Real-Time Track Reconstruction with FPGAs for MUonE

Michael McGinnis, David Monk, Benjamin Lawrence-Sanderson, Kristian Hahn

25.8.28

Supported by NSF award PHY-2111556

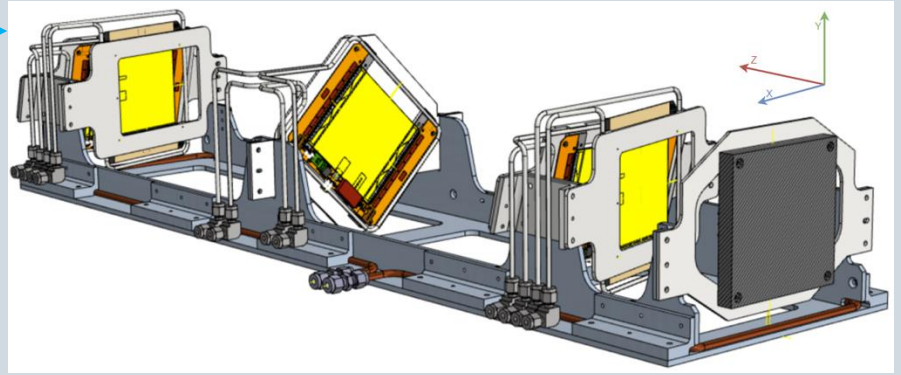
The MUonE Experiment in Brief

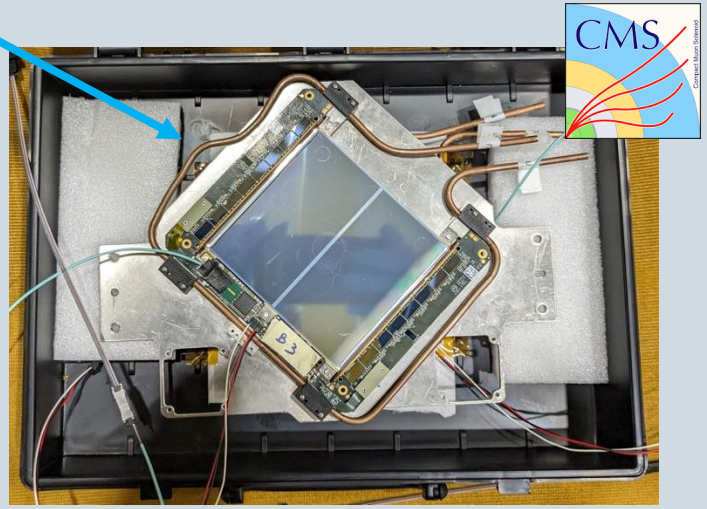
- Largest uncertainty in Muon g-2 results is the calculation of the LO hadronic vacuum polarization
- MUonE aims to extract this by **measuring the the elastic scatter of muons on atomic electrons**
  - Requires reconstructing tracks and vertices to extract scattering angles
- M2 beam at CERN's north area
  - Asynchronous
  - 160 GeV μ
  - Maximum intensity of 2×10^8 μ per spill (~50×10^6 μ /s)

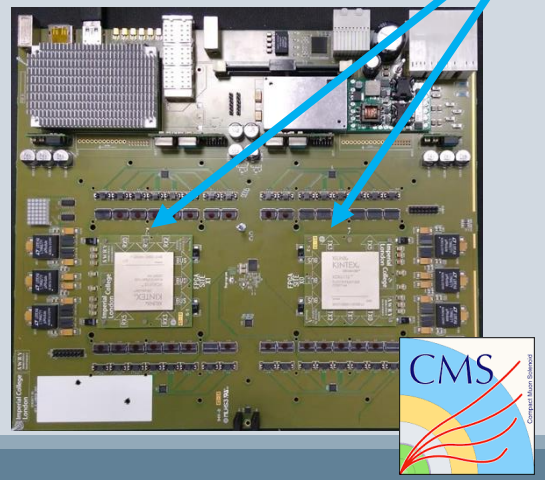
The MUonE Website

Tracking Hardware

- MUonE is composed of stations:
  - 6 strip tracker modules in pairs: XY UV XY
  - UV frames rotated 45°: measurements combined to single hit to rotate to global frame
  - Scattering target in front
- 2S modules (see many other talks and posters)
  - Readout at 40 MHz
- Serenity ATCA board: houses FPGAs for readout and configuring modules



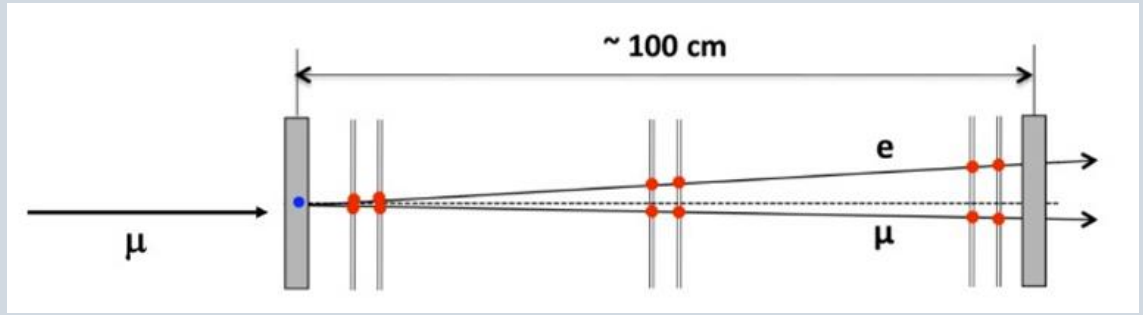




Online Reconstruction

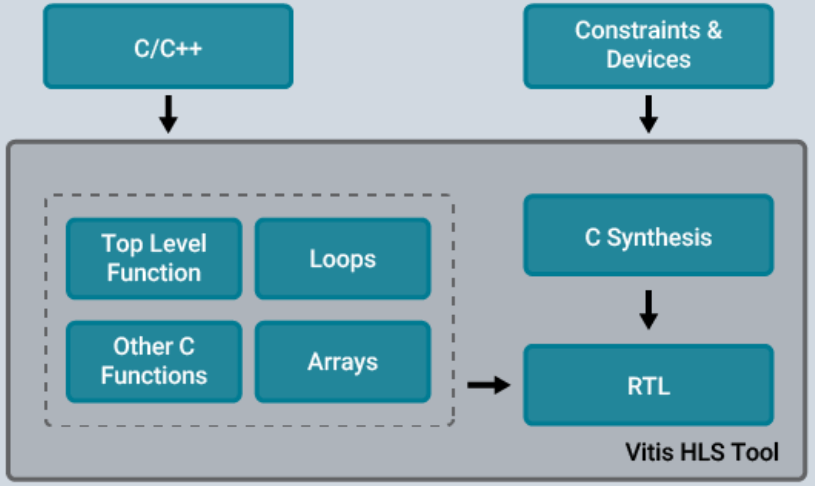
Is it feasible to do online track fitting and vertexing to reconstruct and select for μe scatters?

- At a rate similar to readout
- On FPGA, tested parallel to mainline DAQ
- **Tracking and vertexing apt to select μe scatters**
  - Use number of tracks, vertex position, reconstructed angle
  - EX: signal vs pair production
- Event selection is necessary
  - Tracker stations estimated to produce ~600 Gb/s (16 b/hit, 3 hits/module, 240 modules, 40 MHz readout, with padding)



High-Level Synthesis (HLS)

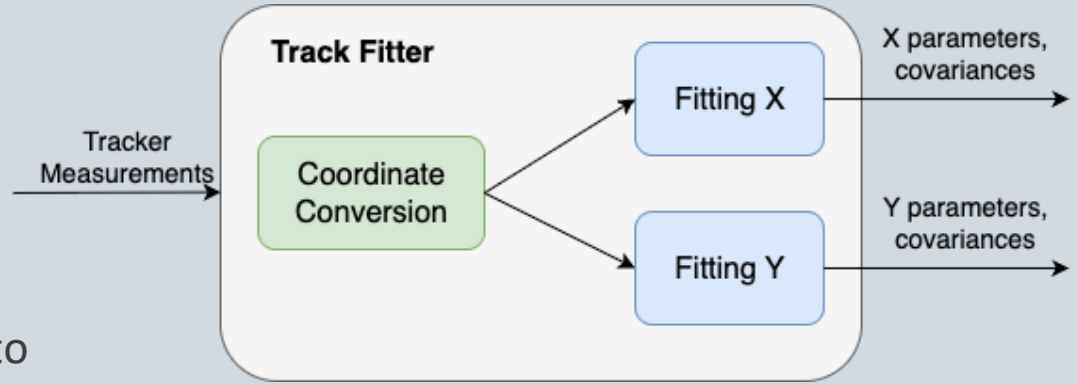
- HLS allows for fast development of firmware
  - Synthesizes C/C++ into an HDL
  - Verifies software and firmware with a test bench
  - Estimates performance and resource use
  - Generates an IP block
  - IP blocks are placed in firmware forming larger designs for FPGAs
- Vitis HLS provides libraries optimized for use with hardware: linear algebra library
- HLS Pragmas afford more control over the implementation
  - Enforce pipelining: block continually accepts new inputs rather than waiting until the whole block finishes
  - Specify resources



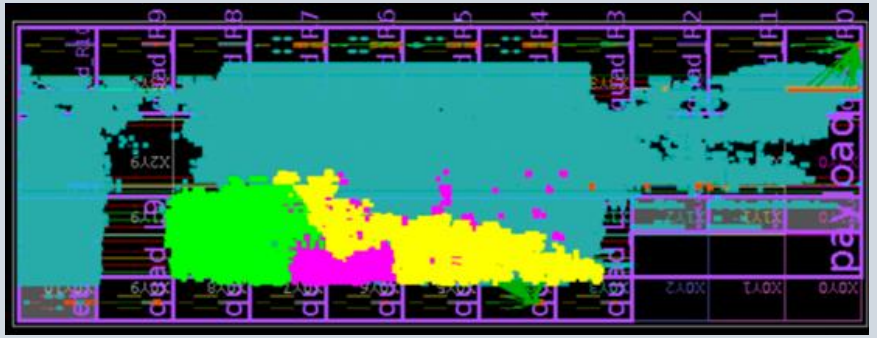
Online Fitting in 2023

Online Fitting in 2023

- Tested online fitting in a station without a target
  - No macroscopic electric or magnetic fields → **tracks are straight lines → X & Y projections are independent**
  - Projections associated via shared UV hits
- **Tested in beam**
  - Fit only events with a single hit in each module (no need to select candidates)

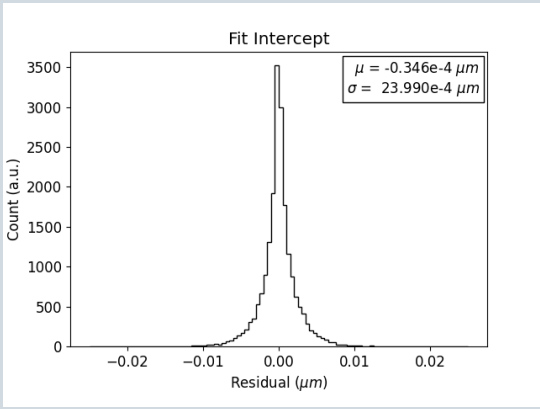


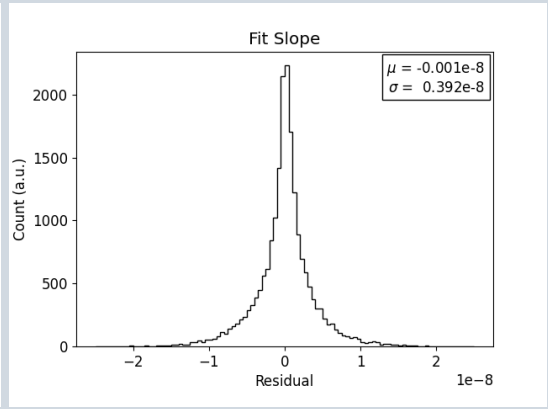
- Latency of ~1200 cycles at 320 MHz clock (3.75 μs)
- 7.75 % of resources
  - Green and yellow are fitting in X & Y
  - Pink is coordinate conversion and infrastructure
  - Blue is rest of DAQ

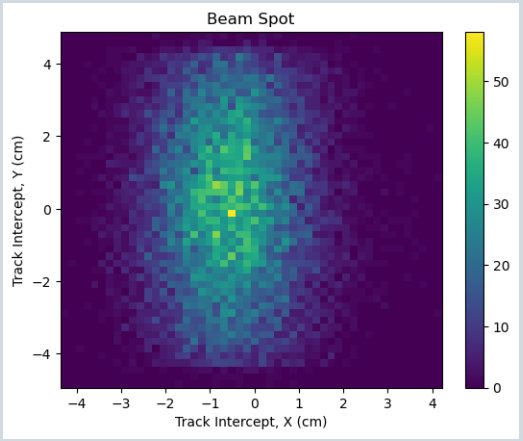


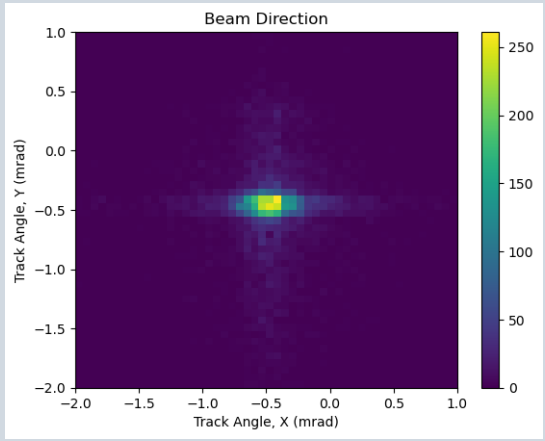
Validation and Results from 2023

- Validate algorithm by comparing fit parameters from online and offline fitting
- **Standard deviations for parameter residuals well below relevant uncertainties**
  - Tracker module uncertainty is 26 μm, intercept standard deviation four orders of magnitude smaller
  - Expect some difference due to floating point rounding





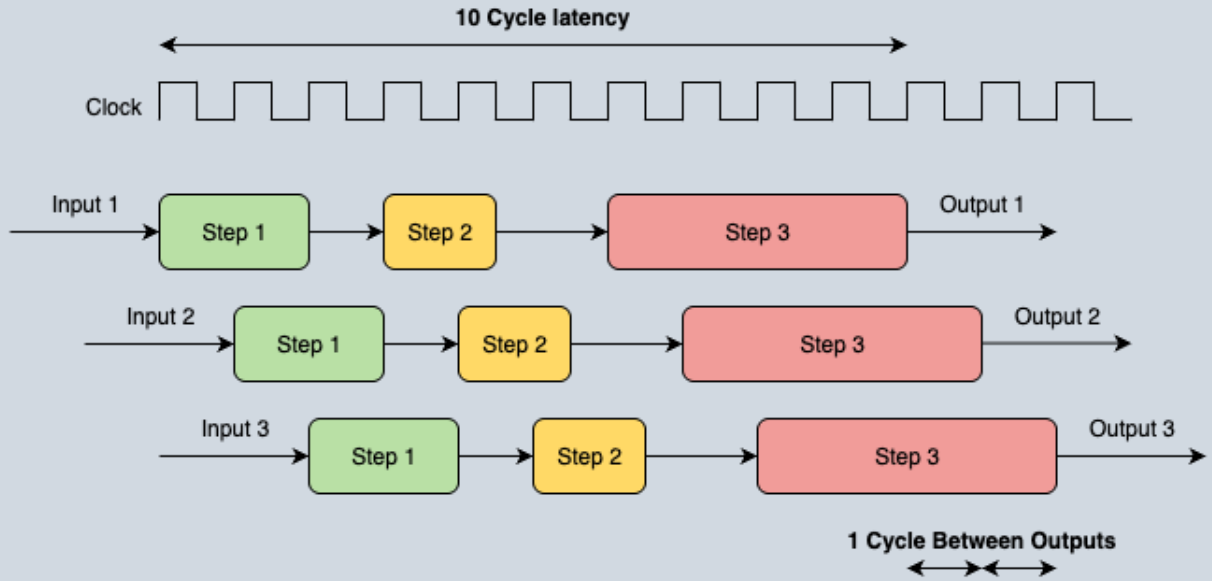




- Able to reconstruct beam spot and direction from fitted intercepts and slopes

Further Development to Fitting Block

- **Pipelining**
  - Decreases the time required between successive fits, the initiation interval (II)
  - **Parallel execution**
  - Typically increases resource use
  - Can decrease latency due to required changes



- **Replacing HLS functions with custom optimized versions**: include basic assumptions
  - Fitting implementation uses QR factorization, matrix inversion, matrix multiplication
  1. Some entries are always zero
  2. A column of matrix to be QR factorized is always one: implement with Gram-Scmidt, choose column of ones as first basis vector to not repeat calculations
  - **Simplify calculations**

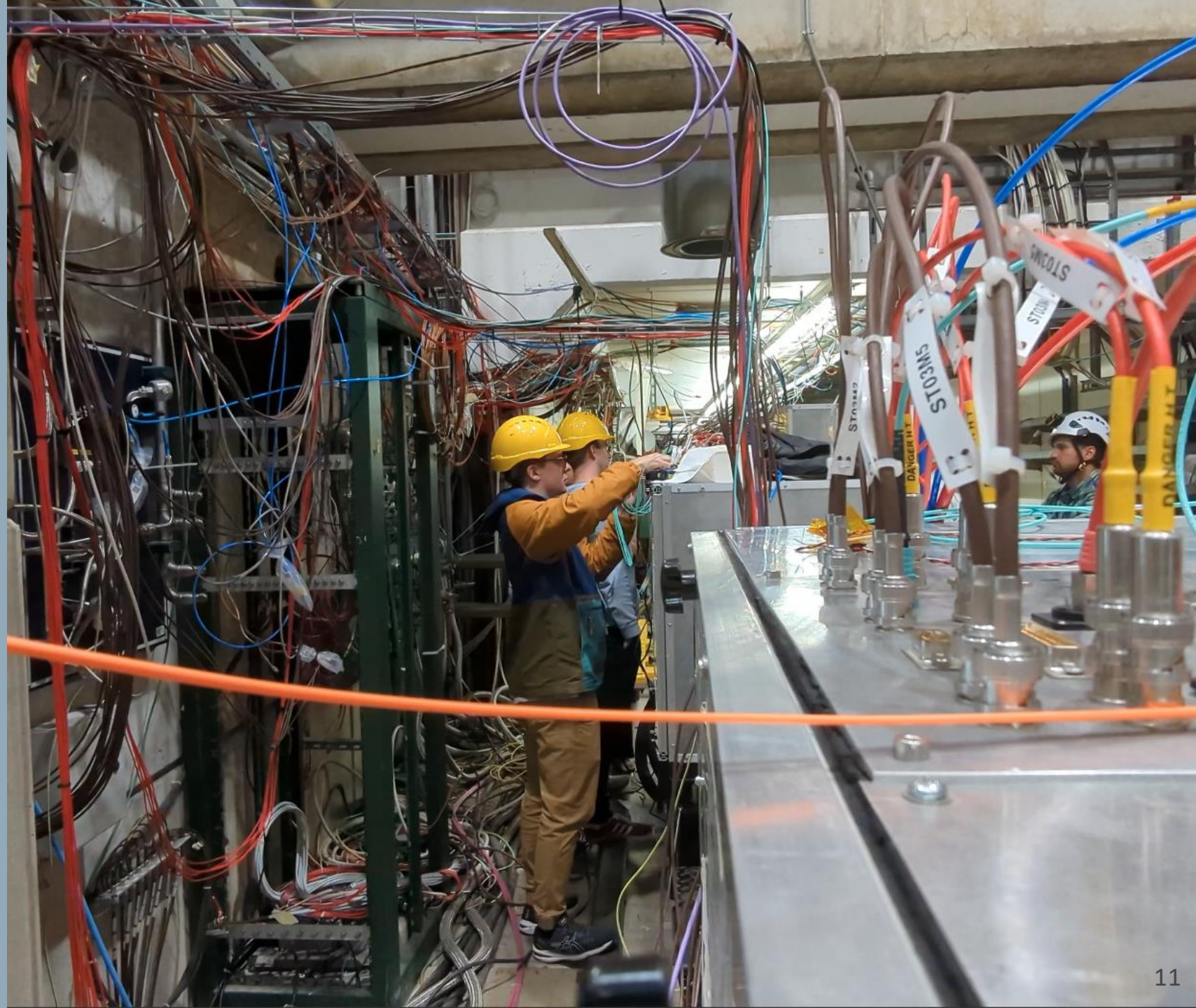
Results from Further Development

- HLS vs Custom linear algebra functions, with and without pipelining

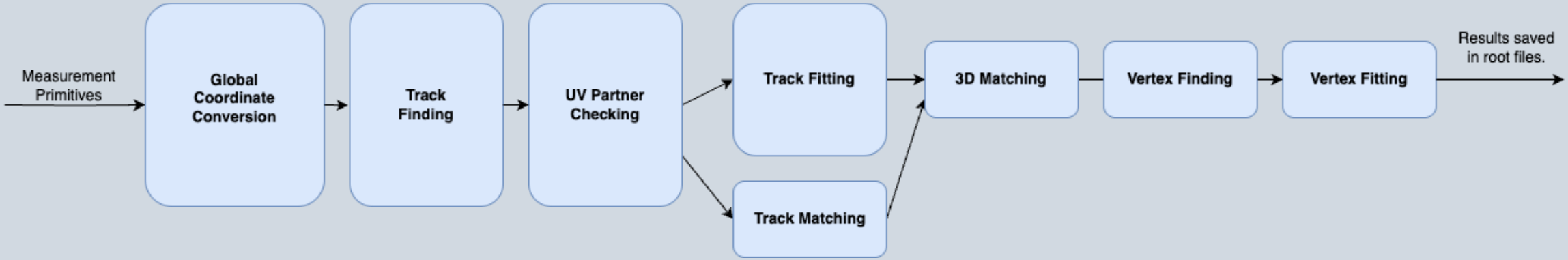
	HLS	Custom +Pipeline
Latency (cycles [μs])	1369 [4.28]	**431 [1.35]**
II (cycles [μs])	1372 [4.28]	**6 [0.02]**
LUTs (%)	7.74	4.56
Flip-Flops (%)	7.02	4.42
DSPs (%)	9.30	6.66
BRAM (%)	4.47	2.24

- The combination of custom functions and pipelining significantly decreases latency, II , and resource use
- **At 320 MHz, II = 6 corresponds to fits calculated at ~50 MHz, in line with peak rate at CERN M2 beam**

Online Fitting and Vertexing in 2025

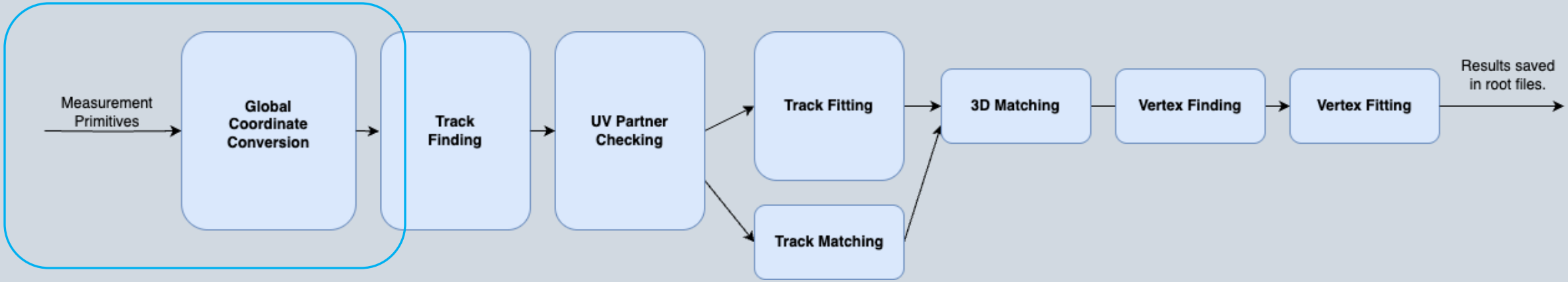


Online Fitting and Vertexing in 2025



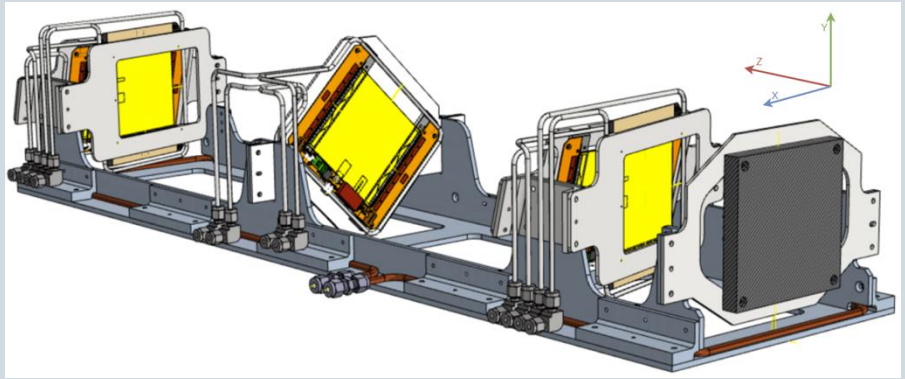
- Extend online reconstruction to vertexing in 2025 test beam
  - Requires 2 hits per module
- Aim to validate the algorithm, not push for performance
  - No pipelining, etc.
  - Due to time constraints from beam test

Online Fitting and Vertexing in 2025

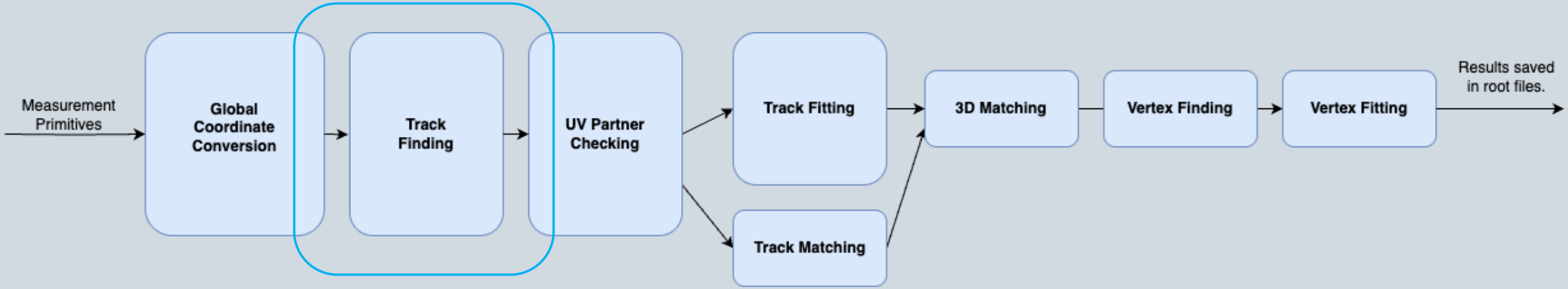


Global Coordinate Conversion

- Converts outputs from tracker modules to positions in global coordinate frame
- Event ID used to correlate online results with data from mainline DAQ
- UV modules are rotated, pairs of hits combined and rotated together

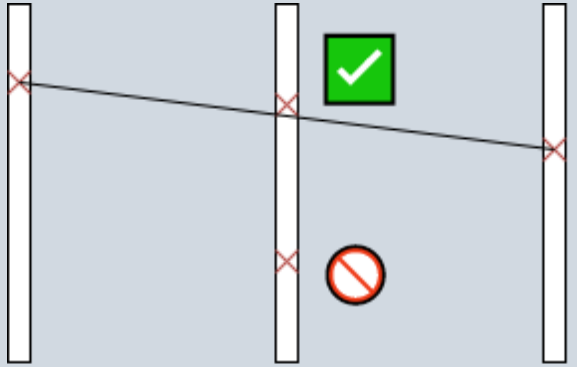


Online Fitting and Vertexing in 2025

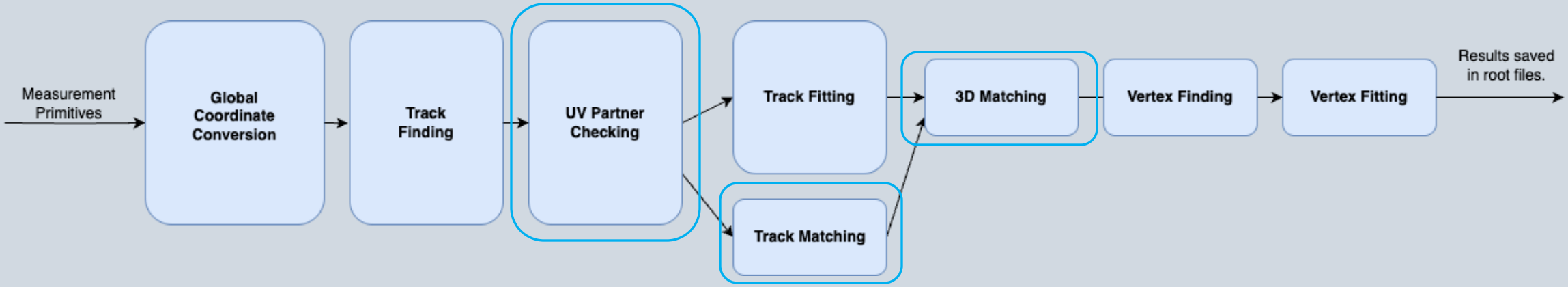


Track Finding

- Select track candidates before fitting (separately in X & Y):
  1. Select a pair of hits in first and last modules
  2. Project to UV plane
  3. Select hits close to projection to form candidates



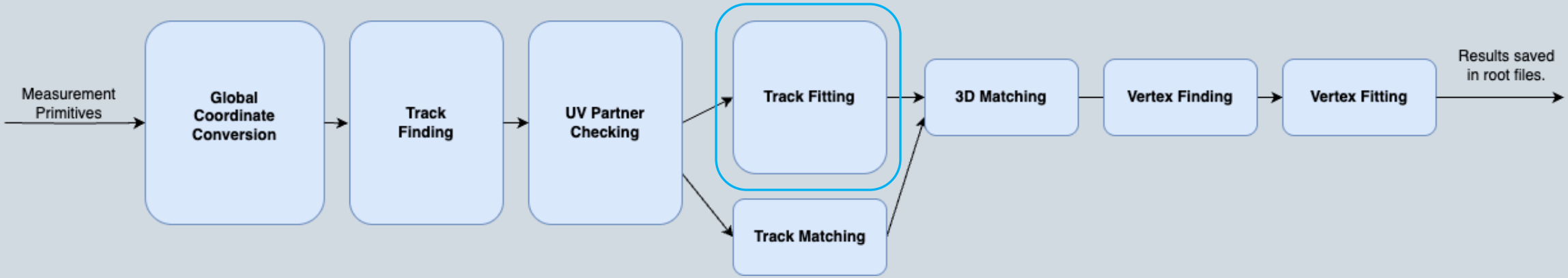
Online Fitting and Vertexing in 2025



Matching X & Y Track Projections

- Three blocks involved with matching X & Y fits
- **UV Partner Checking**: before fitting, check if there are candidates in X & Y for each UV hit.
- **Track Matching**: parallel with fitting, find each possible way to pair X & Y candidates.
- **3D Matching**: after fitting, form 3D fit candidates

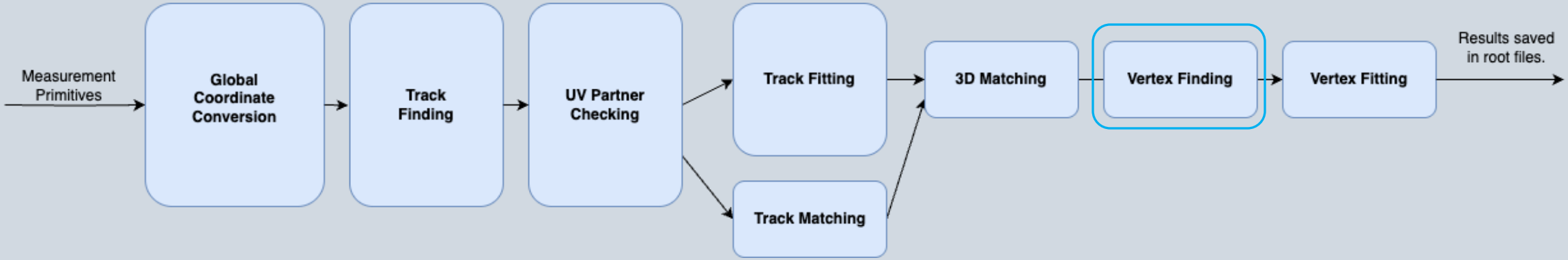
Online Fitting and Vertexing in 2025



Track Fitting

- Calculate track parameters and parameter covariances (separately in X & Y)
- Changes since 2023:
  - Custom functions
  - Adapt to more tracks per event

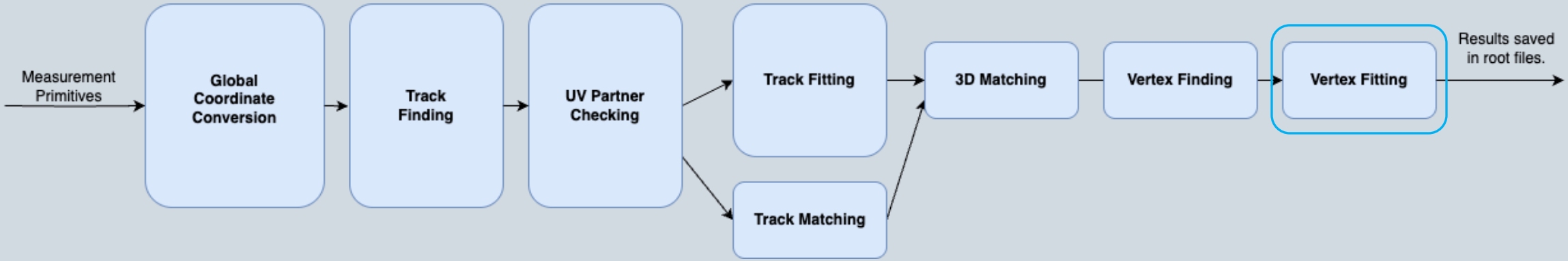
Online Fitting and Vertexing in 2025



Vertex Finding

- Select pairs of 3D track candidates for vertexing
  - No hit sharing
- Simple due to time constraints

Online Fitting and Vertexing in 2025

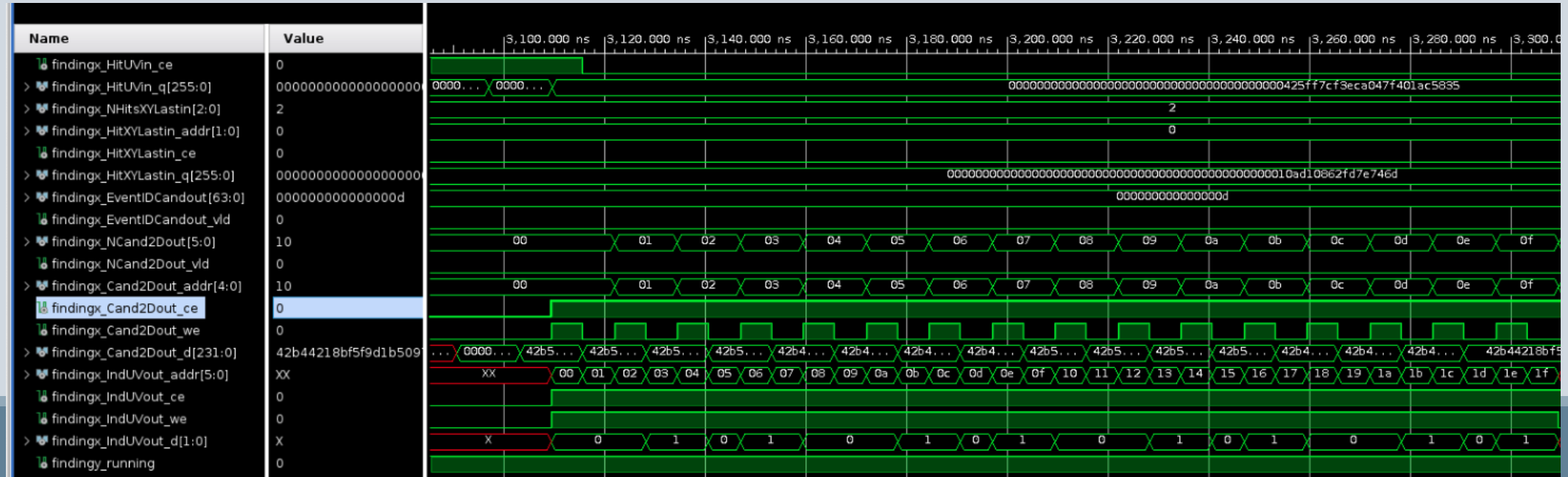


Vertex Fitting

- Estimate vertex position for pairs of 3D tracks
- Relies on Newton's Method
- Iterative: data dependencies limit pipelining
  - Will need to balance number of iterations and accuracy
- Involves many steps of linear algebra

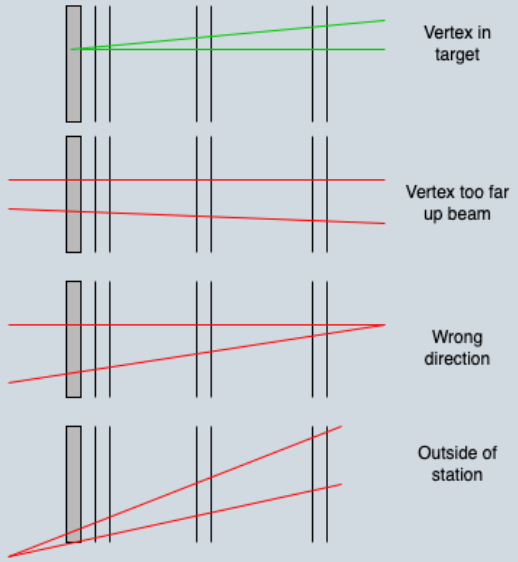
Current Status for 2025 Test Beam

1. Individual blocks implemented and validated in HLS
2. Simulate full chain in VHDL, verify results and control firmware
3. Test on FPGA, injecting “by hand” data taken so far
4. Test in a station with a target
   - Aim to save fit tracks and vertices for validation and analysis

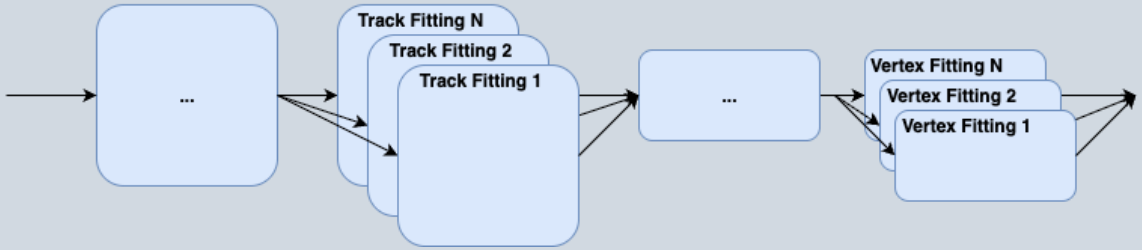


Post-TB Plans

- Pipeline full chain
- Custom linear algebra functions elsewhere, particularly vertexing
- Implement more robust vertex finding
  - Currently, just select two 3D tracks which do not share hits
  - Rough calculations: position, distance at closest point



- Understand expected number of hits, fits, vertices and latency budget
  - How many track fitting blocks in parallel?
  - How many vertex fitting block in parallel?



  - How aggressive need selections be? Track finding window, track quality…

General Application

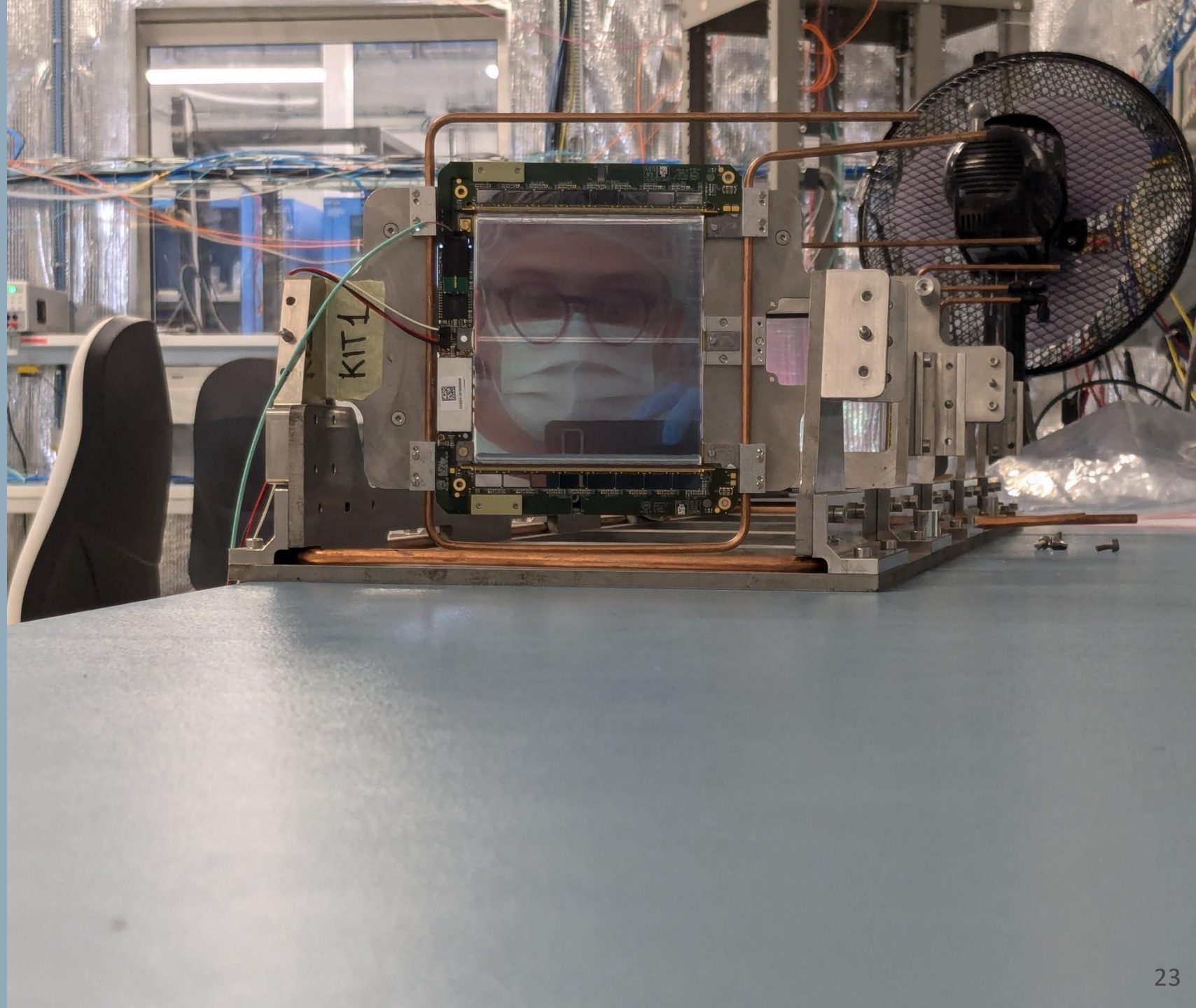
- **Aim for track finding, track fitting, vertex finding, vertex fitting blocks to remain general**
  - Agnostic to tracker geometry: only number of hits per track may change
  - May be useful for other test beams, cosmic ray tracking
- Assumptions are made that are general to straight tracks
  - X & Y track projections are independent
  - QR factorization of a matrix with a column that is always ones:

    Matrix is a Jacobian of a linear function, derivative by intercept is one
- Other blocks will be geometry dependent
  - Global coordinate conversion
  - Association of X & Y fits

Conclusions

- **Online tracking with HLS has been shown feasible** for event selection in MUonE
- **Online tracking and vertexing with HLS** will be tested during a 2025 test beam
- There is a clear path forward for improvements to current implementation to reduce resource use and latency, and increase throughput
- Current implementations of track finding and fitting, vertex finding and fitting are **general to straight tracks**, and future implementation will preserve this

Thank you.



Backup

Results from Further Development

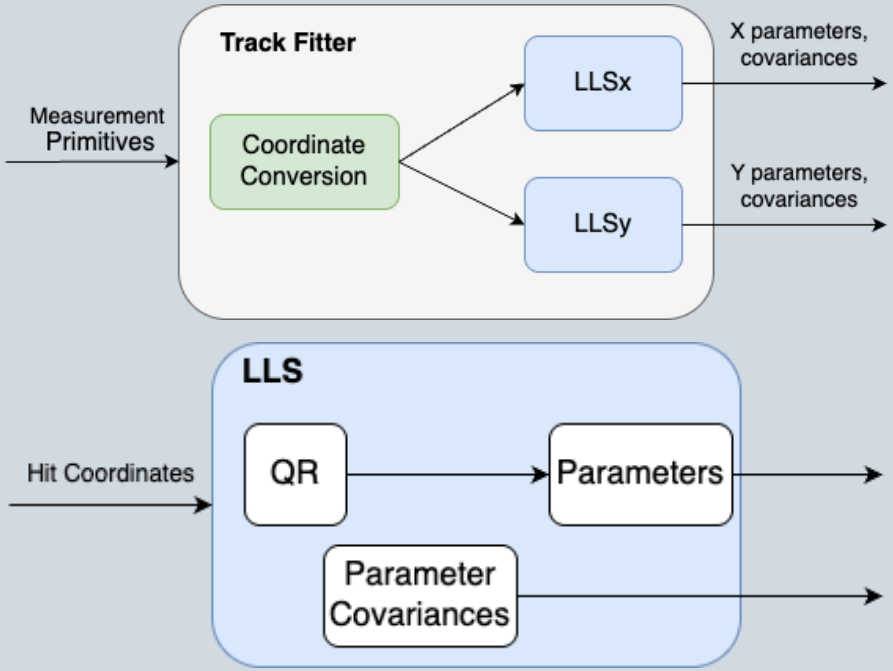
- HLS vs Custom linear algebra functions, with and without pipelining

	HLS	HLS +Pipeline	Custom	**Custom +Pipeline**
Latency (cycles [μs])	1369 [4.28]	852 [2.66]	701 [2.19]	**431 [1.35]**
II (cycles [μs])	1372 [4.28]	470 [1.47]	704 [2.20]	**6 [0.02]**
LUTs (%)	7.74	10.82	4.78	**4.56**
Flip-Flops (%)	7.02	10.43	4.52	**4.42**
DSPs (%)	9.30	22.41	6.55	**6.66**
BRAM (%)	4.47	3.86	2.24	**2.24**

- The combination of custom functions and pipelining significantly decreases latency, II , and resource use
- **At 320 MHz, II = 6 corresponds to fits calculated at ~50 MHz, in line with peak rate at CERN M2 beam**

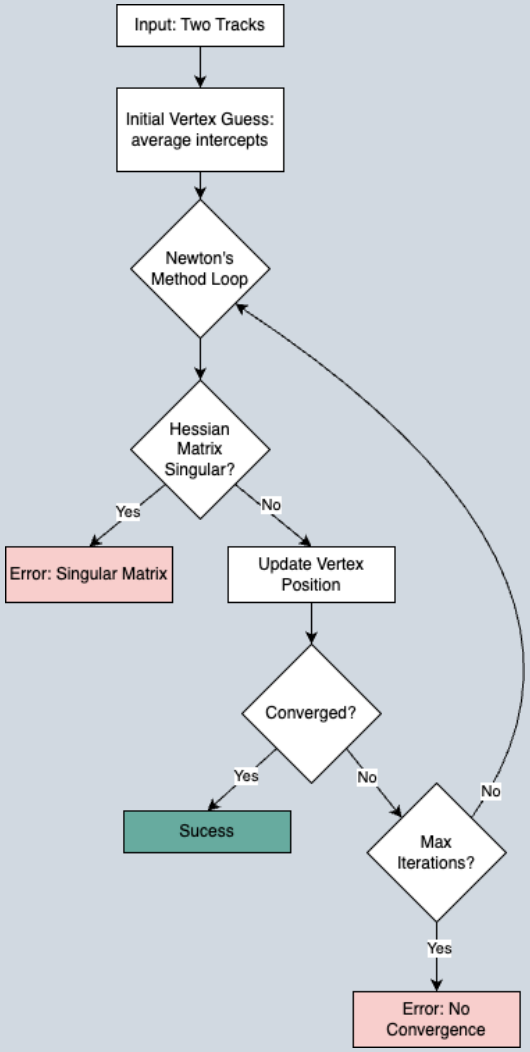
LLS Track Fitting

- Calculate parameters using LLS: $m = Fp$
  - m: vector of measurements
  - F: Jacobian
  - p: track parameters
- F generally noninvertible, use QR Factorization
  - Q orthogonal $\leftrightarrow$ $Q^{-1} = Q^T$
  - R right triangular
- Calculate p: $p = R^{-1}Q^T m$



Vertex Fitting Details

- Initial guess: average of track intercepts, (x, y) at target
- Newton's method: minimizing sum of squared distances between vertex and tracks
  - Compute gradient of this sum (3-vector)
  - Compute Hessian matrix of this sum (3-by-3 matrix)
  - Both require multiple steps of linear algebra
- Hessian matrix considered singular if determinant $< 10^{-10}$
- Update vertex position: $\boldsymbol{v_{n+1}} = \boldsymbol{v_n} - (\nabla^2 S)^{-1} \cdot \nabla S$
  - S: sum of squared distances
  - $(\nabla^2 S)^{-1}$: Hessian matrix
  - ∇S: gradient
- Converged if $\|v_{n+1} - v_n\| < 10^{-6}\ cm$



For more information, see Frühwirth *Pattern Recognition, Tracking and Vertex Reconstruction in Particle Detectors* 8.1.1.1.

Number of Objects

For an event with 2 hits per module, no cuts on track finding window, track quality, etc.

Object	**Number**
Hits Per Module	2
UV Virtual Hits	4
2D Tracks (X & Y each)	16
3D Tracks	64
Vertices	32

Rates for single μe scatters + N PU μ

Pattern	Rate (MHz)
1μ int.	0.32
Int + 1 PU	0.18
Int + 2 PU	0.06
Int + 3 PU	0.01
Int + $\geq$3 PU	0.12