FPGA Implementation of the General Triplet Track Fit

Kadir Tastepe[1], Sebastian Dittmeier[1], Abhirikshma Nandi[1], Christof Sauer[1,2], André Schöning[1]

[1]Physikalisches Institut - Universität Heidelberg, [2]now at CERN

Contact: tastepe@physi.uni-heidelberg.de

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Online Track Reconstruction in High Rate Experiments

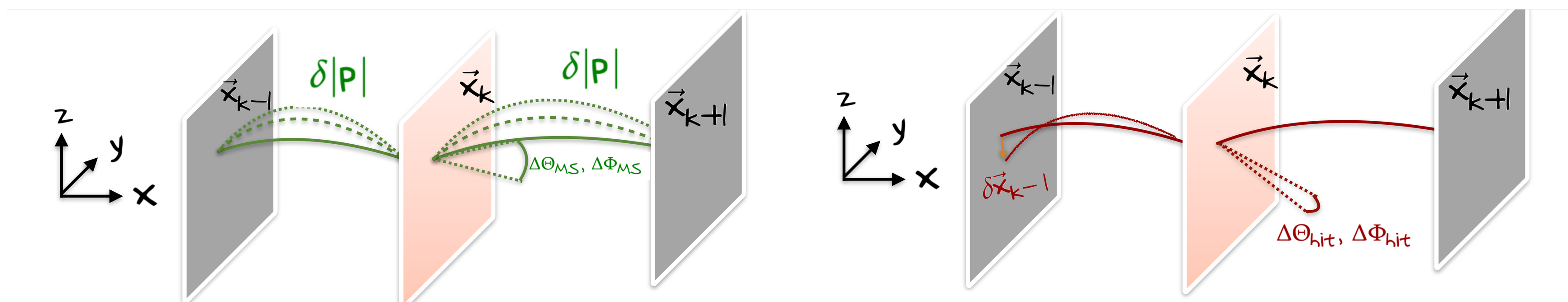
Reconstruction of charged particle tracks in high-energy physics experiments is a computationally intensive task. Due to the increasing number of simultaneous particle collisions in future high-luminosity colliders, like HL-LHC and FCC-hh, the challenge of tracking and online event reconstruction becomes even more significant. This demands for innovative algorithms running on accelerated hardware, e.g. FPGAs, GPUs and ASICs.

The **General Triplet Track Fit [1]** is a novel parallelisable track-fitting algorithm that extends the MS(Multiple Scattering)-only fit [2] by including hit uncertainties, making it usable for high-energy collider experiments.

With their inherent **parallelism**, **power efficiency** and **reconfigurability**, FPGAs are becoming increasingly attractive as co-processors for large data centres, such as filter farms, to meet the challenges of increasing throughput and computational complexity. A preliminary FPGA implementation of the General Triplet Track Fit for future heterogeneous online farms is developed, using **high-level synthesis** on **AMD FPGAs** with algorithmic optimisations to exploit the device's potential fully.

[3]

General Triplet Track Fit (GTTF)



Minimize to get track paramaters

$$\chi^2 = \sum_{\text{Triplets } k} \left\{ \frac{\Delta\Theta^2}{\sigma_\Theta^2} + \frac{\Delta\Phi^2}{\sigma_\Phi^2} \right\}_{MS,k} + \sum_{\text{Hits } m} \left\{ \vec{\delta x}^T V^{-1} \vec{\delta x} \right\}_m$$

Multiple scattering — Hit uncertainties

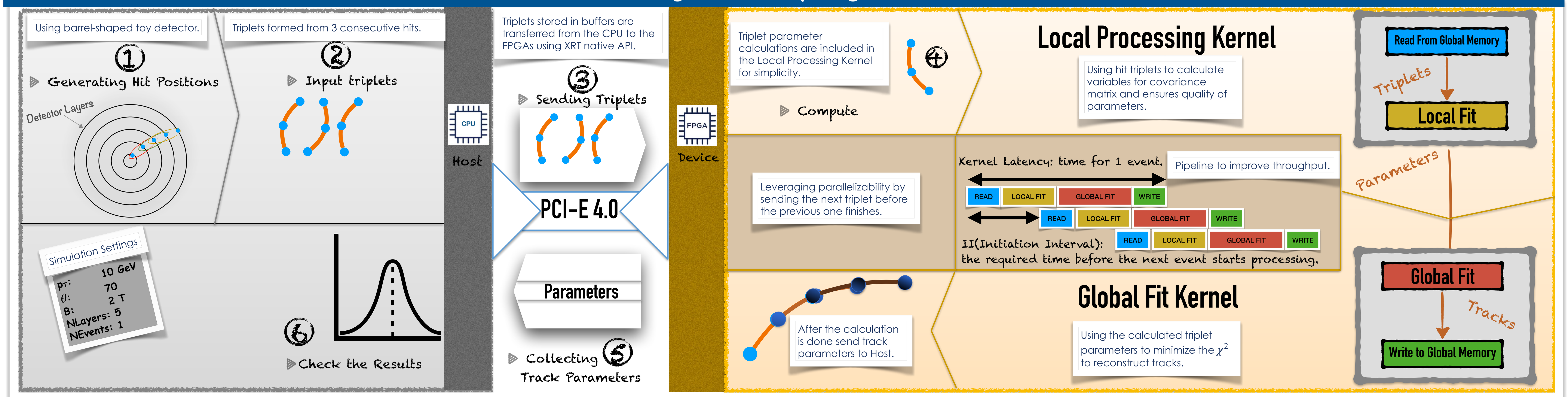
Triplet track fit based on two-step procedure:

(1) Accounts for all detector-specific information, e.g., B field, material budget. **Local Fit** is optional for filtering.

(2) Detector independent; compute track parameters from triplet parameters.

Local processing of Triplets involves computationally complex trigonometric functions.

Heterogeneous Computing Workflow



Optimization for Throughput

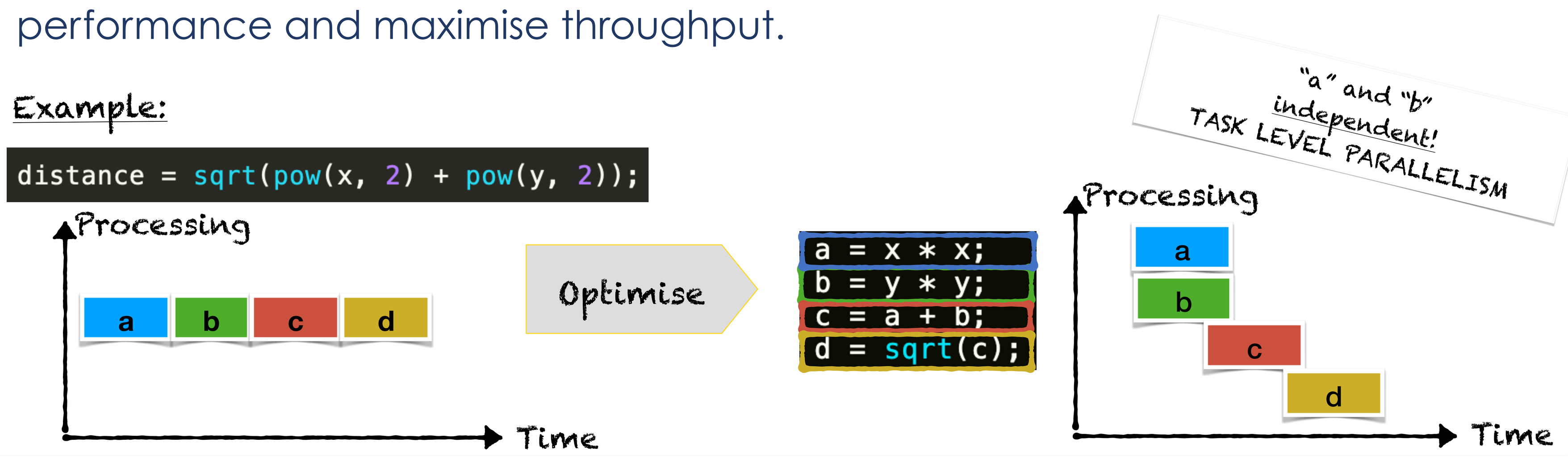
The CPU-based GTTF algorithm is translated into HLS C++. The design is optimised by simplifying complex calculations and breaking them into smaller, reusable steps, which reduce logic depth and improve throughput. Loop optimisations (pipelining, unrolling, flattening, etc.) are applied to enhance performance and maximise throughput.

Example:

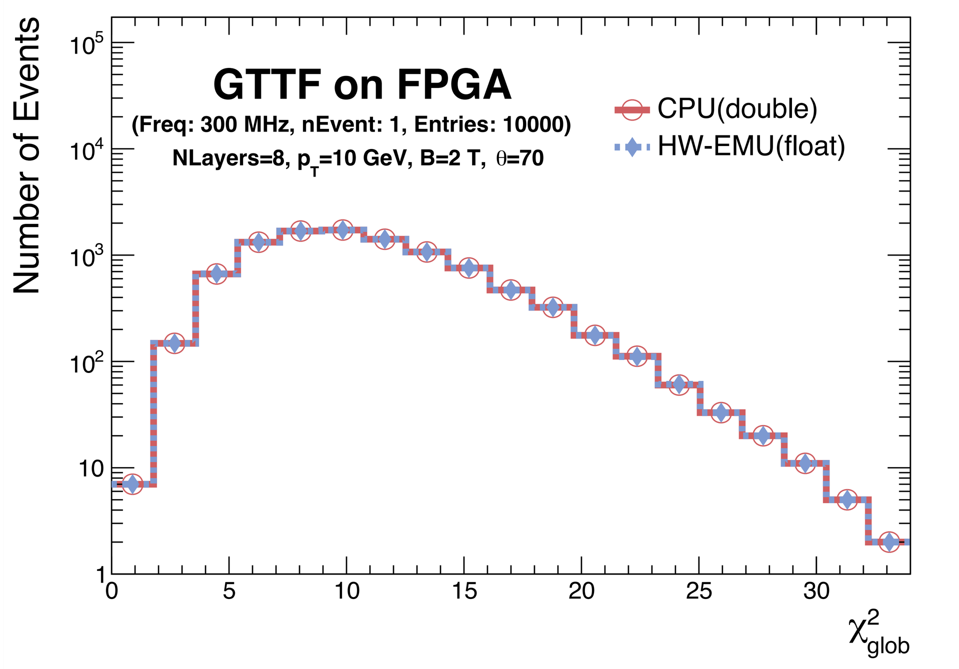
```
distance = sqrt(pow(x, 2) + pow(y, 2));
```

Optimise

```
a = x * x;
b = y * y;
c = a + b;
d = sqrt(c);
```

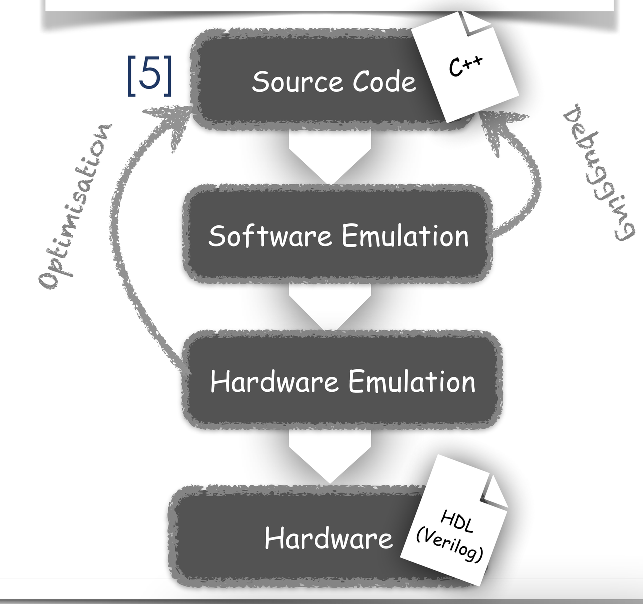


Functional Verification

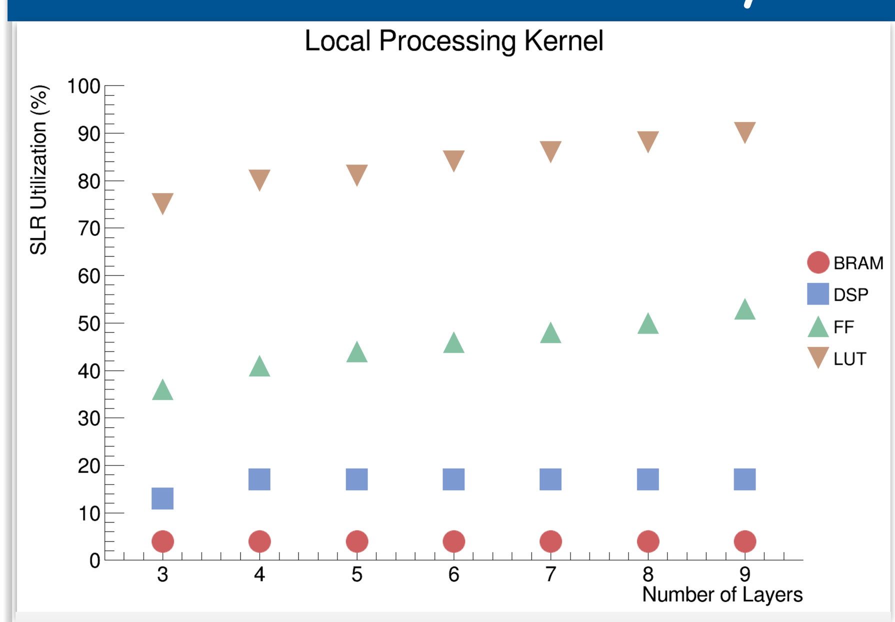


Calculated track parameters are consistent with CPU results.

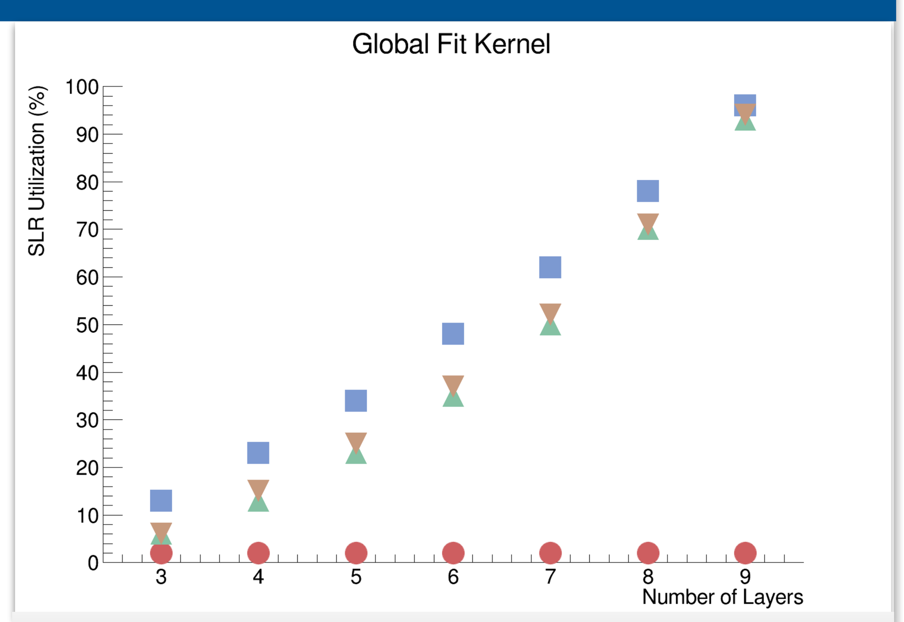
AMD Vitis HLS Toolflow [4] allows rapid design iterations and resource-throughput optimisations.



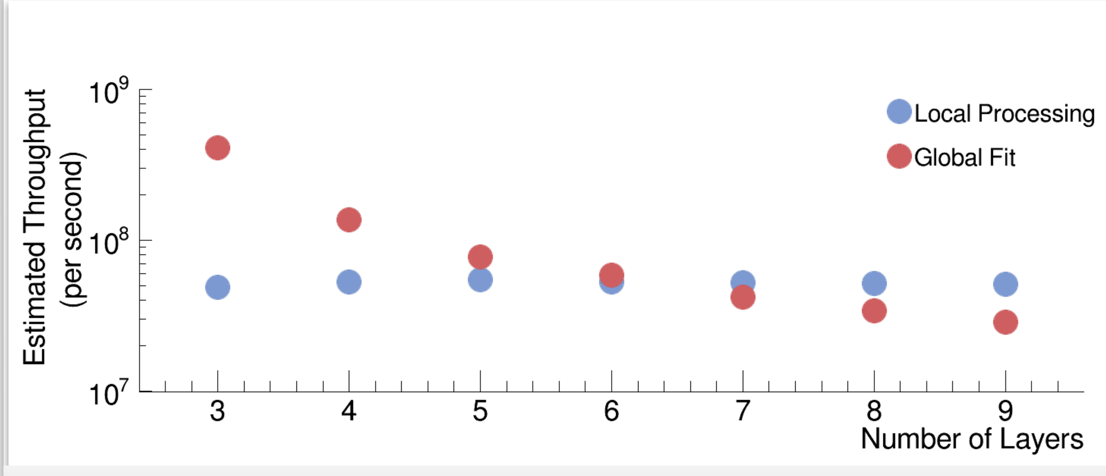
Synthesis Results



Preliminary implementation of the local processing is resource-intensive, because of the trigonometric functions.



With an increase in the number of triplets, the size of the covariance matrix also grows, becoming more resource-intensive.



Global Fit throughput decreases with an increasing number of layers, because of the matrix inversions. Local Processing Kernel scales with the number of triplets.

II: Initiation Interval

Clock Period: Duration of one clock cycle

NTriplets = NLayers − 2

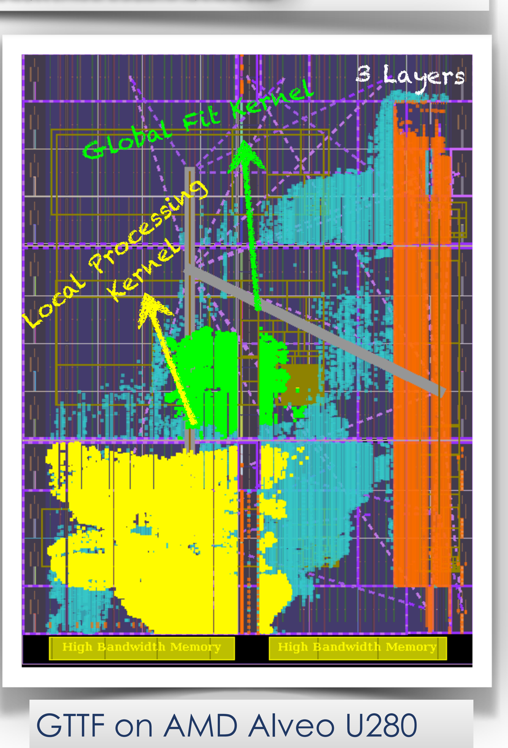
Throughput: Number of executed kernels per unit time

Local Processing: $\text{Throughput} = \frac{1}{II \times \text{Clock Period}} \times \text{NTriplets}$

Global Fit: $\text{Throughput} = \frac{1}{II \times \text{Clock Period}}$

Summary

The GTTF is a potential track fit for future experiments. A preliminary high-throughput FPGA implementation is developed using AMD Vitis HLS. The synthesis results indicate that processing rates of approximately **5x10⁷ local processings/s** and **4x10⁷ global fits/s** for **8 layers** with **float** precision can be achieved. Using approximations and trading off precision, further performance improvements are possible. Currently, various kernel models are tested if they increase performance. A resource-minimized version will also be implemented, allowing for optimisation based on application-specific needs.



GTTF on AMD Alveo U280

References

[1] Schöning, A. (2024). *A General Track Fit based on Triplets.* https://arxiv.org/abs/2406.05240

[2] Berger, N., Kozlinskiy, A., Kiehn, M., & Schöning, A. (2017). *A new three-dimensional track fit with multiple scattering.* https://doi.org/10.1016/j.nima.2016.11.012

[3] AMD. *Alveo U280 Data Center Accelerator Card.* AMD, https://docs.amd.com/r/en-US/ug1314-alveo-u280-reconfig-accel/Introduction

[4] AMD. (2023.1). *Vitis Application Development Flow User Guide* (UG1393)

[5] Mouser: *Acceleration RFSoC solutions with Vitis*, https://www.mouser.hk/blog/accelerating-rfsoc-solutions-vitis