Developments of an FPGA based track reconstruction pipeline for the ATLAS event filter

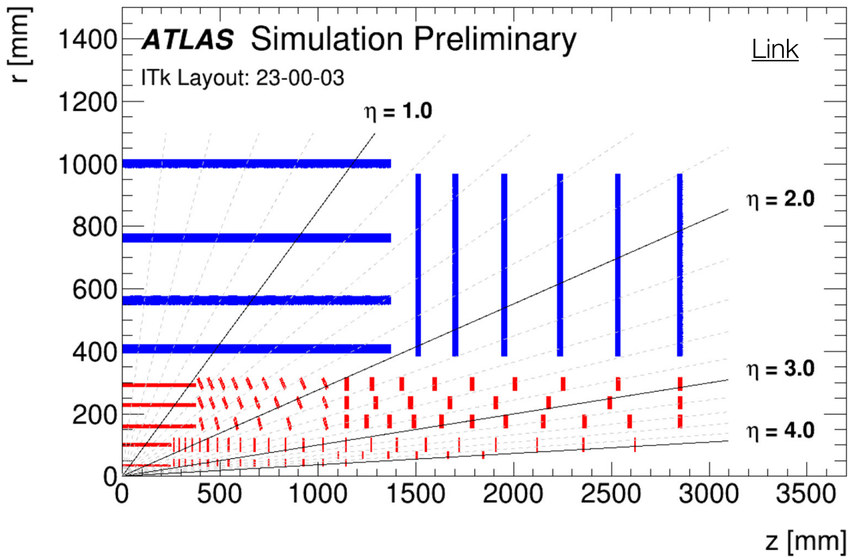
Haider Abidi
On behalf of the ATLAS TDAQ collaboration
Oct 24, 2024

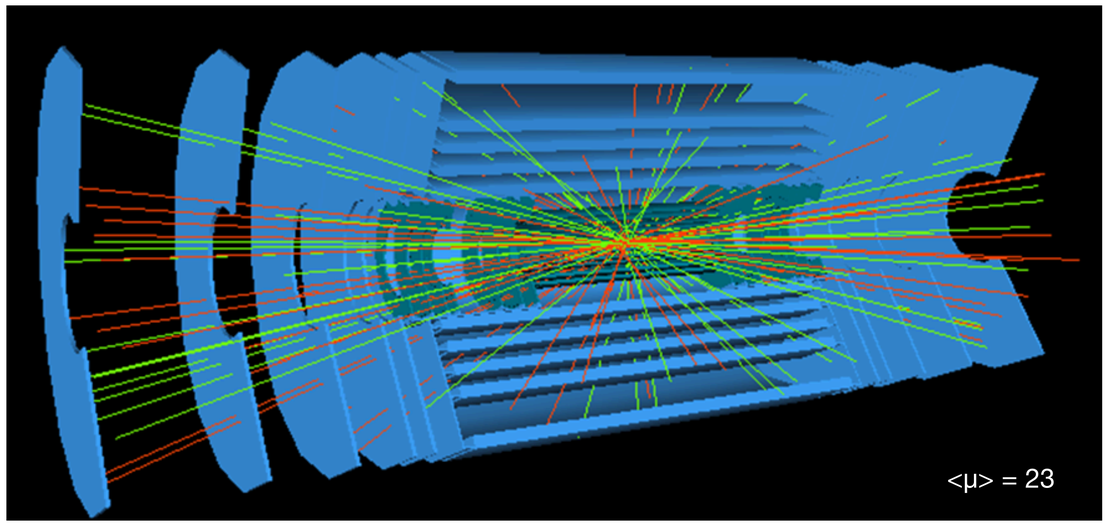
ATLAS EXPERIMENT

Brookhaven National Laboratory

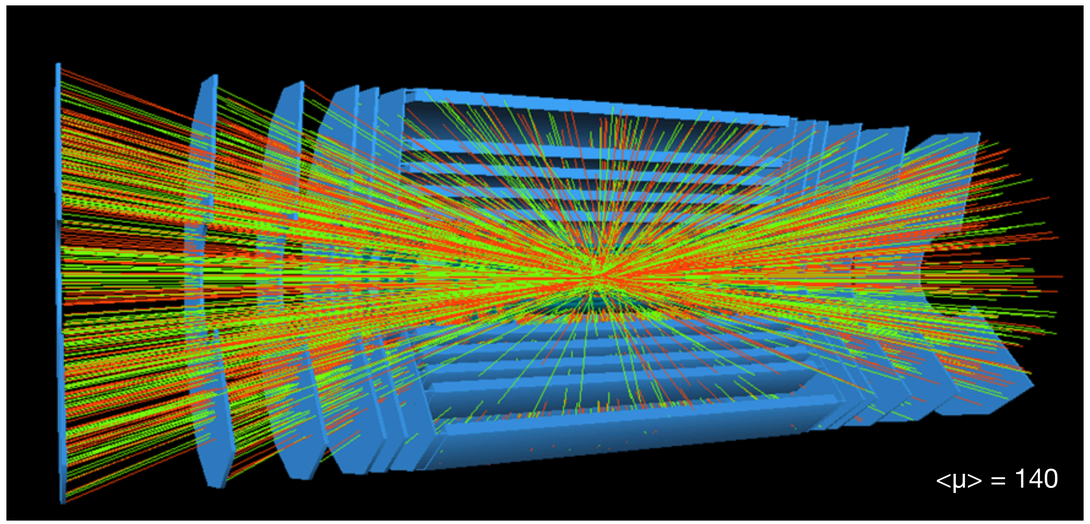
ATLAS & HL-LHC upgrade

- LHC and ATLAS are constructing upgrades to continue ensure physics reach
  - New conditions are challenging - an average of 200 simultaneous collisions (pile-up) per bunch crossing are expected
- New detectors, such as new inner tracking detector to allow high-precision reconstruction of charged particle tracks (ITk), are being constructed
- The ATLAS trigger system will upgraded to cope with these conditions
  - Tracking at trigger level is essential to control rates while maintaining good efficiency for relevant physics processes





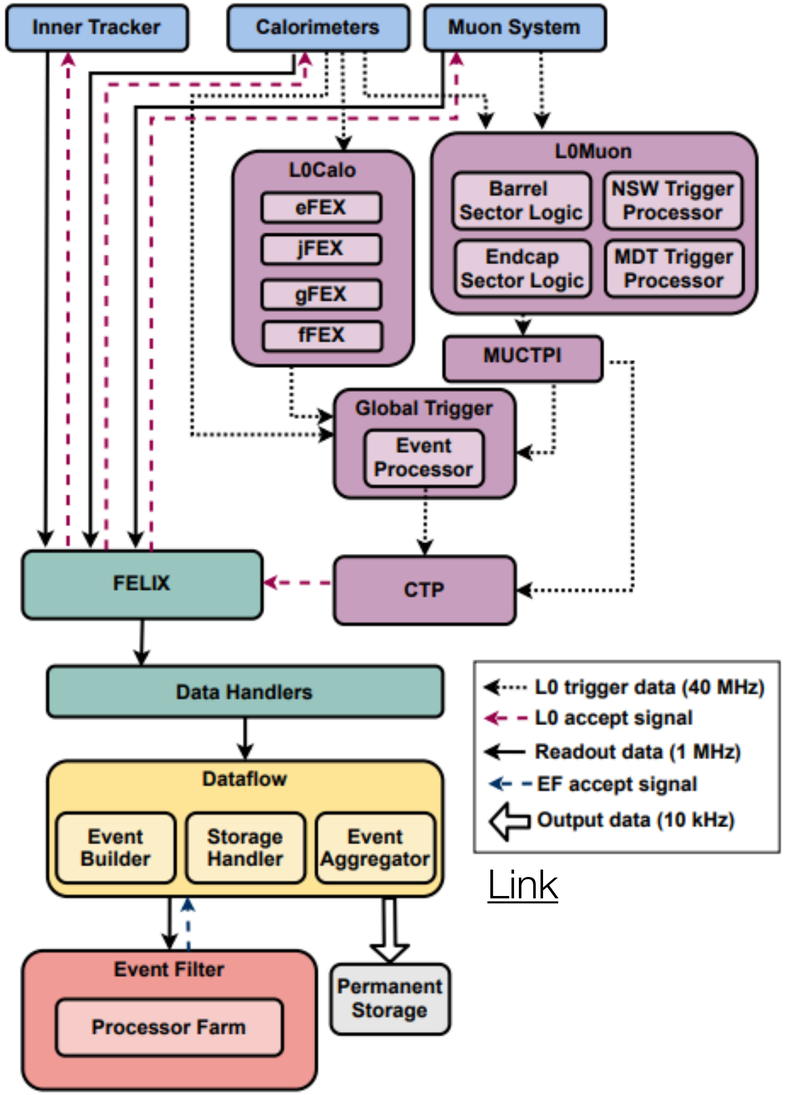
Current environment inside ATLAS at LHC



Expected environment inside ATLAS at HL-LHC

ATLAS TDAQ System for HL-LHC

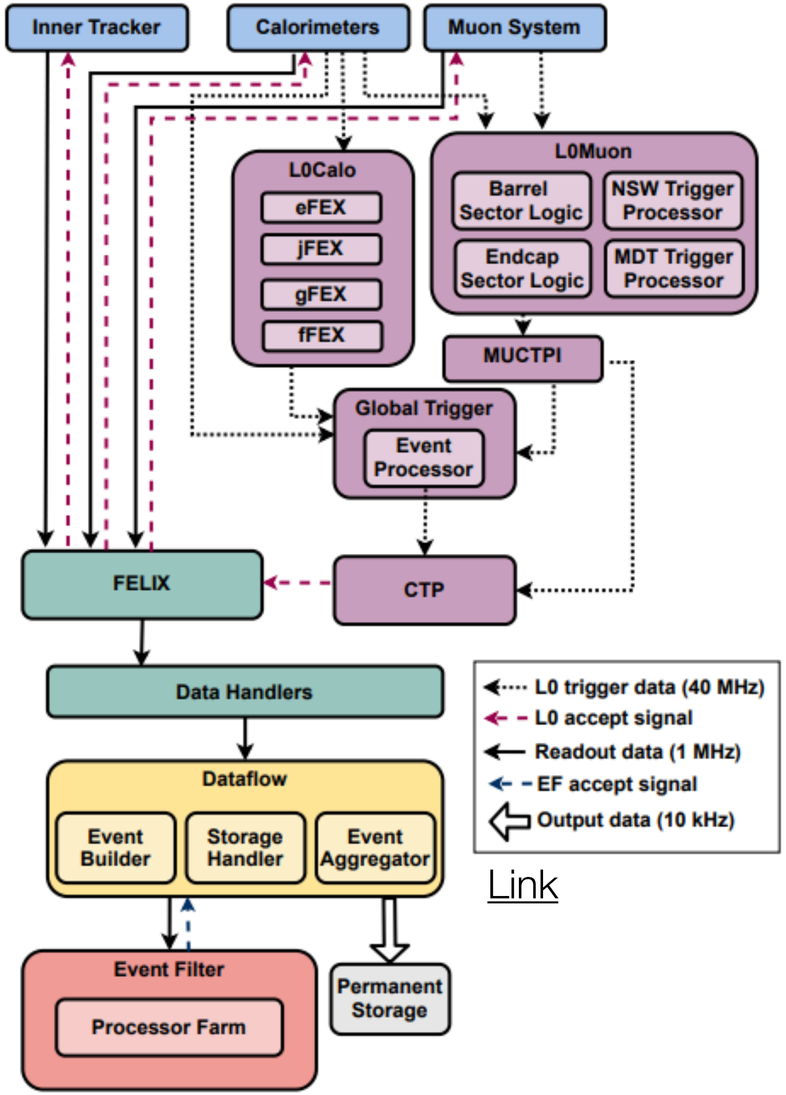
- Two tiered system for triggering
  - Level-0 - Hardware based trigger to reduce rate from 40 MHz to 1 MHz
  - Event filter system - higher level triggering system with “precision” reconstruction algorithm to reduce the rate to 10kHz
- Event filter system will perform track reconstruction for various selected physics processes:
  - Perform regional tracking in Regions of Interest defined based on objects identified at Level-0
    - Used to verify the presence of high-pT tracks in single lepton triggers and to associate objects to a common vertex
  - Run full-scan tracking at a reduced rate after the regional tracking for jets and Et miss signatures
  - Large radius tracking for exotics signal
    - Focuses on tracks with high impact parameters like those resulting from the decays of Long Lived Particles



Link

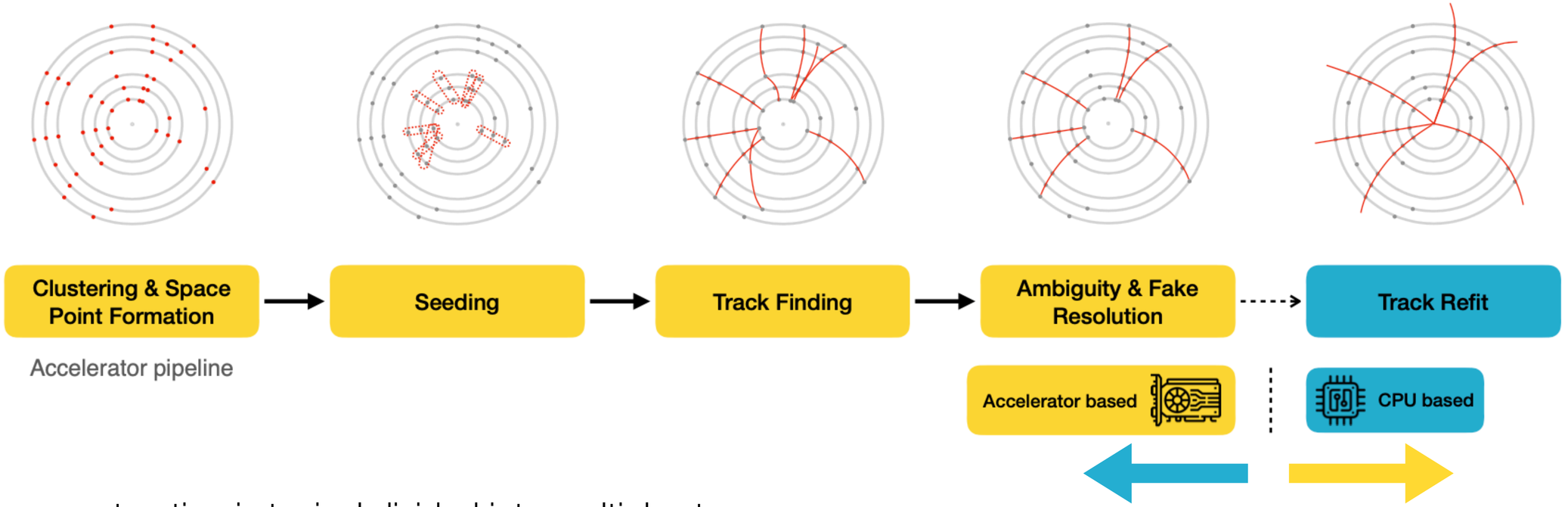
EF Tracking System

- Originally assumed a hardware-based track reconstruction based on associative memory ASICs and FPGAs
  - Lower requirements on input rate (4MHz→1MHz)
  - Software tracking improvements
  - The rise of commercial accelerator cards
- Revisited the solution for EF Tracking (Link)- **"ATLAS commit to a commercial solution for EF Tracking at HL-LHC,"**
  - Heterogeneous devices (e.g., GPUs and FPGAs) allow the CPU to offload specialized tasks, and may provide **power saving** and/or **throughput increase**
  - **Changed planed with respect to the original assumption**
- Exploring CPU, GPU and FPGA based solutions - Technology choice in 2025
- **Focus on CPU + FPGA based solutions for this talk**
  - **Emphasis on the FPGA implementation compared to algorithmic details**



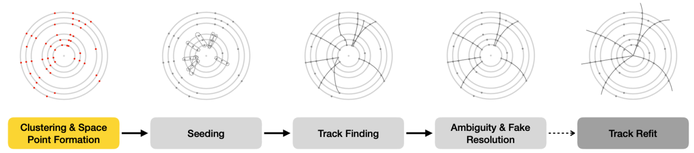
Link

Tracking On FPGA in a Nutshell

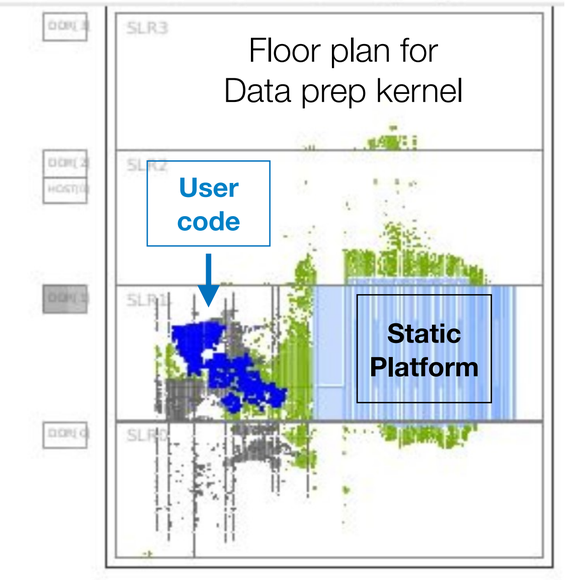


- Track reconstruction is typical divided into multiple steps
  - Certain steps are more efficient on CPUs than FPGA
    - Example: A full precision track fit using Kalman filter is almost impossible to port to FPGA to the large memory requirement
  - However, data transfer between CPU and FPGA card has a latency overhead
- Leads to a situation where the boundary between CPU and FPGA has to be optimized for best physics performance
- **Focus on one example solution for each step - final pipeline still to be decided**

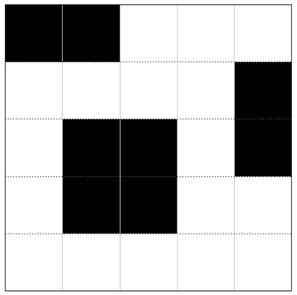
Data Preparation



- Data from the detector has to be processed to create the “clusters” which are used for tracking
  - Multi-step process with simple algorithms that can take advantage of FPGAs
- Each step is implemented as separate a FPGA kernel within the Xilinx Vitis workflow
  - Performance is individually tested through test vector created from full physics simulation
  - Successfully compiled into a single FPGA binary with strict requirements for interfacing
- Limited resources required for these steps

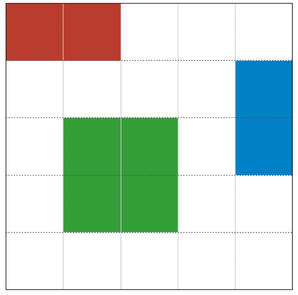


Raw detector information



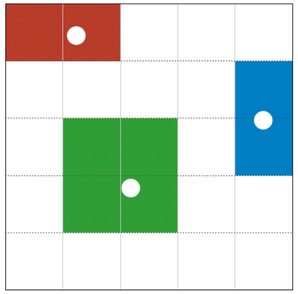
Unpack detector data to identify which sensors have been fired

Clustering



Combine neighbouring sensors to form cluster

Local to global



Compute centroid and provide a global position of the cluster

Rest of FPGA pipeline

Kernel Route Utilization

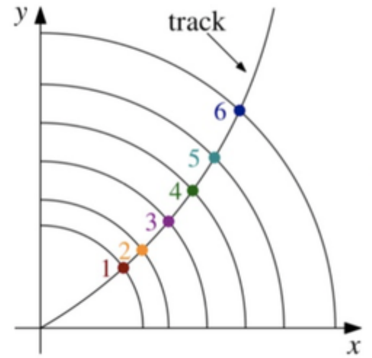
Name	LUT	LUTAsMem	REG	BRAM	URAM	DSP
Platform	90334	8640	156830	215	0	4
User Budget	1635874	781552	3295586	2473	1280	12284
Used Resources	31648	2252	31326	496	0	32
Unused Resources	1604226	779300	3264260	1977	1280	12252
Clustering (Pixel)	5068	46	1451	0	0	0
	5068	46	1451	0	0	0
Local to global (Pixel)	4650	434	5076	154	0	1
	4650	434	5076	154	0	1
Local to global (Strip)	6696	402	6214	288	0	2
	6696	402	6214	288	0	2
Clustering (Strip)	8762	672	10337	15	0	0
	8762	672	10337	15	0	0

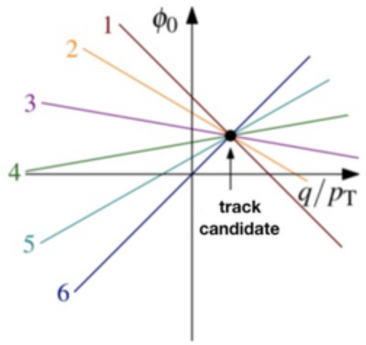
Resource usage in Alveo U250

Pattern Recognition

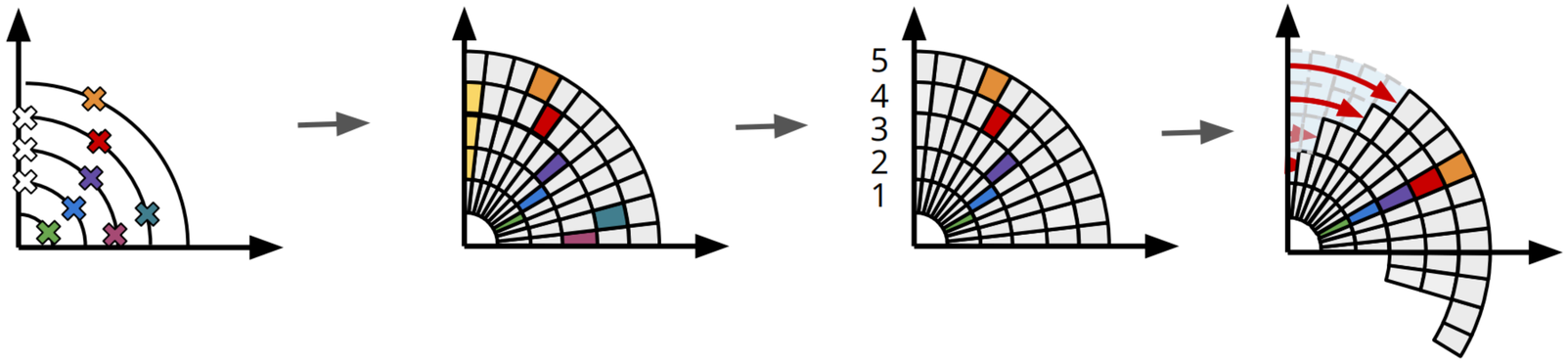
- Raw clusters need to be grouped into track candidates
  - Extremely complex problem due to combinatorial growth
- Classical solution such as Kalman Filter are not easily portable to FPGA
- Conformal transforms (eg Hough transform) are simpler algorithms that provides the same output
  - Optimized for FPGA implementation by using a shift-register inspired implementation
  - One of the different potential ideas being explored

Hough Transform -
group track candidate based on a conformal transform





1D bitshift Algorithm



Get Clusters

Fill “bins” in memory

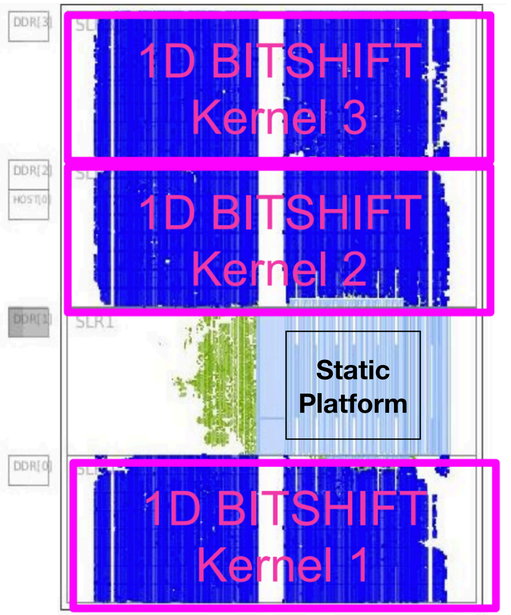
Retrieve results for a slice

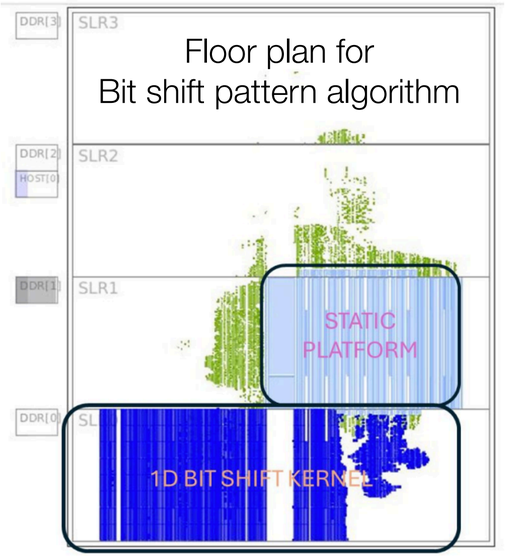
Shift operation for track candidates

Pattern Recognition

- Implementation has been refined from previous implementation
  - Designed to fit within one super logic region in a FPGA
  - Similar testing procedure with testbeds using full physics simulation
- Multiple copies of the same kernel for parallelization have been tested and gives similar results as single one
  - One static platform for communication with the memory blocks

Fitting multiple kernels on one FPGA





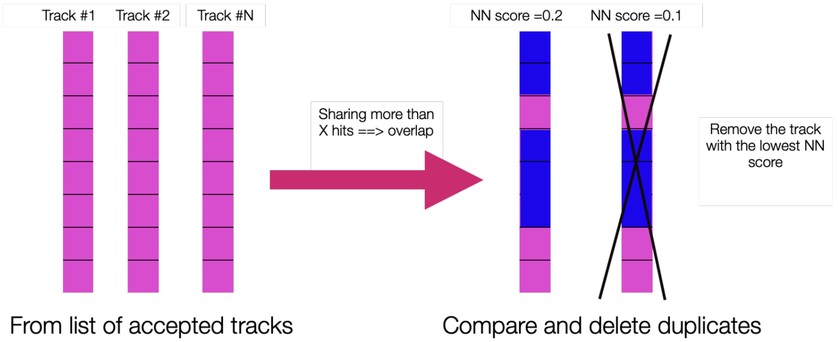
Refinement of algorithm for optimized physics performance

Source	LUT(%)	Flip-Flops(FF%)	BRAM/URAM (%)	DSP(%)
ATLAS-TDR-029-ADD-1	12	7	27	1
Current Work	21	10	14	1

ML Algorithm for Tracking

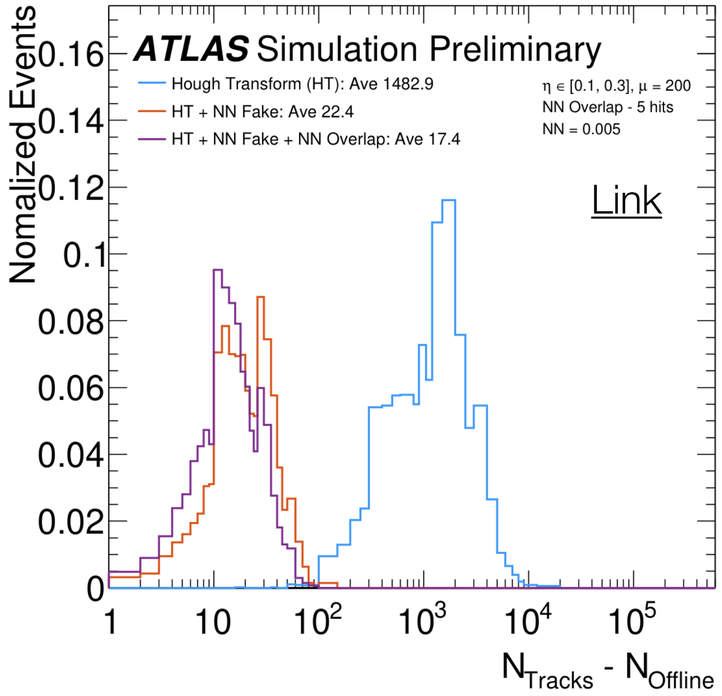
- Conformal transform is a relative simple/cheap pattern recognition algorithm
  - Cost: Lot of fake hit combinations & No figure of merit on fit quality
- Need to preform a preliminary “Ambiguity resolution” without using the time consuming fit for each track
  - Leverage the performance of ML to predict this figure of merit?
- Classify a vector of x/y/z position coordinates as coming from a 'true or fake track’
- Large reduction in the fake tracks with relatively high efficiency!
  - Reject tracks with low score
  - This AI/ML idea allows conformal transform algorithm to fit within the data bandwidth requirements and enable this pipeline for FPGAs

Fake rejection algorithm



Limited resources used by the NN

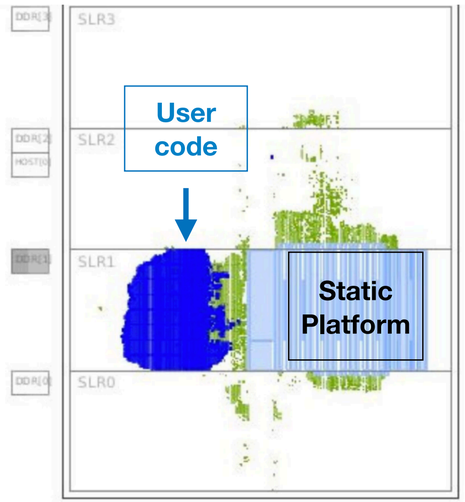
Name	BRAM_18K	DSP48E	FF	LUT	URAM
Total	1	3589	42923	140591	0
Available	5376	12288	3456000	1728000	1280
Utilization (%)	~0	29	1	8	0



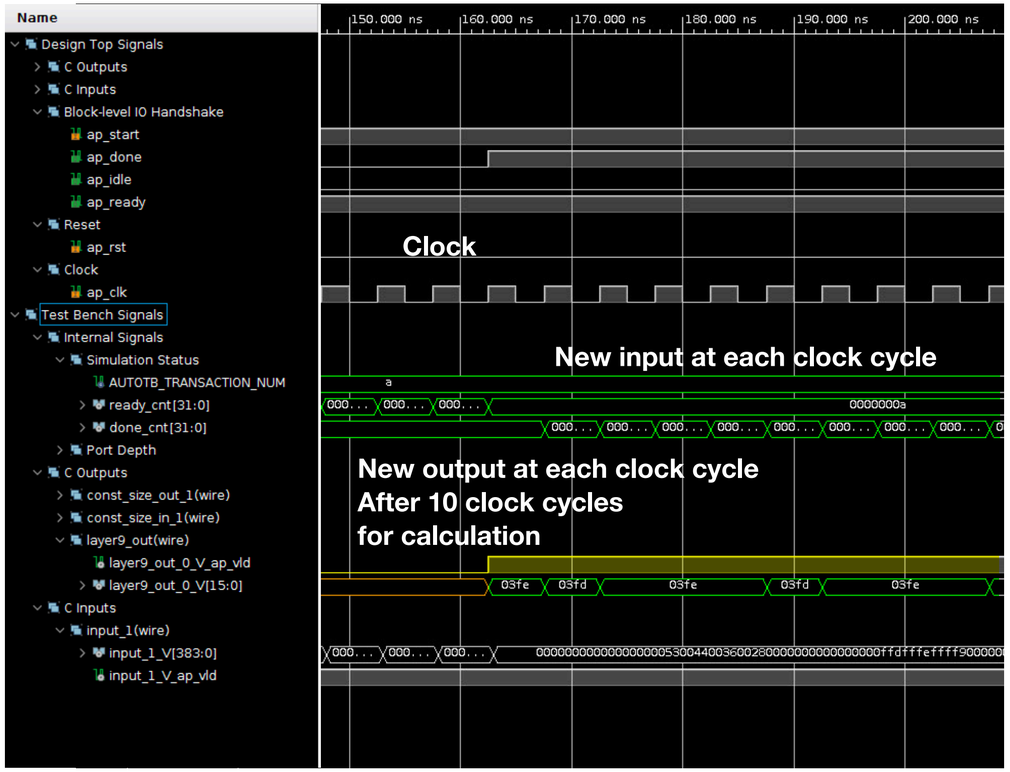
AI/ML on accelerators

- Use HLS4ML to convert the ML algorithm for FPGA implementation
  - Understanding limitation, creating implementation, validating inference calls
  - Establishing pruning/quantization strategies for efficient implementation
- For FPGAs, the Vitis kernel flow has been established & validated for NNs
  - Matrix multiplication is perfectly pipelined
  - For N evaluations, total latency is O(N + constant), not O(constant*N)
- Overall footprint of this ML algorithm with the overlap removal on FPGA is small

Floor plan of the overlap removal code



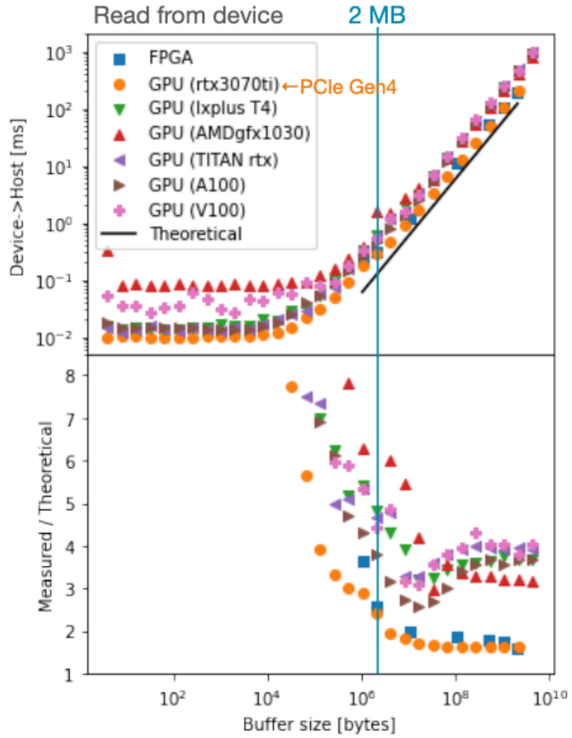
Validation of NN on FPGA



Integration in Athena

- Pipelines are required to start/end in Athena, the ATLAS software base which control the overall algorithm flow
  - All algorithms need ‘plumbing’ to make this happen - common for any accelerator choice
- Accelerators require complex data transfer and EDM management
  - FPGAs communication have been established and validated in Athena
  - Work ongoing to characterize data transfer rates in realistic conditions
- Accelerator algorithm need to interface to CPU based Kalman fitter for precision fits
  - Accomplished through developing interface **in collaboration with CPU developers**
  - On going work to convert FPGA data formats to CPU formats

Data Transfer latency



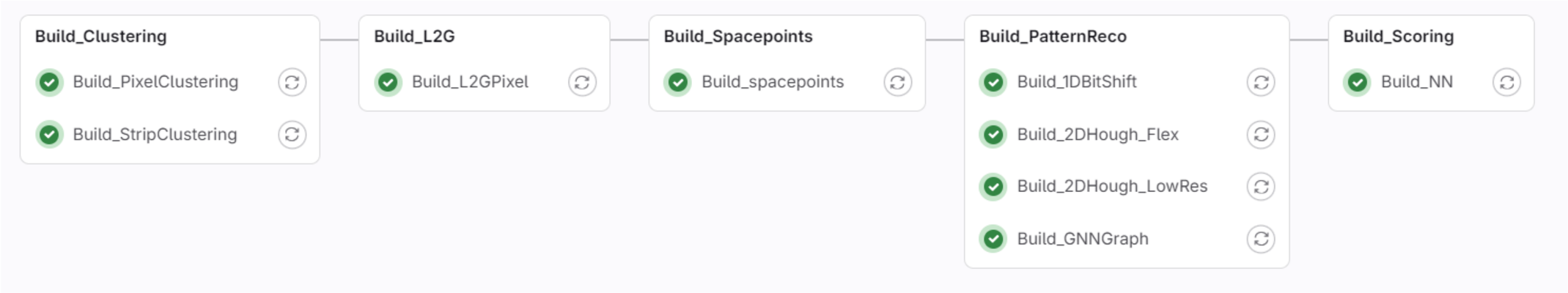
Generic CPU interface for all accelerator based algorithms

```
/// @brief small non-persistent data class to wrap the output
/// of the EF-tracking development pattern finding placeholder
You, 6 days ago | 1 author (You)
namespace ActsTrk{
  You, 6 days ago | 1 author (You)
  struct ActsEFProtoTrack{
    /// set of measurements assigned to the same pattern
    std::vector<ActsTrk::ATLASUncalibSourceLink> measurements = {};
    /// estimate of initial track parameters for this pattern
    std::unique_ptr<Acts::BoundTrackParameters> parameters = nullptr;
  };
};
```

Integration and Testing

- Integrating FGPA development is complex - large dependancy on exact software versions and data transfer/format between kernels need to be agreed upon
  - Additionally, stable physics performance is required as the system gets closer to deployment

- Developed a full gitlab based CI pipeline to ensure that each kernel is compiled and linked with the same software version
  - Ongoing work to use FPGA-based testbed to run full hardware kernel and monitor output & performance through development cycle
  - Updating to use the latest Vitis version



CI pipeline for various hardware kernels

Conclusions

- Ongoing redesign of the EF Tracking for Phase-II upgrade of ATLAS
  - Commodity systems based on CPUs/GPUs/FPGAs identified as viable solutions in the amendment of the TDAQ technical design report
- Developing tracking algorithms for trigger on accelerator requires innovation and adaptation of current paradigms
  - Initial demonstrator with hardware implementation has been developed and is being integrated together
- The next year will be used to optimize the performance and work towards the technology choice for next year