatial resolution planes for particle tracking (18 μm pitch)
 - Time reference plane $\mathcal{O}(\text{ns})$
 - Device under test (DUT)
 - Spatial, time resolution, efficiency...
 - Trigger logic unit (TLU)



Synchronization and Event building

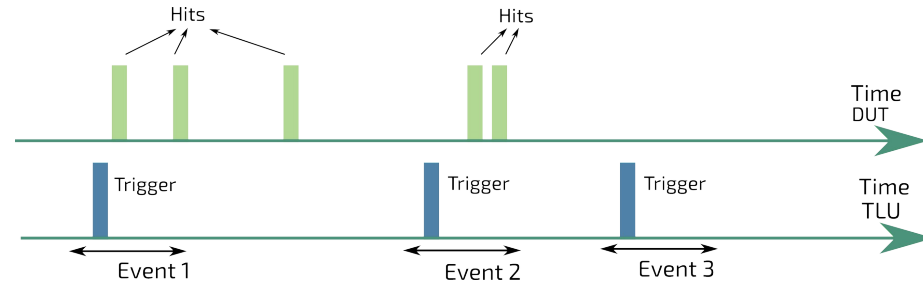
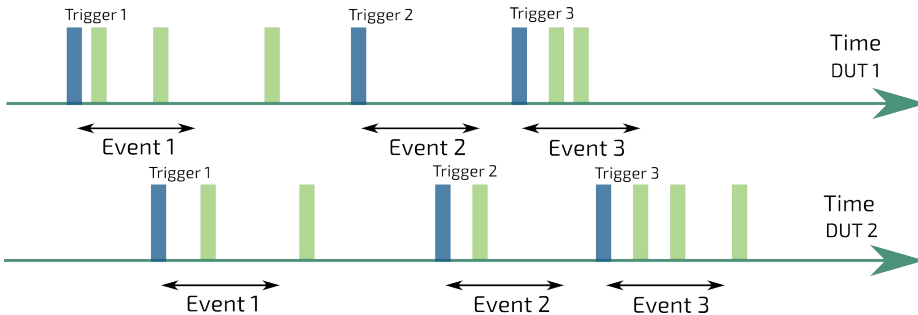
- Assignment of individual hits in one device to hits of another device



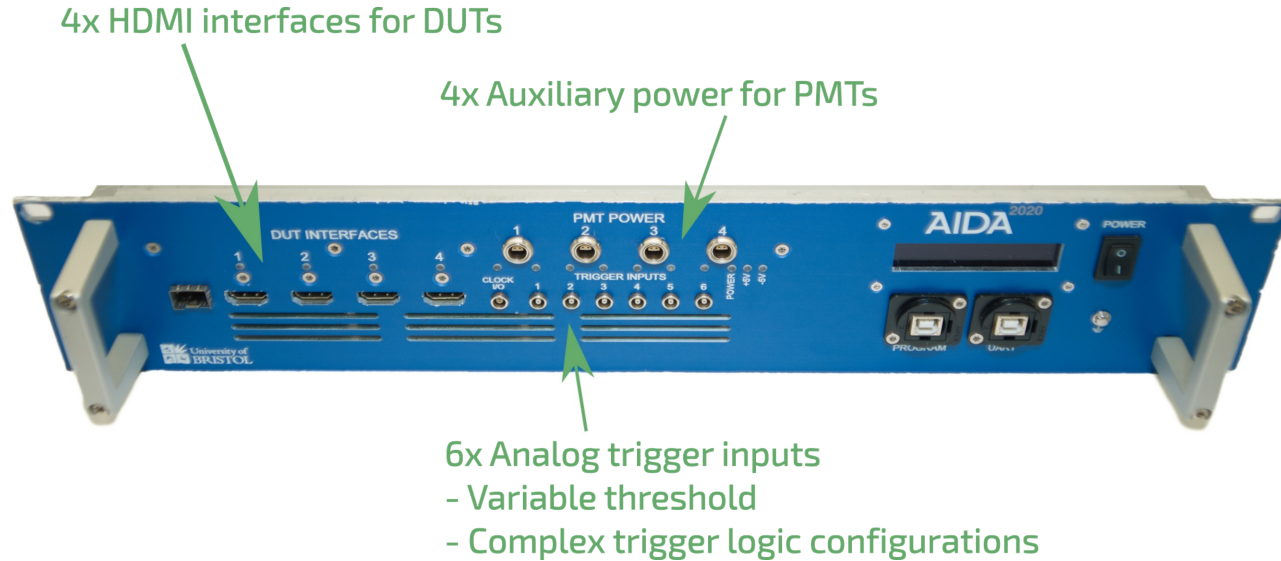
Synchronization and Event building

- Trigger number-based synchronization
 - Devices have individual time stamp
 - Assign hits to trigger number per device
 - Match devices by trigger number

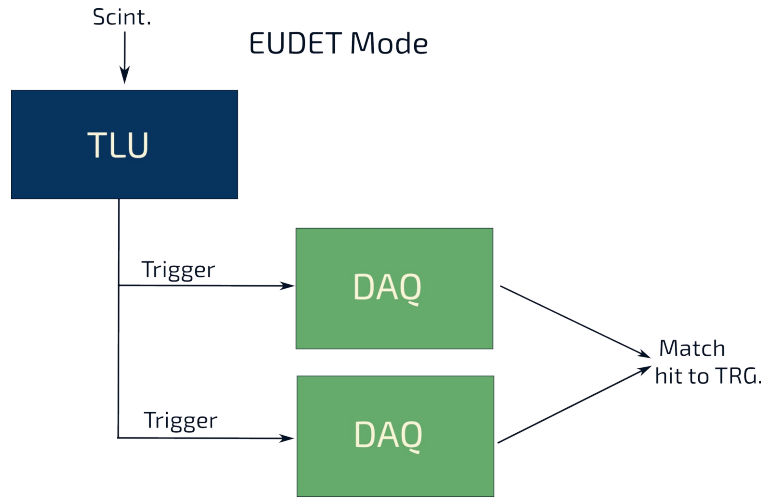
- Time-based synchronization
 - Devices have synchronous time stamp
 - Match hits by time stamp
 define event by time stamp and time window



The AIDA-2020 Trigger Logic Unit

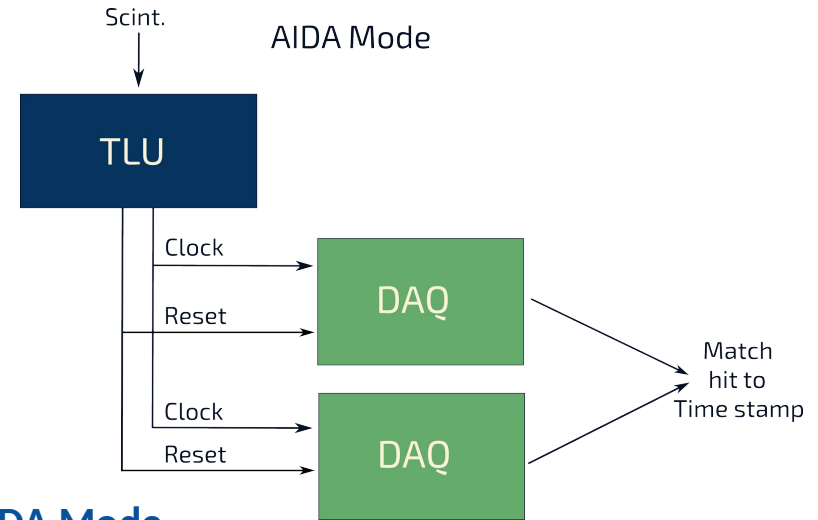


TLU: Operation with a DUT



EUDET Mode

- Synchronization between devices using trigger number which is sent in addition to trigger signal

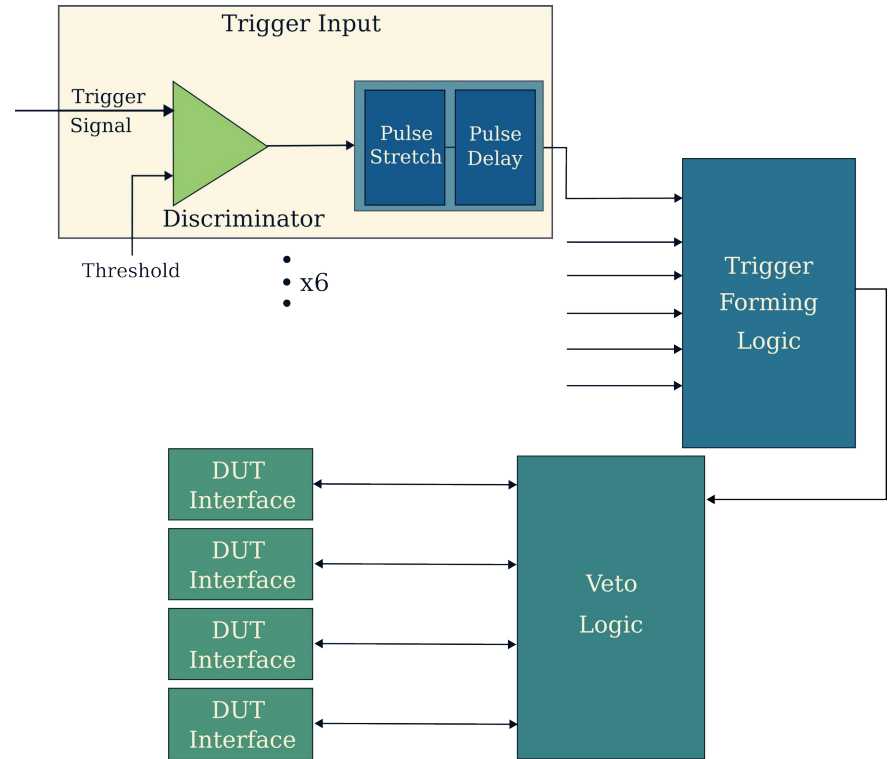


AIDA Mode

- TLU distributes common clock to DAQ and a reset for time stamp
- Synchronization of hit using time stamp

TLU: Trigger Processing

- 6 LEMO trigger inputs
 - Configurable threshold
 - Stretch & delay
- Trigger Forming Logic
 - Each trigger set as enabled or veto
 - Combinations of enabled or veto can lead to valid triggers
- Veto logic
 - DUT prevents TLU from issuing new trigger
- DUT interface
 - Trigger, busy, reset, clock



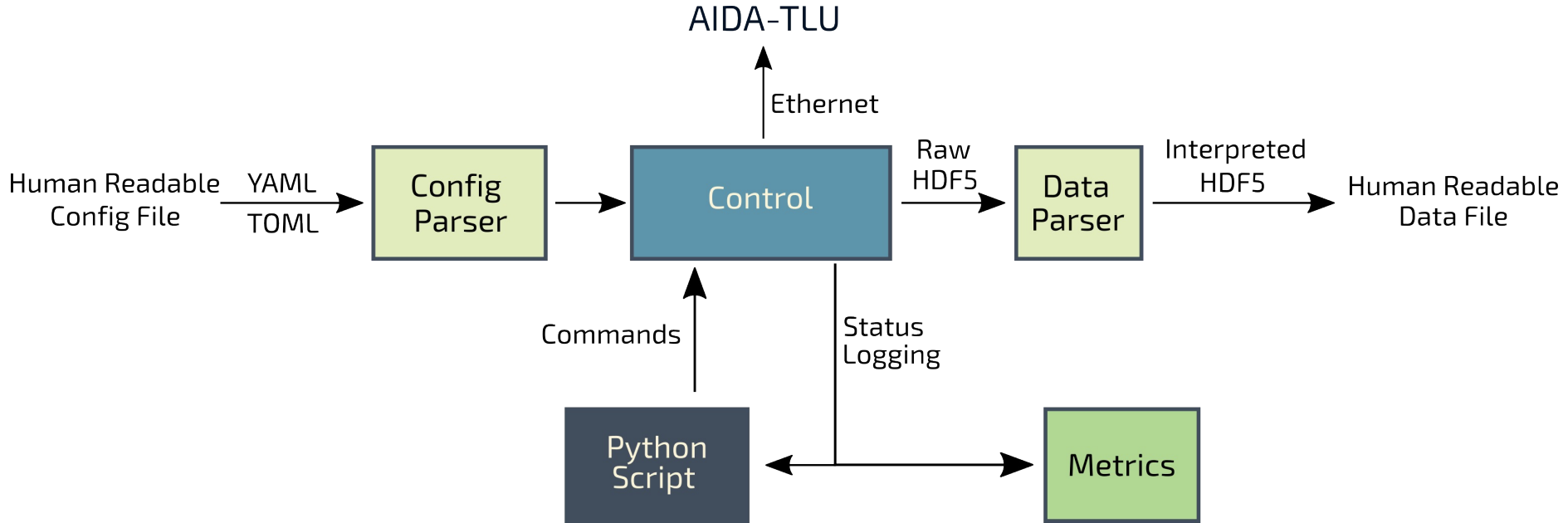
TLU: Trigger Logic

CH3	CH2	CH1	Pattern	Config. Word
0	0	0	0	0xE
0	0	1	1	
0	1	0	1	
0	1	1	1	
1	0	0	0	0x0
1	0	1	0	
1	1	0	0	
1	1	1	0	

- Example with three trigger inputs
- (CH1 or CH2) and \neg CH3
→ Config. Word: 0x0E

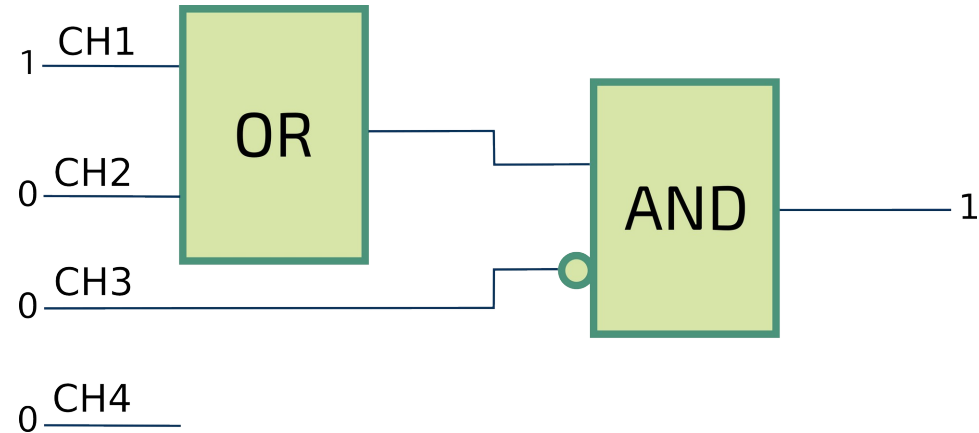
- Setting trigger configuration
 - Set a 1 in the pattern row for all valid trigger patterns
 - Configuration word is the hexadecimal representation of the valid trigger patterns.
- 6 trigger inputs each in a binary state (HIGH or LOW)
 - $2^6 = 64$ possible trigger patterns

Python based Control Software



Python Software: Trigger Logic

- Configuration statement is evaluated using the Python logic operators:
- (CH1 or CH2) and (not CH3)
- Software translates statement into correct configuration word by testing all trigger patterns against statement



- Hierarchical Data Format (HDF5)
 - Compression filters
- Widely used in scientific research and industry
- Existing API for C, C++, Python, Java, R, Julia...
- GUI: ViTables

	eventnumber	timestamp	overflow	eventtype	input1	input2	input3	input4	input5	input6	sc1	sc2	sc3	sc4	sc5	sc6
0	1	79629	0	1	True	True	False	False	False	False	115	114	160	242	241	248
1	2	163747	0	1	True	True	False	False	False	False	43	42	160	242	241	248
2	3	247865	0	1	True	True	False	False	False	False	238	237	160	242	241	248
3	4	251870	0	1	True	True	False	False	False	False	160	159	160	242	241	248
4	5	360022	0	1	True	True	False	False	False	False	148	147	160	242	241	248

<https://www.hdfgroup.org/solutions/hdf5/>

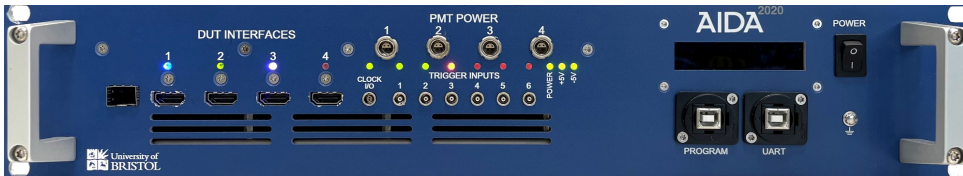
Usage Example: Constellation



- Implemented satellite
- Metrics for live visualization of trigger rates...
- Logging of configuration and status messages
- Easy configuration using TOML file format
 - LED feedback of configuration

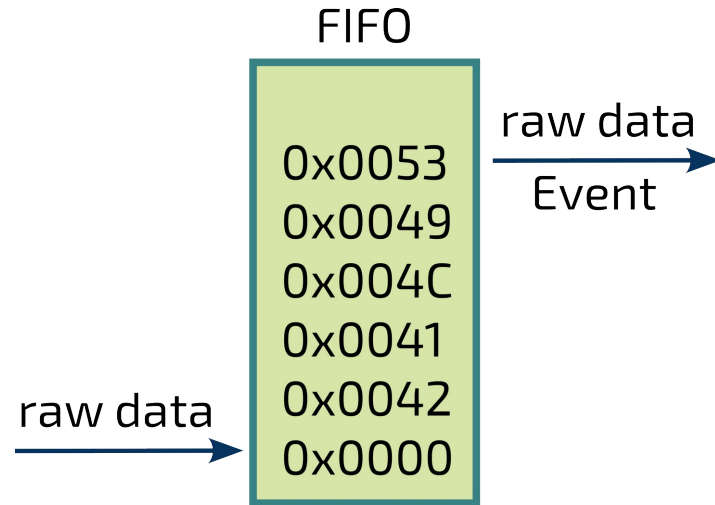
```

1
2 [satellites.AidaTLU]
3
4 internal_trigger_rate = 0
5 dut_interfaces = ['aida', 'eudet', 'aidatrig', 'off']
6
7 trigger_threshold = [-0.1, -0.1, -0.1, -0.1, -0.1, -0.1]
8 trigger_inputs_logic = '(CH1 or CH2) and (not CH3)'
9 trigger_polarity = 'falling'
10 trigger_signal_stretch = [2, 2, 2, 2, 2, 2]
11 trigger_signal_delay = [0, 0, 0, 0, 0, 0]
12
13 enable_clock_lemo_output = true
14 pmt_power = [0.8, 0.8, 0.0, 0.0]
15 output_data_path = ''
16
    
```



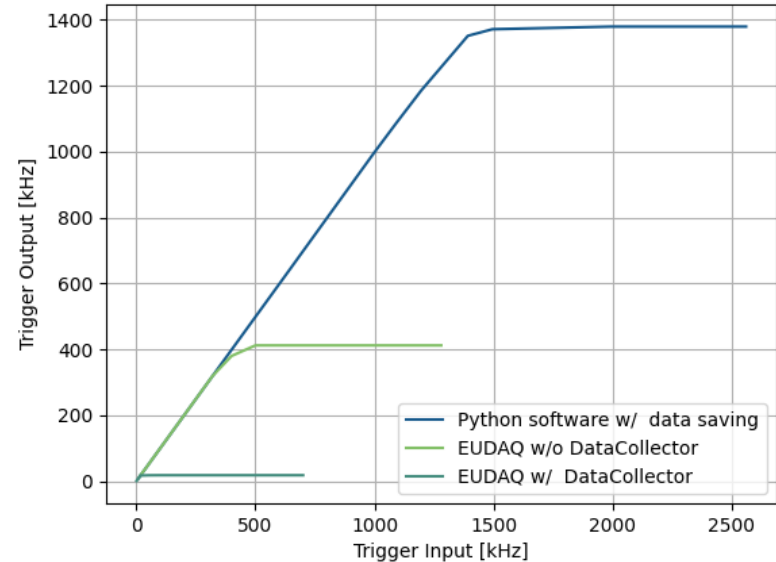
Python Software: Performance at High Rates

- Benchmark tests for high rate
- TLU stores trigger data in 32-bit words in FIFO
 - 6 words for one trigger
 - Last word (trailer) expected to be 0s
- Python software pulls FIFO continuously
 - Saves data in a raw data file
 - Raw data file is interpreted offline
- Comparison: EUDAQ interprets data live



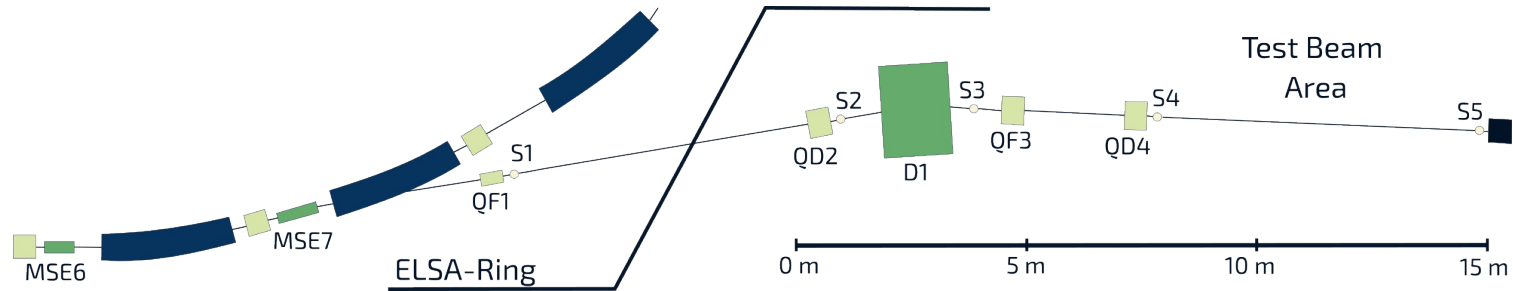
Comparison EUDAQ – Python AIDA-TLU

- Tested standalone software performances using pulse generator
- EUDAQ DataCollector or LogCollector reduce trigger output rate
- Python software saturates at ~1.3 MHz

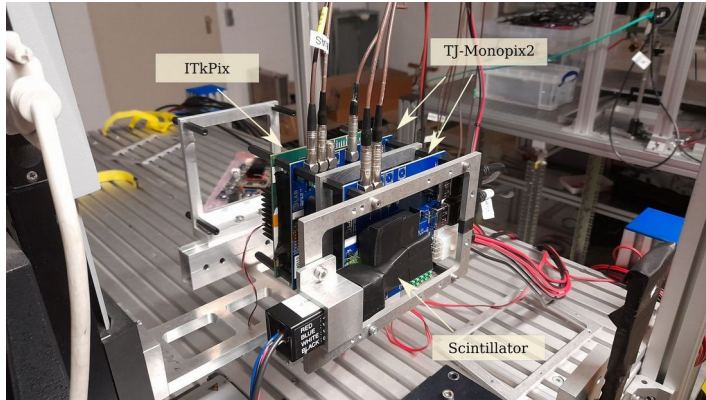


High Rate Tests at the ELSA Test Beam Area

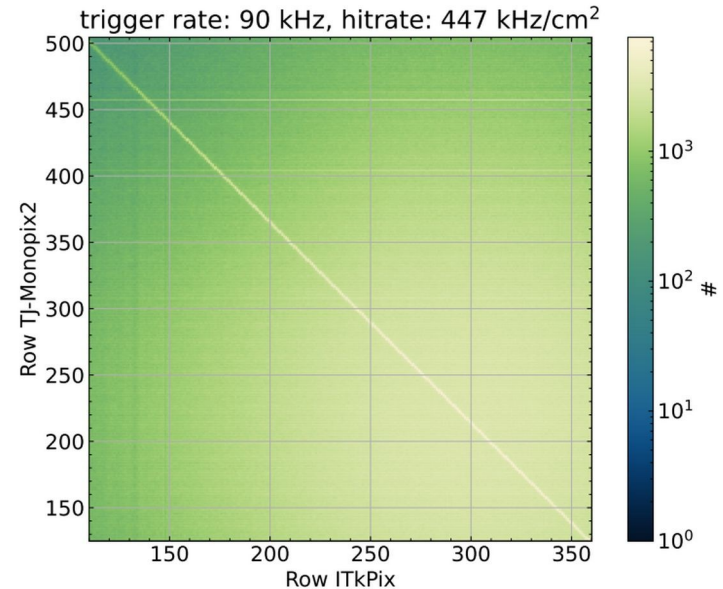
- Electron accelerator at the University of Bonn
- Primary electron beam
 - Rates ($\mathcal{O}(\text{Hz})$ - $\mathcal{O}(100 \text{ MHz})$)
 - Unique tests for software, hardware, readout...



High Rate Tests at the ELSA Test Beam Area



- Absolute hit rate
 - Calculated from total amount of measured hits during the measurement run
- Specific readout configuration limits output trigger rate to 90 kHz.
 - Leads to high track multiplicity



Conclusion

- Successful implementation of standalone Python-based control software for the AIDA-TLU
- Usable with Constellation framework
- Software tested in multiple test beams
 - AIDA-TLU software tested at high trigger rates (90 kHz) at the ELSA test beam facility

Outlook

- Software development finished
- Request of features or bug reports in Github
- Tests at even higher particle rates (1 MHz) expected this week



<https://github.com/SiLab-Bonn/aidatlu>

Thank you for your attention!

The measurements leading to these results have been performed at the Test Beam Facility at DESY Hamburg (Germany), a member of the Helmholtz Association (HGF)