Imperial College London

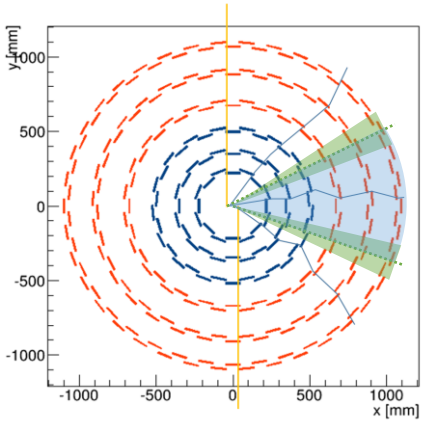
A Kalman Filter for Track Fitting

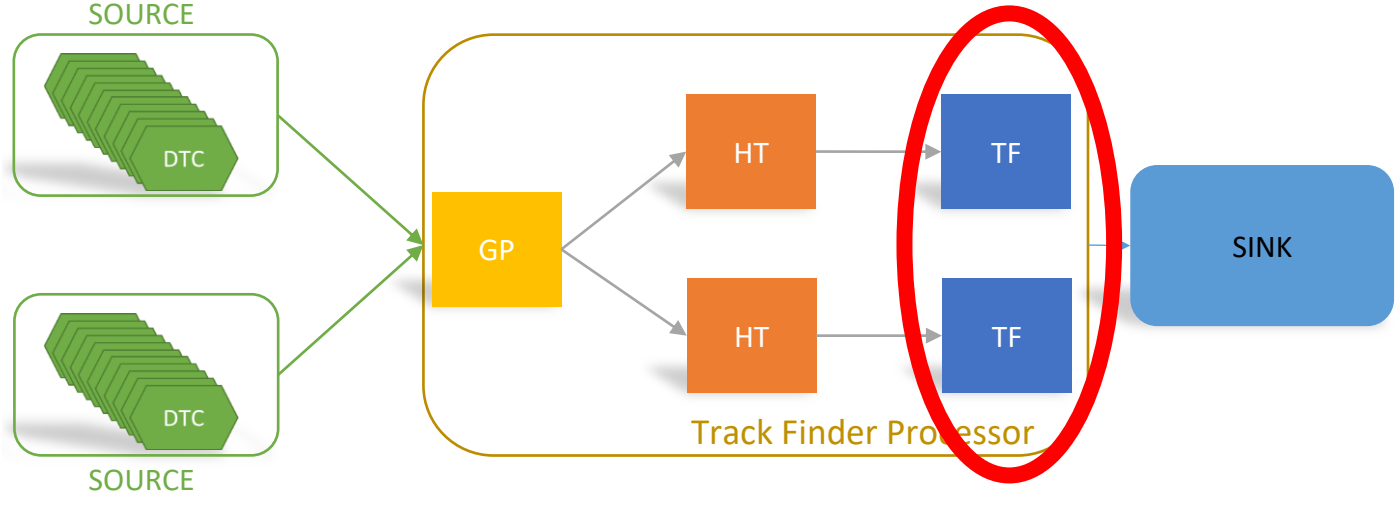
Imperial College London

Andrew W. Rose
Imperial College London

The Track Fitting environment

- Average 330 track candidates in TTbar + 200PU
  - 42-100 per octant spread in time
- Average 7.3 stubs per candidate, most candidates/stubs fake
- Coarse r-phi helix estimates which can seed the track-fit





WHAT IS A KALMAN FILTER?

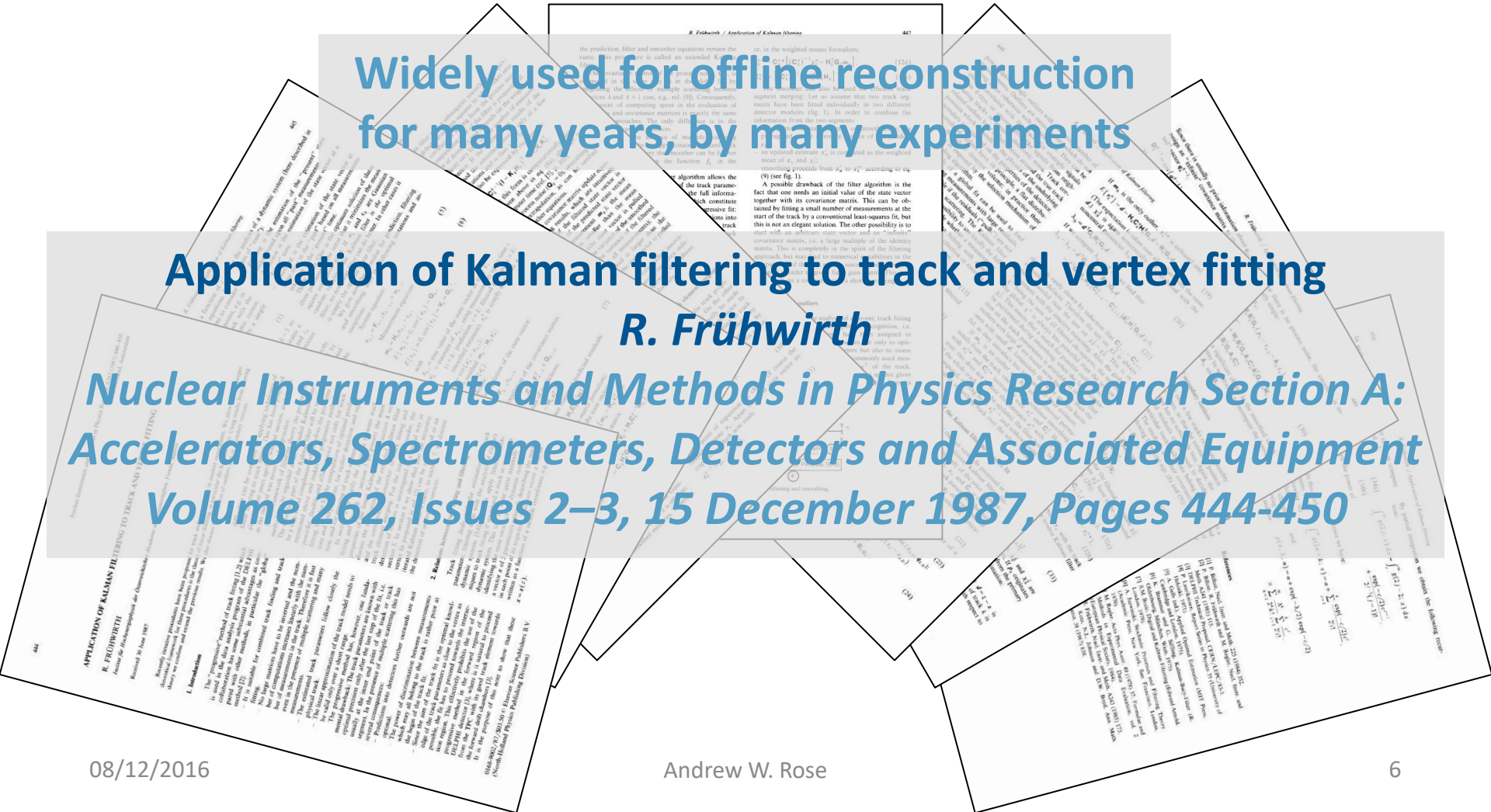
What is a Kalman Filter?

Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that **iteratively uses a series of measurements**, usually observed over time, **containing statistical noise and other inaccuracies**, **and produces estimates of unknown variables** that tend to be more precise than those based on a single measurement alone, by using Bayesian inference and estimating a joint probability distribution over the variables for each measurement. The filter is named after **Rudolf E. Kálmán**, one of the primary developers of its theory.

What is a Kalman Filter?

Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that **iteratively uses a series of measurements**, usually observed over time, **containing statistical noise and other inaccuracies**, **and produces estimates of unknown variables** that tend to be more precise than those based on a single measurement alone, by using Bayesian inference and estimating a joint probability distribution over the variables for each measurement. The filter is named after **Rudolf E. Kálmán**, one of the primary developers of its theory.

Kalman Filter for Track Fitting



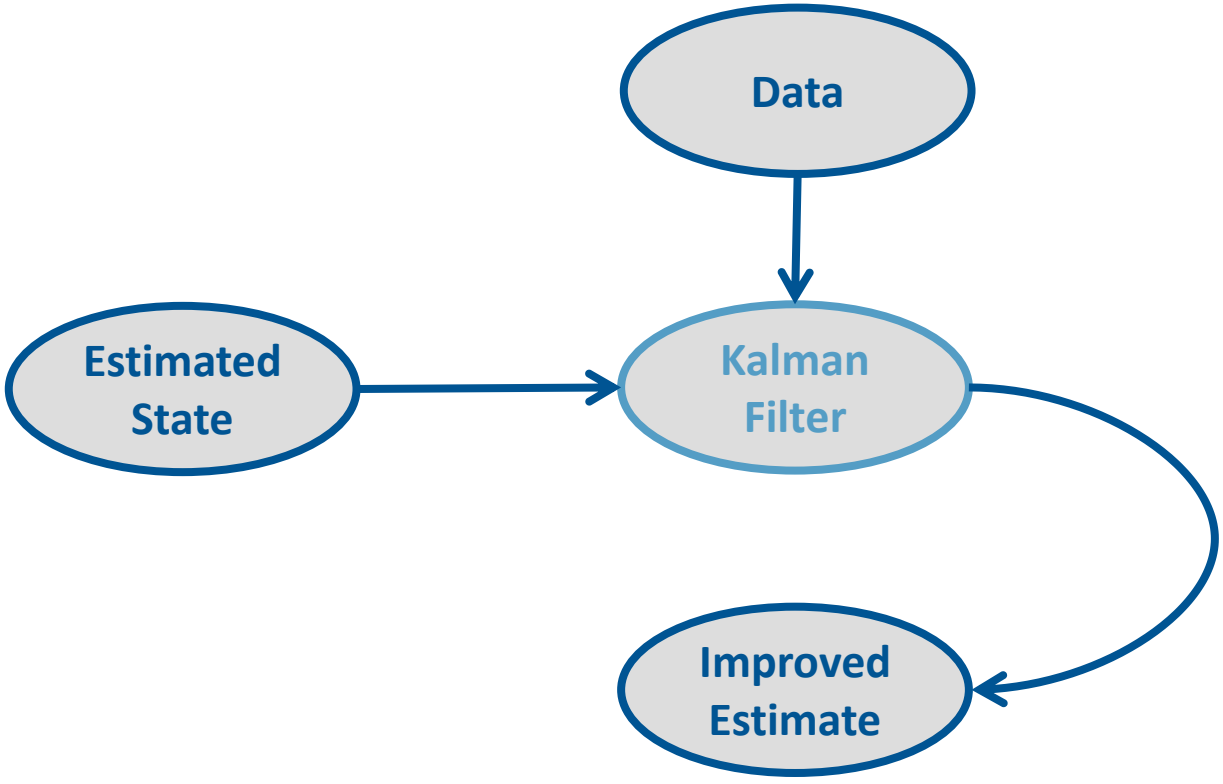
Widely used for offline reconstruction for many years, by many experiments

Application of Kalman filtering to track and vertex fitting

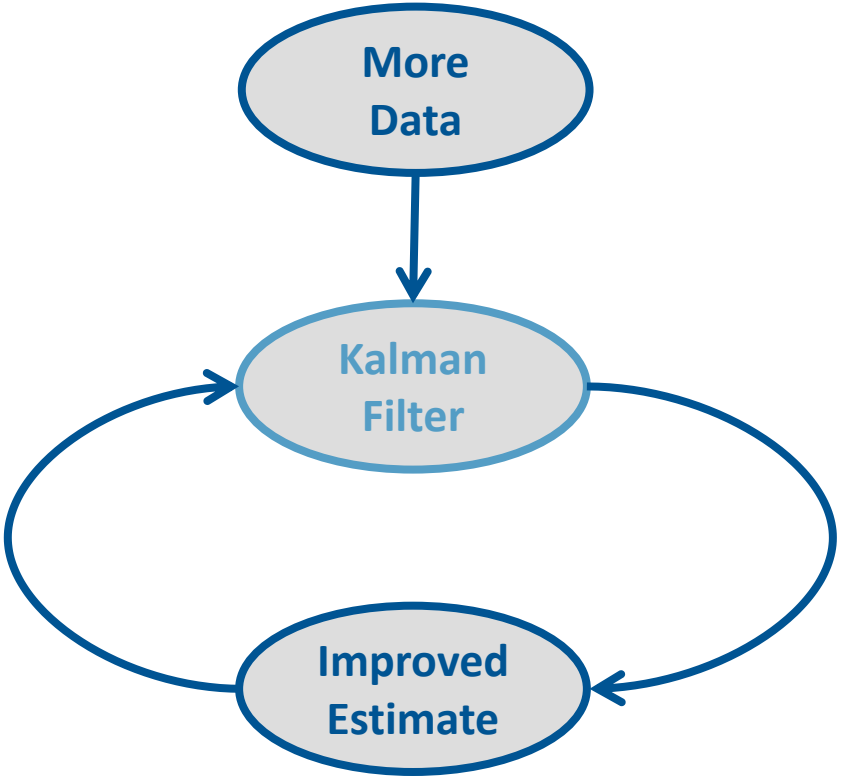
R. Frühwirth

Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment Volume 262, Issues 2–3, 15 December 1987, Pages 444-450

What is a Kalman Filter?



What is a Kalman Filter?



The advantages of the Kalman filter

The textbooks tell you that the Kalman filter is the optimal linear filter in cases where:

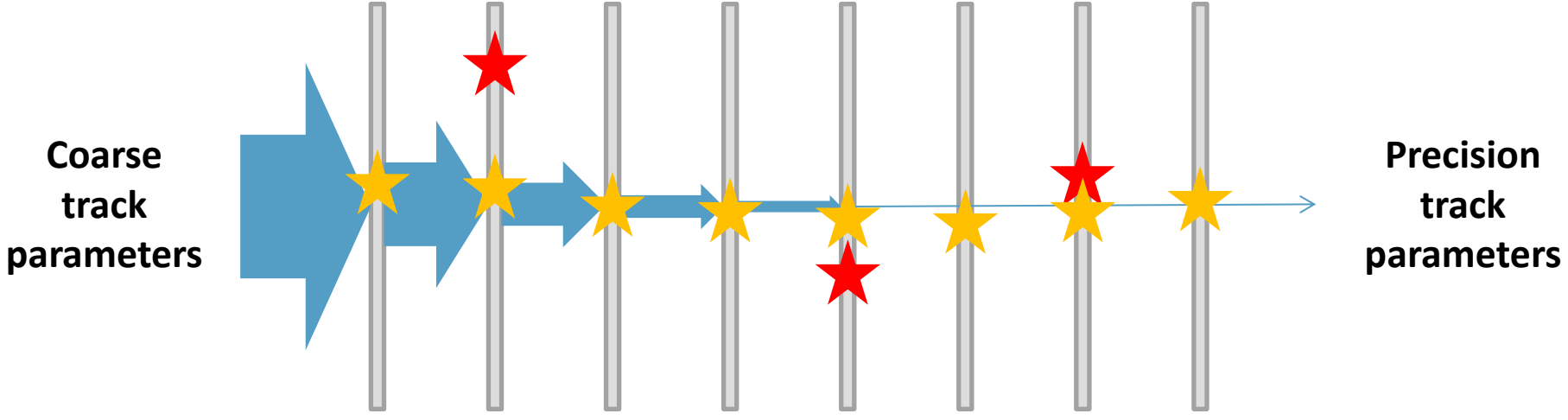
- **the model perfectly matches the real system**
- **the entering noise is white (uncorrelated)**
- **the covariances of the noise are exactly known**

The pragmatic advantages

- **Uses full stub granularity, no averaging or blurring over stubs**
  - **Best possible performance**

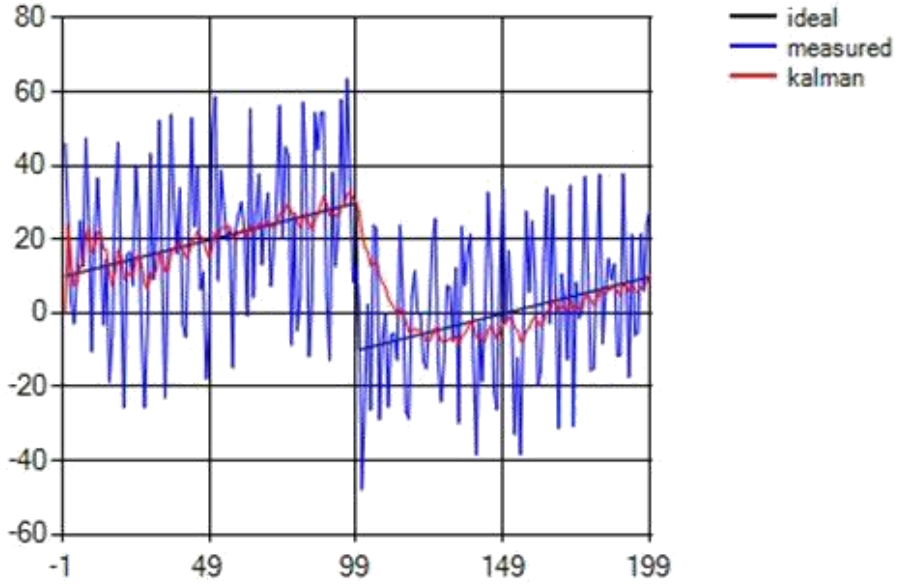
The pragmatic advantages

- Uses full stub granularity, no averaging or blurring over stubs
- It is local – one stub at a time
  - It is a filter, not just a fitter: can reject bad hits “on the fly”



The pragmatic advantages

- Uses full stub granularity, no averaging or blurring over stubs
- It is local – one stub at a time
- Intrinsically handles non-smooth functions and discontinuities
  - Multiple scattering can easily be modelled
  - Resilient to misalignment, material budget mismeasurement, etc.

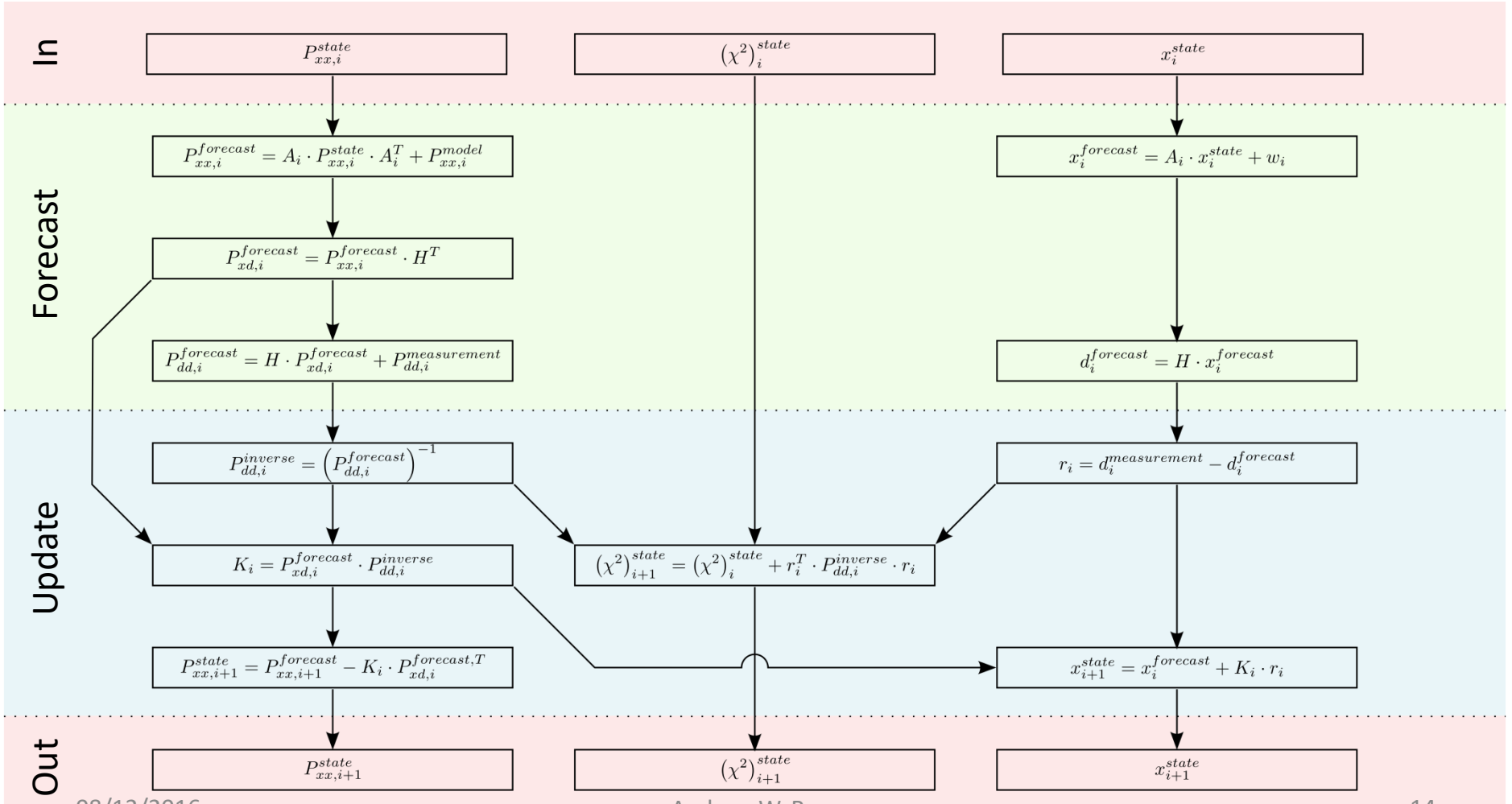


Why is everyone not using it, then?

- **It is everything a firmware engineer usually tells you to avoid:**
  - **It is iterative**
  - **It results in combinatorics**
  - **The data-flow is data-dependent**

- **Plus…**
  - **There is a lot of matrix maths (although this is true of most track fitting approaches)**

I mean, a lot of matrix maths



More subtleties

- **Because it is iterative, quantization errors accumulate**
  - **Requires use of very wide numbers to keep precision**
  - **In turn requires heavy use of block-RAM**
  - **Increases firmware complexity**

More subtleties

- Because it is iterative, quantization errors accumulate
- Certain points very sensitive to numeric instability
  - There are certain shortcuts which would save resources or latency which, unfortunately, cannot be taken

More subtleties

- Because it is iterative, quantization errors accumulate
- Certain points very sensitive to numeric instability
- Barrel + Endcap have different favoured parameterizations
  - Will come back to this later

More subtleties

- Because it is iterative, quantization errors accumulate
- Certain points very sensitive to numeric instability
- Barrel + Endcap have different favoured parameterizations
- Matrix inverse means division means YUK/OUCH/EEK!
  - Although, unlike other track-fitting approaches which require inversions, the KF only has to invert a 2×2 matrix, so look on the bright side

So the questions are surely…

DOES IT WORK?
HOW DO YOU KNOW?

Four testing handles

- **Compare:**
  - **Floating point CMSSW**

The “Proof of the Pudding”
The only test that really matters

  - **Hardware – Use the standard MP7 injection & capture framework**

Four testing handles

- **Compare:**
  - **Floating point CMSSW**
  - **Modelsim – plus VHDL itself exports XML to simplify clock-by-clock debugging**
  - **Hardware – Use the standard MP7 injection & capture framework**

Four testing handles

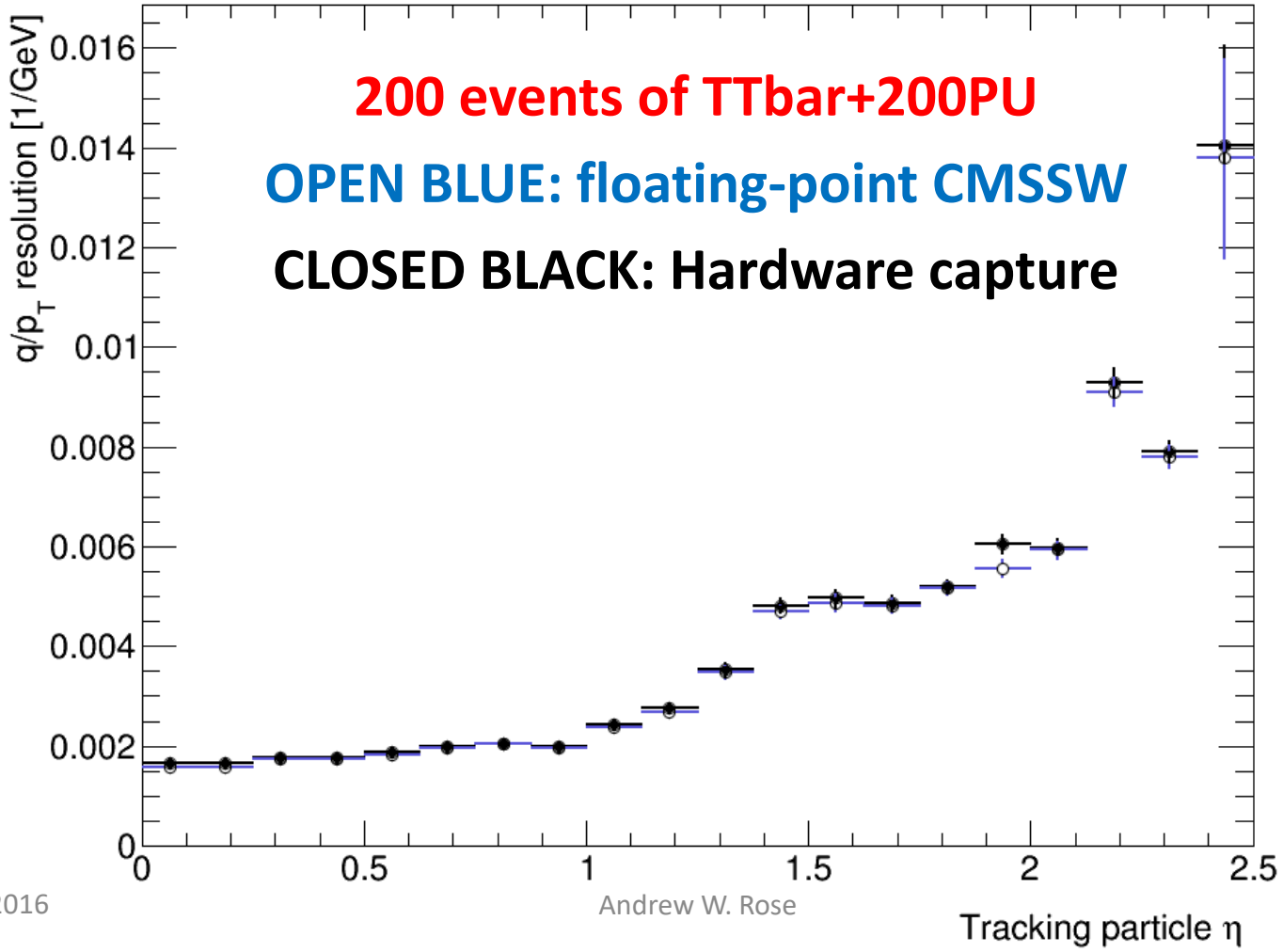
- **Compare:**
  - **Floating point CMSSW**
  - **CMSFW – bitwise & clock-cycle accurate C++ simulation of the Kalman maths, generated from the firmware and embedded into CMSSW**
  - **Modelsim – plus VHDL itself exports XML to simplify clock-by-clock debugging**
  - **Hardware – Use the standard MP7 injection & capture framework**

Four testing handles

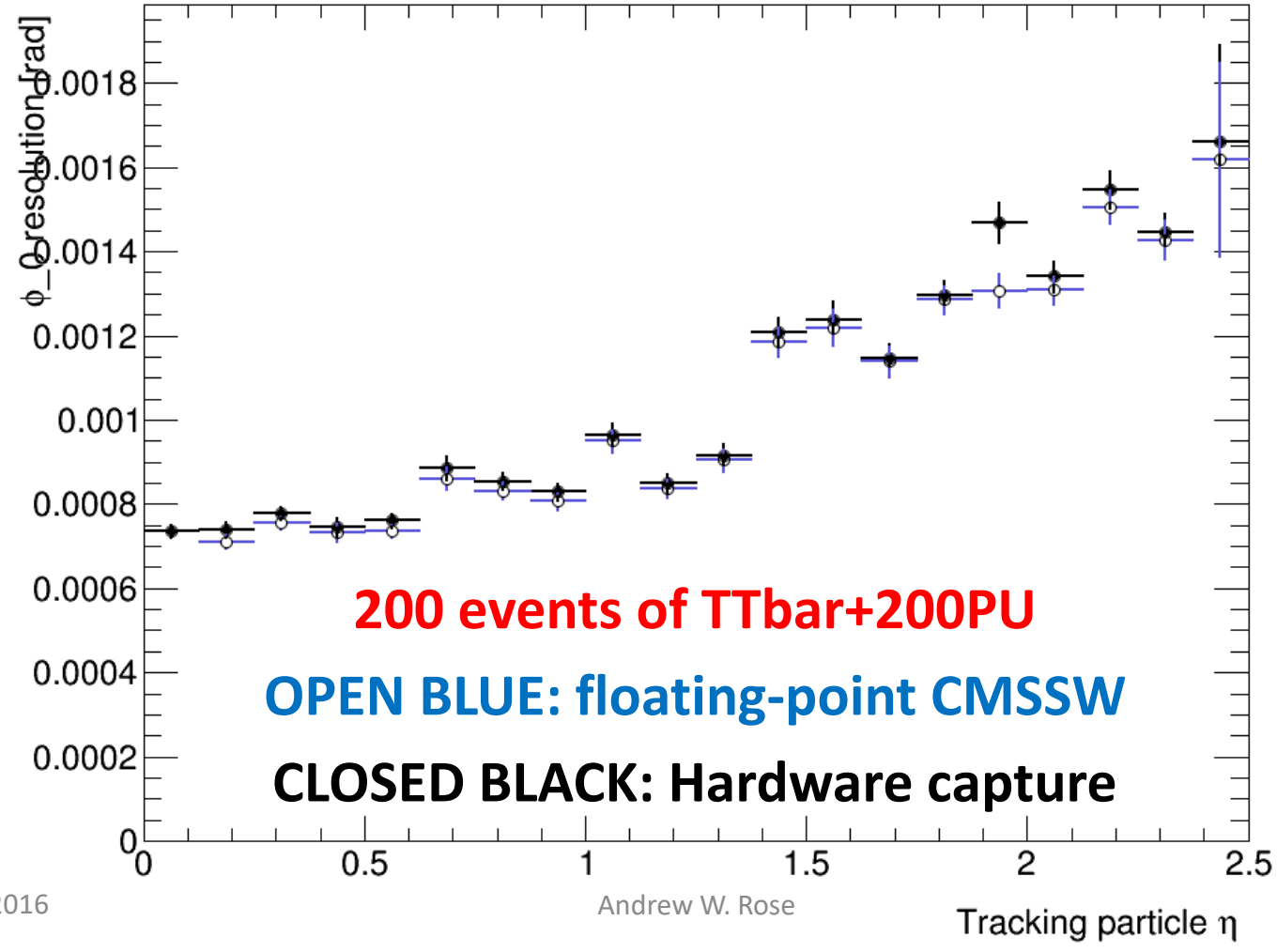
- Compare:
  - Floating point CMSSW

    - This is what is shown in the next few slides
    - For performance, please see later talks:
      - “Full Demonstrator Chain”
      - “Simulation results”
  - Hardware

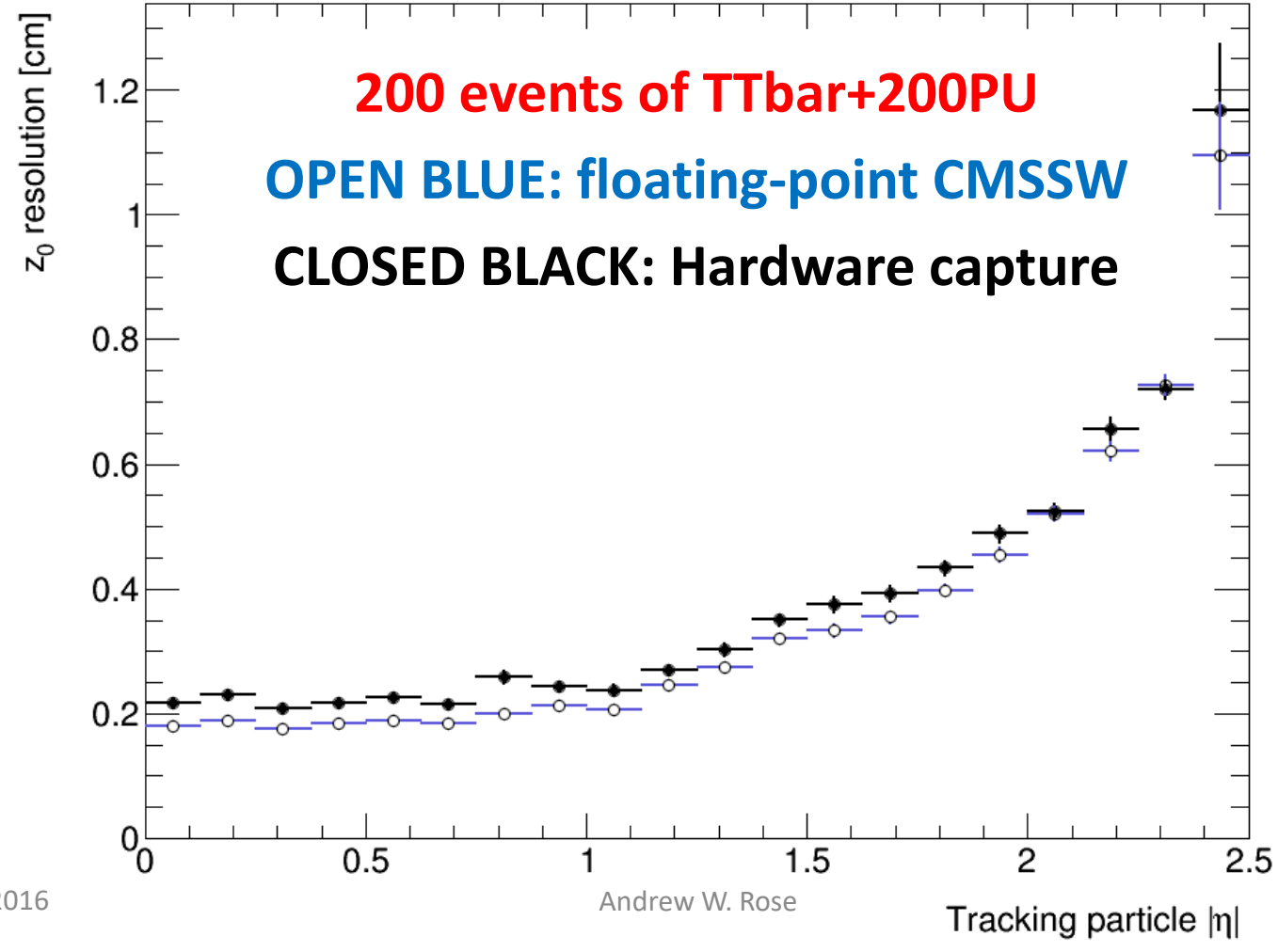
q/p_T resolution (1/GeV)



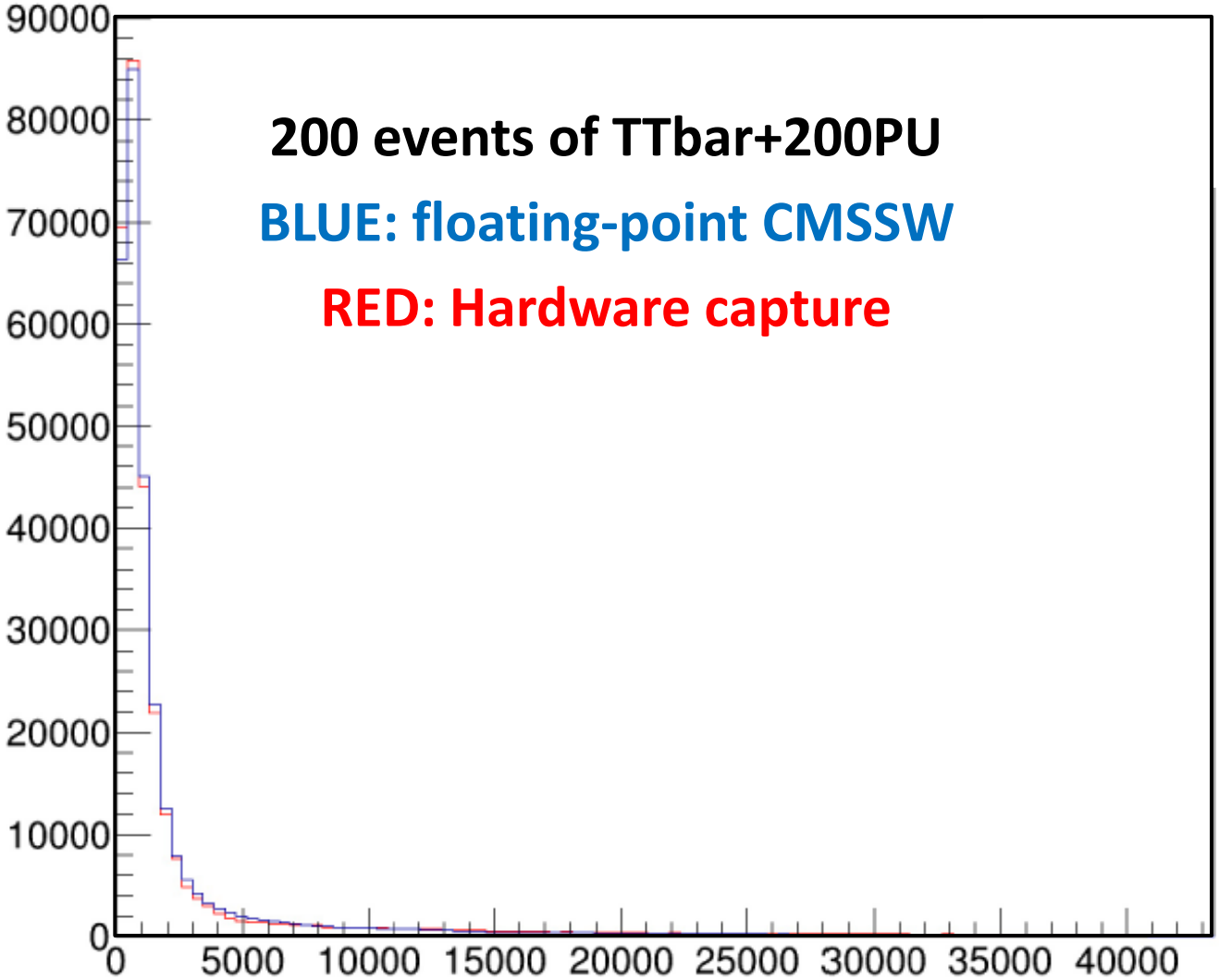
φ_0 resolution (rad)



z_0 resolution (cm)



χ^2 distribution



IMPLEMENTATION

Implementation: The maths

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

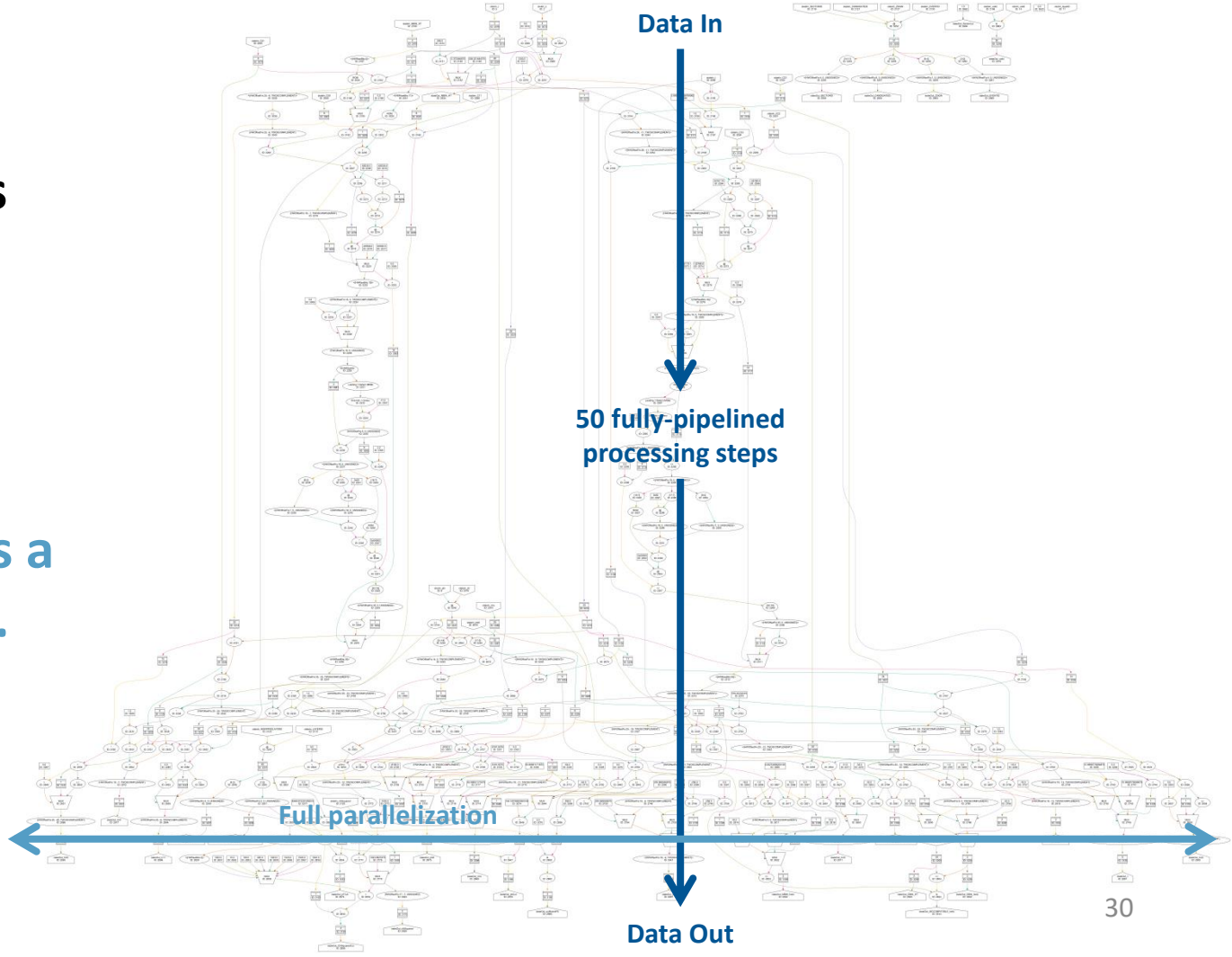
- The Kalman Filter maths is written in a Java-based hybrid HDL/HLS called Max-J
- Massively simplifies the pipelining of the logic:
  - Allows users to focus more on the maths itself
  - Allows non-FPGA experts to contribute to online system
- When compiled provides a netlist and a C++ simulation model
  - Has been integrated into CMSSW

Using MaxCompiler for High Level Synthesis of Trigger Algorithms
S. Summers, TWEPP 2016
http://indico.cern.ch/event/489996/contributions/2210923/

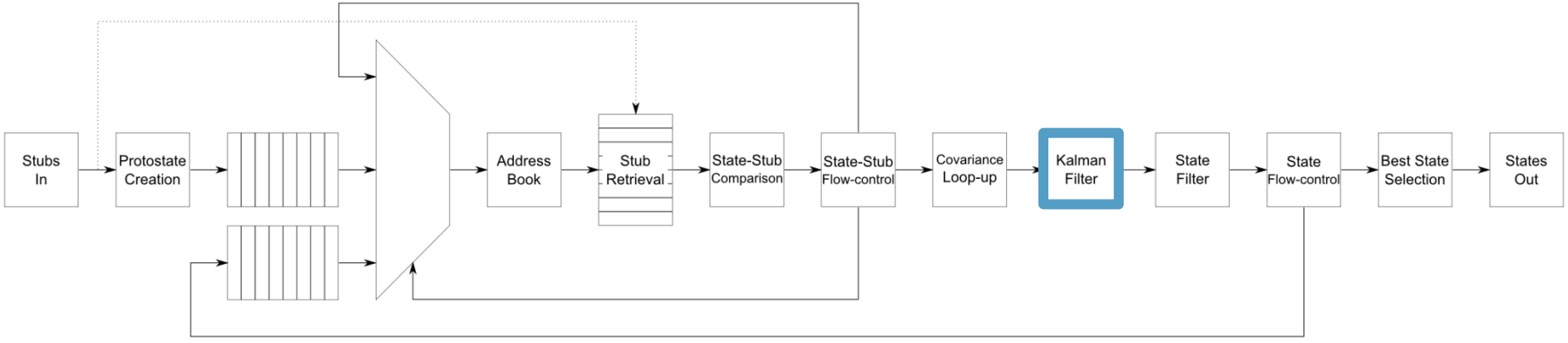
Implementation: The maths

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

- Maxeler tools also provide data-flow visualization tools
- There really is a lot of maths…

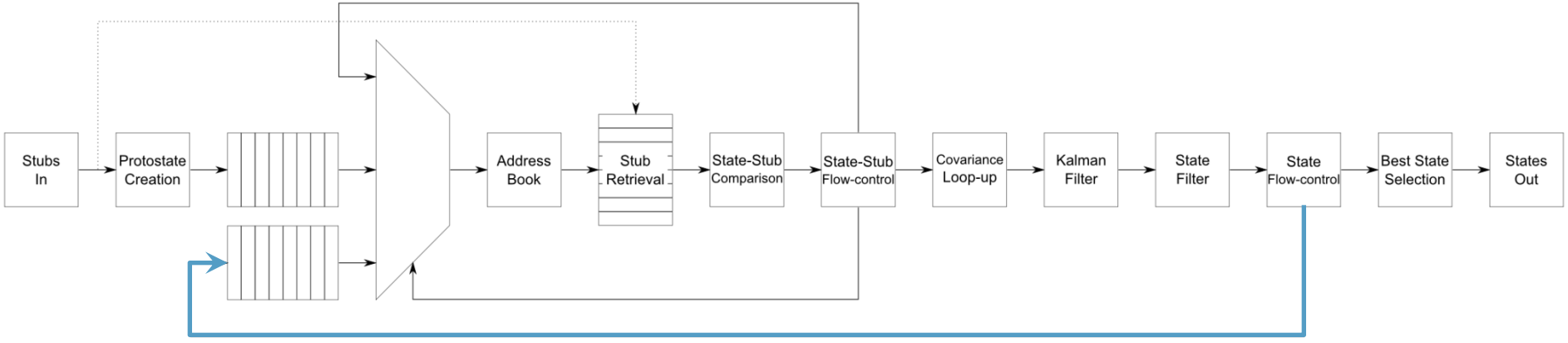


Implementation: The infrastructure



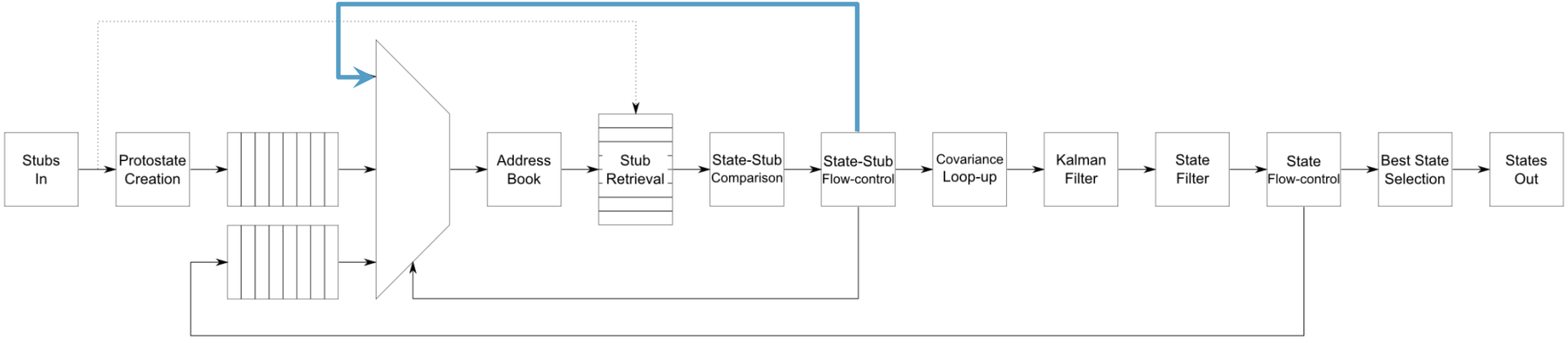
- **The maths is a relatively simple part of a more complex whole**

Implementation: The infrastructure



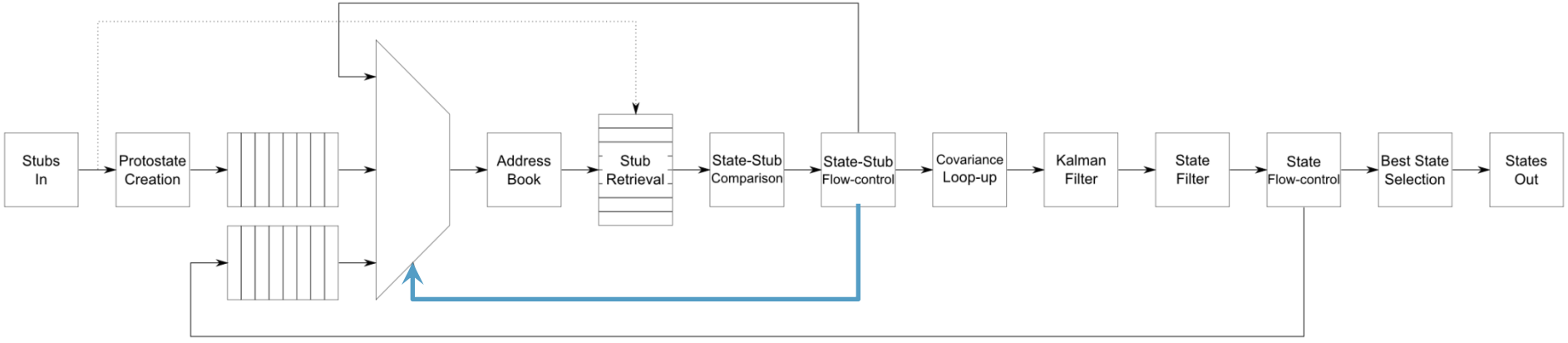
- **Kalman Filter is iterative**

Implementation: The infrastructure



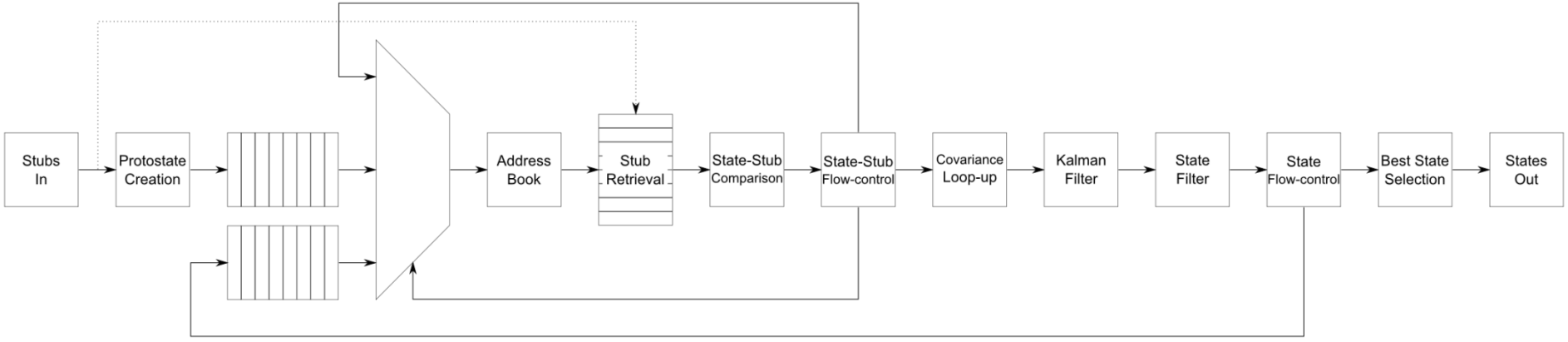
- Kalman Filter is iterative
- Kalman Filter must handle combinatorics

Implementation: The infrastructure



- Kalman Filter is iterative
- Kalman Filter must handle combinatorics
- Kalman Filter data-flow is data-dependent

Implementation: Virtex-7 FPGA

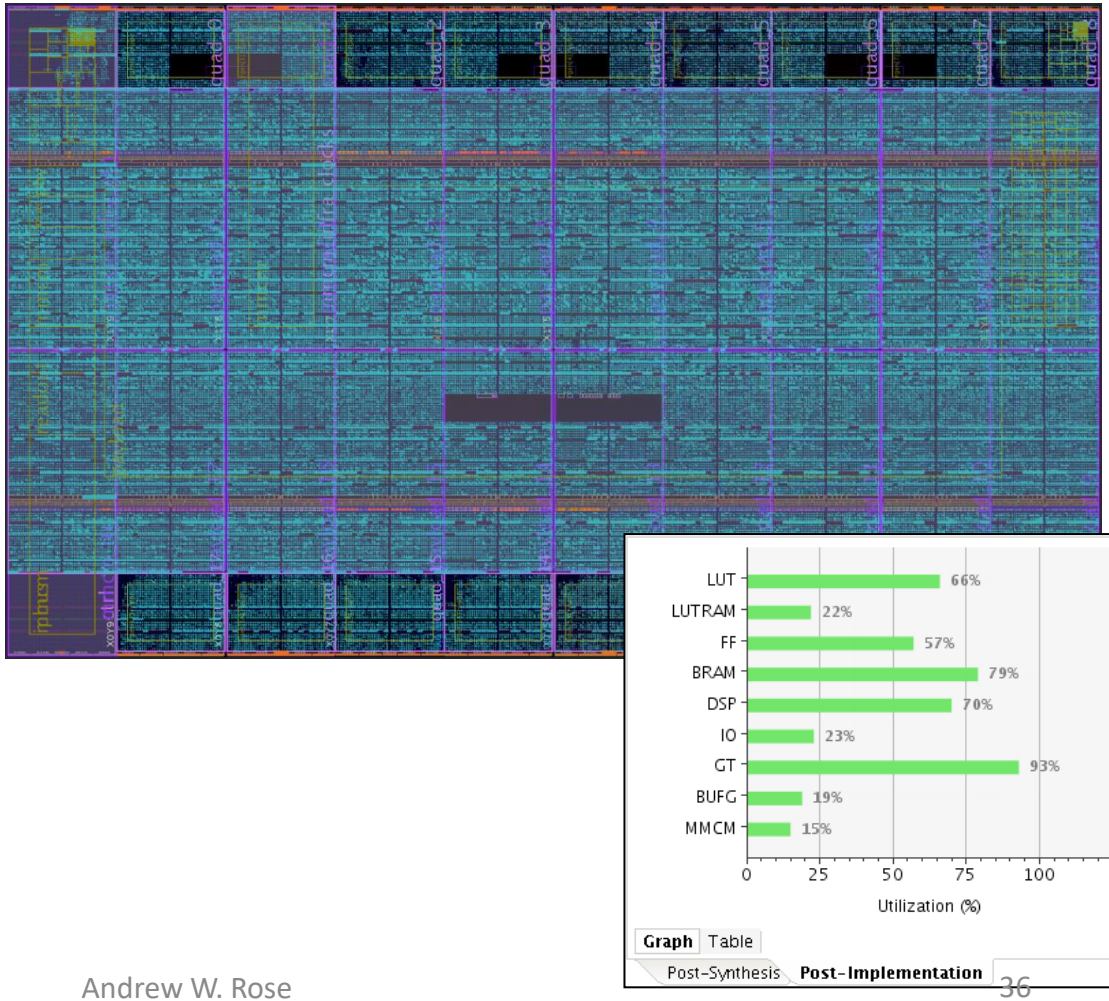


- 36 Independent Kalman chains in an MP7
- Each chain fed by a pair of links at 10Gbps
- Daisy-chain outputs from Kalman chains
- Output up to six quads of links at 10Gbps

Implementation: Virtex-7 FPGA

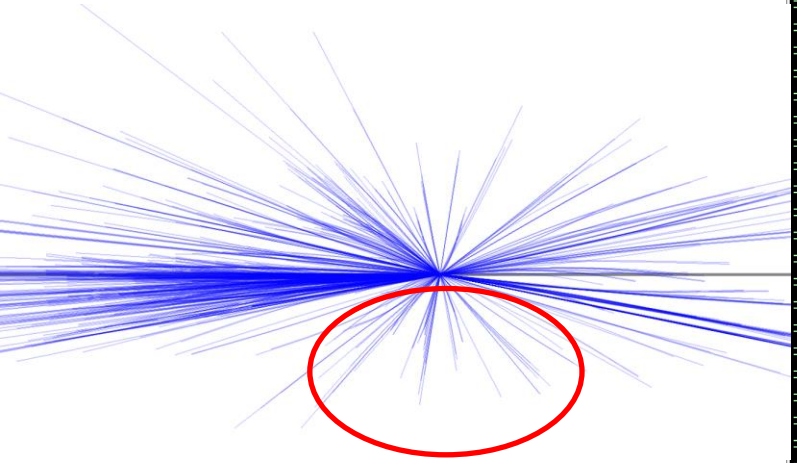
- 36 Independent Kalman Chains in an MP7
- Running at 240MHz

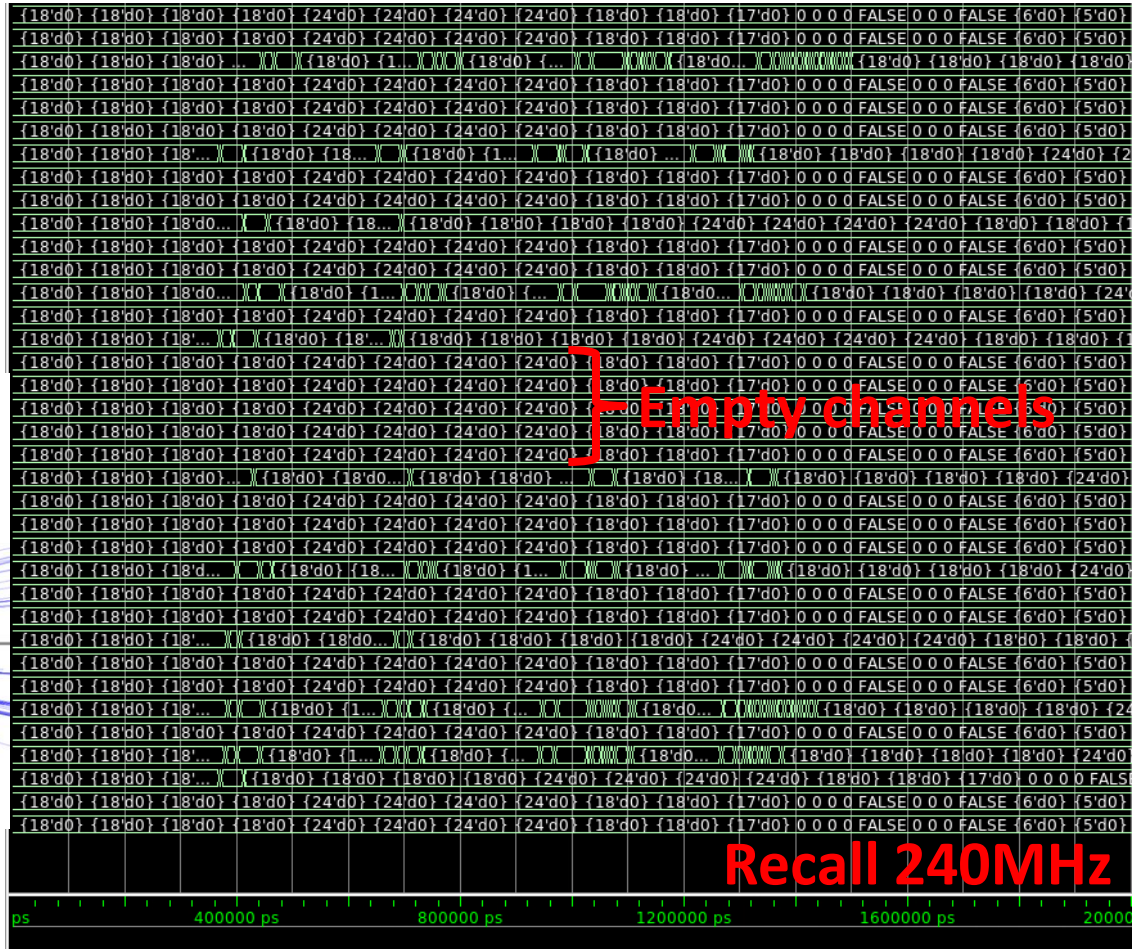
- Per Kalman chain:
  - 1.25% LUTs
  - 1.65% BRAMs
  - 1.95% DSPs



Is there spare capacity?

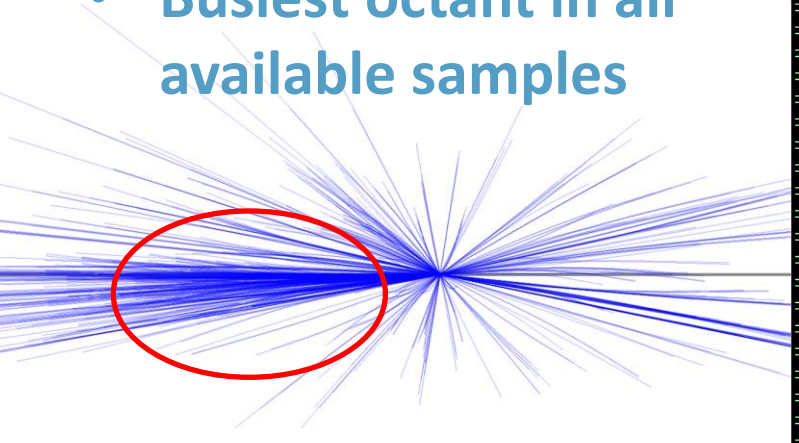
- **Typical TTbar+200PU: Massively underutilized**

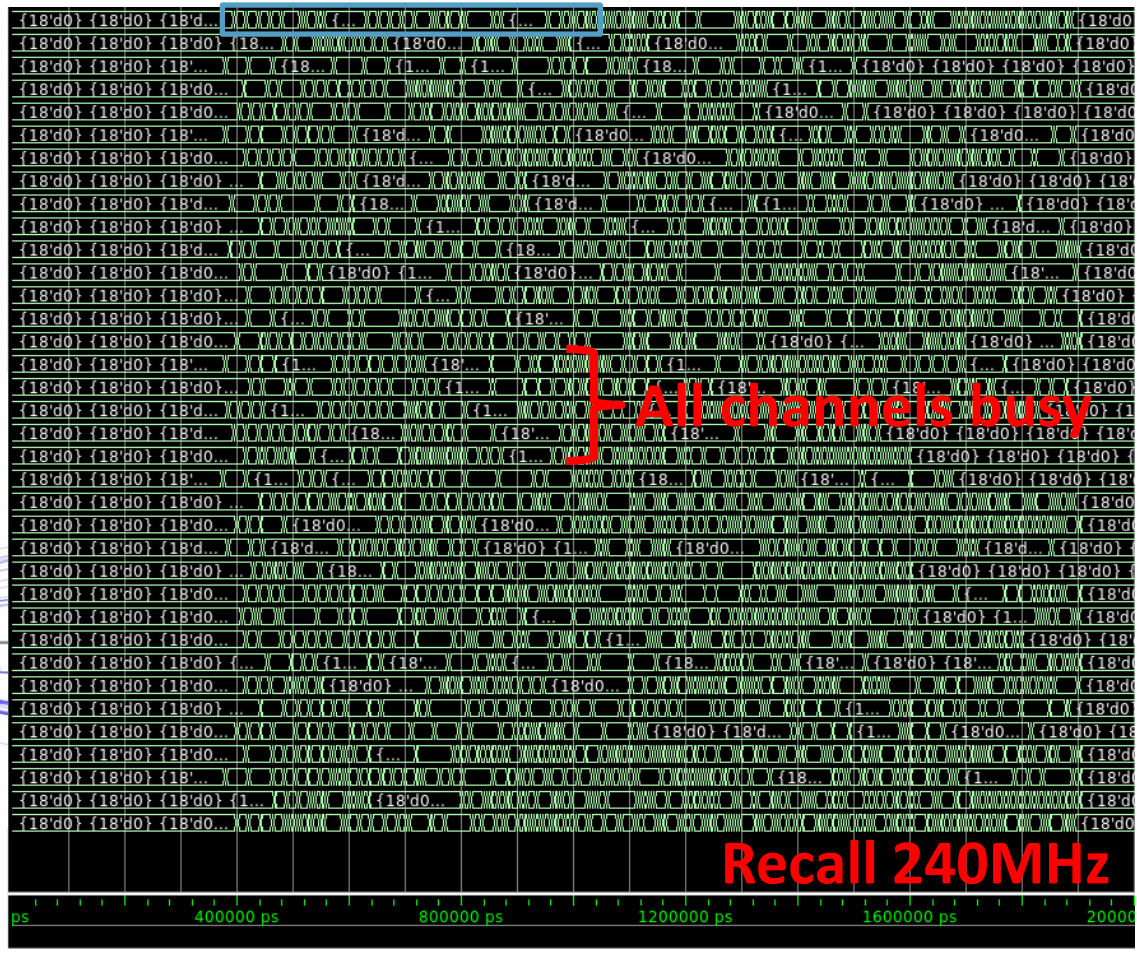




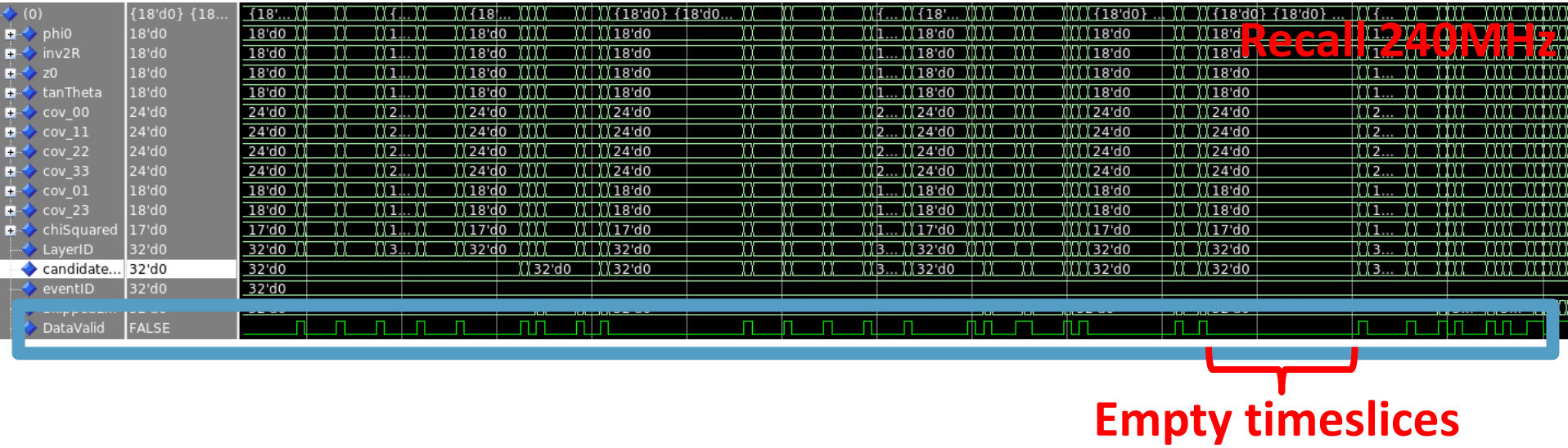
Is there spare capacity?

- TTbar+200PU, jet centre: Much more satisfactory utilization
- Busiest octant in all available samples





Is there spare capacity?



- Busiest TTbar+200PU, jet centre available:
  - Timeslice utilization still ~20-25%
- Current design can handle >80% utilization
- Could be brought to 100%

Final read out & Latency

- Recall, the Kalman filter is iterative, results in combinatorics, and has a data-flow dependent on the data itself: By itself it has variable latency. At first sight
  - This is horrible for firmware engineers
  - This is horrible also for the trigger

Final read out & Latency

- Recall, the Kalman filter is iterative, results in combinatorics, and has a data-flow dependent on the data itself: By itself it has variable latency. At first sight
  - This is horrible for firmware engineers
  - This is horrible also for the trigger
- Turn what could be a horrible problem into a strength!
  - Take every state on all iterations
  - Accumulate the states with the lowest χ^2/DOF
  - Output the best after a fixed accumulation period

Final read out & Latency

- **What does this mean in practical terms?**
  - **you are not waiting for a fit to converge but have a perfectly valid, if inaccurate, estimate of the track parameters from the outset**
- **This is a massive strength of the iterative KF over global fitting approaches**

Final read out & Latency

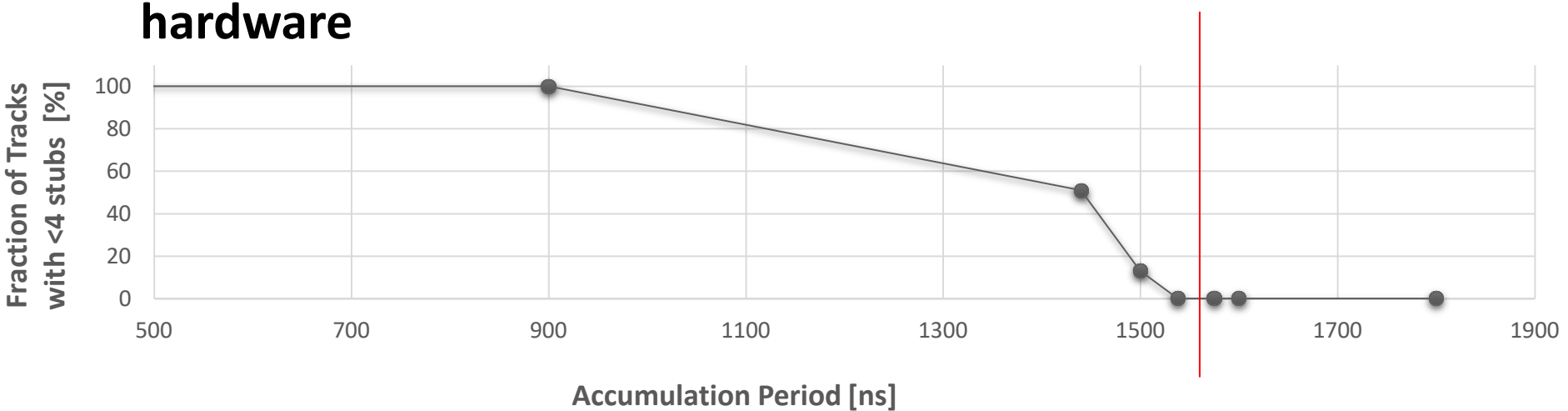
- **If a track has not completed iterating by the timeout, (say because a black hole flooded the tracker with tracks) every track is still read out, just with lower precision:**

The trigger will always get the tracks!

- **In addition, the χ^2 & DOF are encoded in the output so the trigger can handle them with appropriate level of trust**

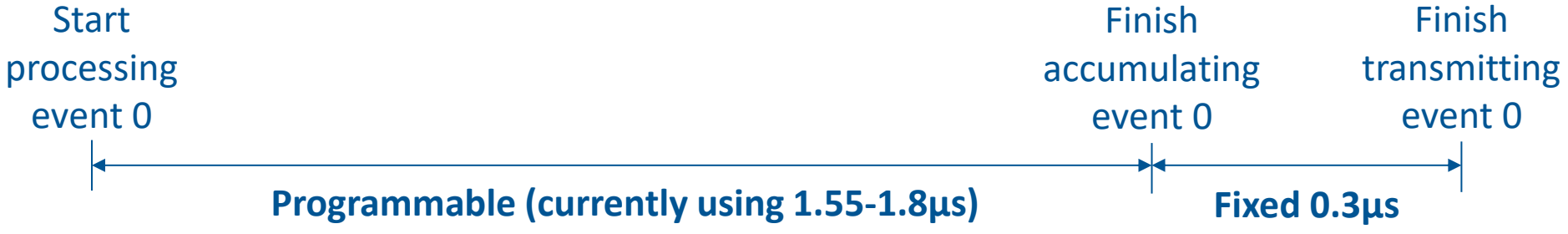
Final read out & Latency

- Algorithm latency can be dynamically set and can be tuned
- Accumulation period was chosen to be 1.8µs from the arrival of the first frame of an event
  - This figure was set by looking by eye at TTbar + 200PU in ModelSim
- Recently updated to 1.55µs after preliminary studies in hardware



Final read out & Latency

- **Transmission period fixed to 300ns (for 2 output channels)**
  - **Truncation less than 0.1% for TTbar + 200PU**



FUTURE WORK

Future work: Impact of tilted geometry

- **The Kalman filter:**
  - **uses all available stub information**
  - **fully handles the errors through the covariance matrices**
- **There should be minimal effect on resolution from using the tilted geometry**
- **Simply requires the updating of the covariance matrix look-up tables (2 block-RAMs per Kalman Chain)**

Future work

- **We have several alternative parameterizations which software indicates may better handle the differences between barrel and endcap, but no time to implement in firmware up to now**
- **Input and experience from the tracker software community would be very welcome here!**

Future work

- Currently we do not take multiple-scattering into account because of lack of systematic software studies, but have the firmware handles to do so:
  - Additive term in the matrix maths, currently set to 0
  - No need for larger matrices, additional Mults, etc.
- Preliminary studies indicate inclusion will improve efficiency further, especially for electrons
- Input and experience from the tracker software community would be very welcome here!

Future work

- With additional latency, could add a second pass “smoothing step” which could improve resolution (usually recommended)
- Will be porting our code to Ultrascale: Can we run faster? Can we make use of UltraRAMs?

- Applying the Kalman Filter to FPGA-based hardware acceleration in the HLT (part of the grant funding the industrial partnership with Maxeler)

Science & Technology Facilities Council

Future work: more speculative

- The Kalman filter is highly performant:
  - Benchmark the KF against increasingly extreme inputs
  - Is it performant enough to remove the need for Hough Transform altogether?
    - Recall, current input too sparse for efficient utilization
    - Potential to reduce latency and increase performance?
  - Modular, fully-TM architecture would make it possible to test this in current demonstrator
  - Development time and advances in technology on our side

Conclusion

The Kalman Filter achieves a performance comparable to the CMSSW floating point in current technology

---

The Kalman Filter is highly performant with slack even when processing the centre of TTbar jet + 200PU

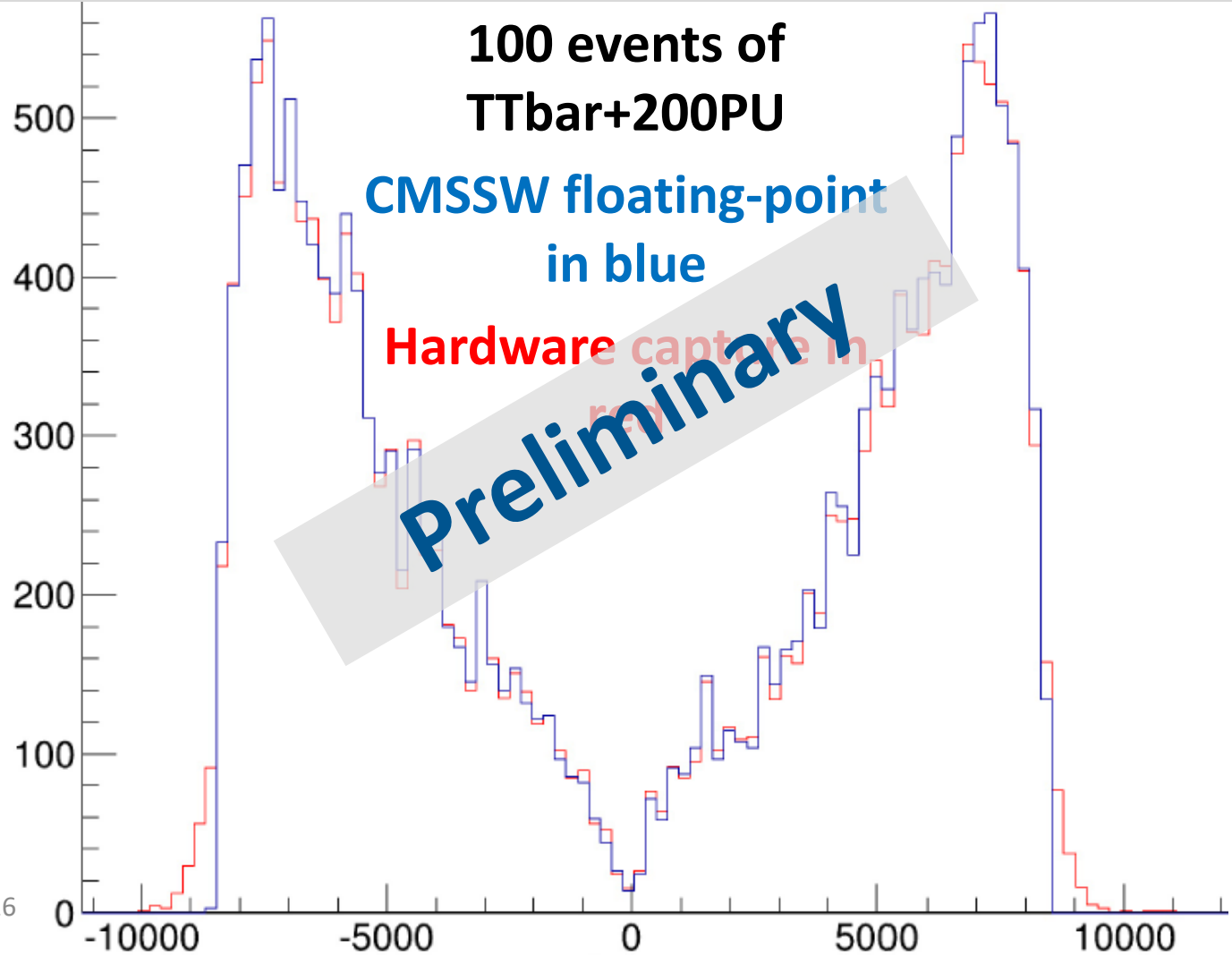
The trigger receives the best estimate for all tracks after a programmable accumulation period

---

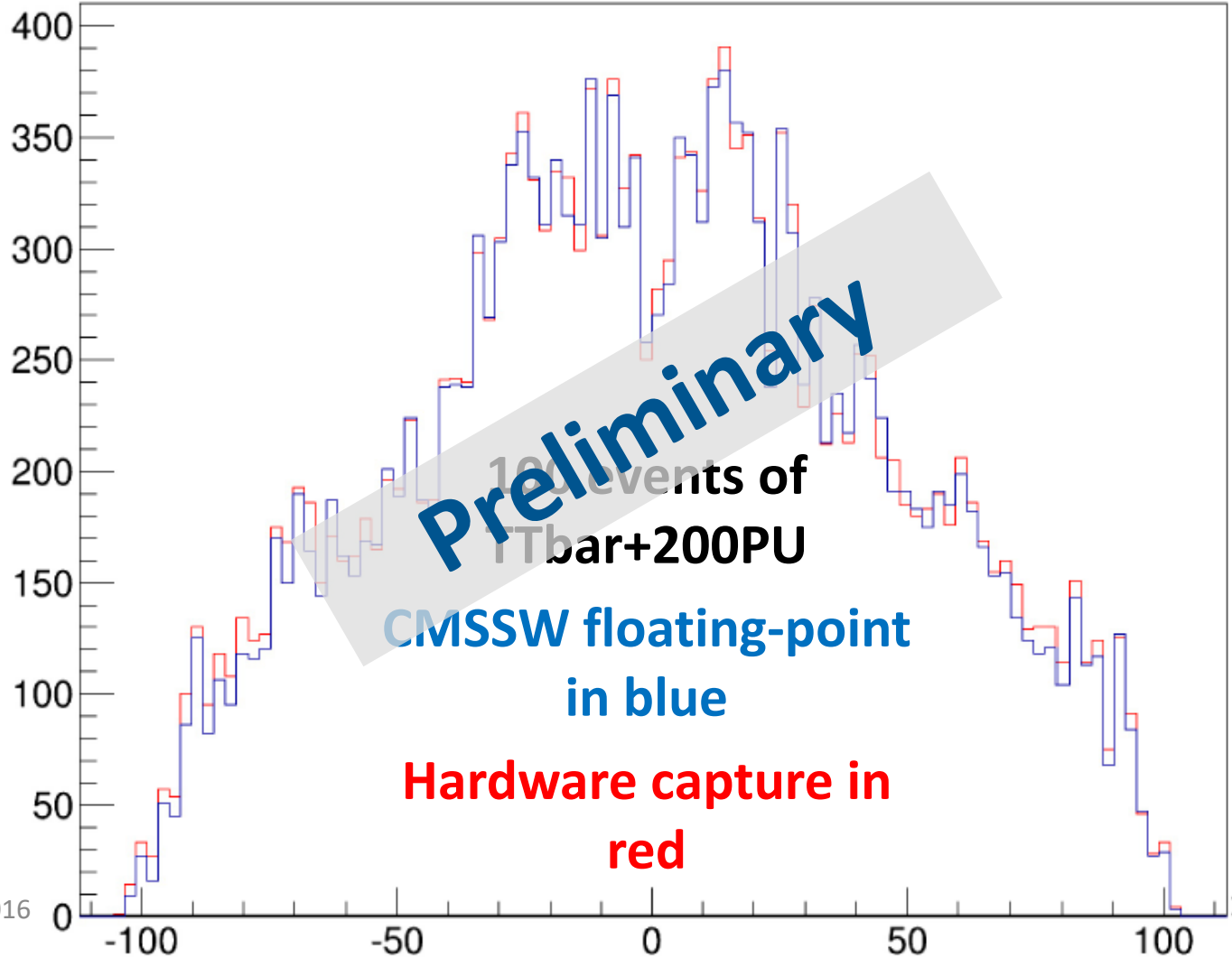
Plenty of scope for further improvements and involvement from others, especially tracking community

SPARES

$1/2r_{curv}$ distribution (HWU)

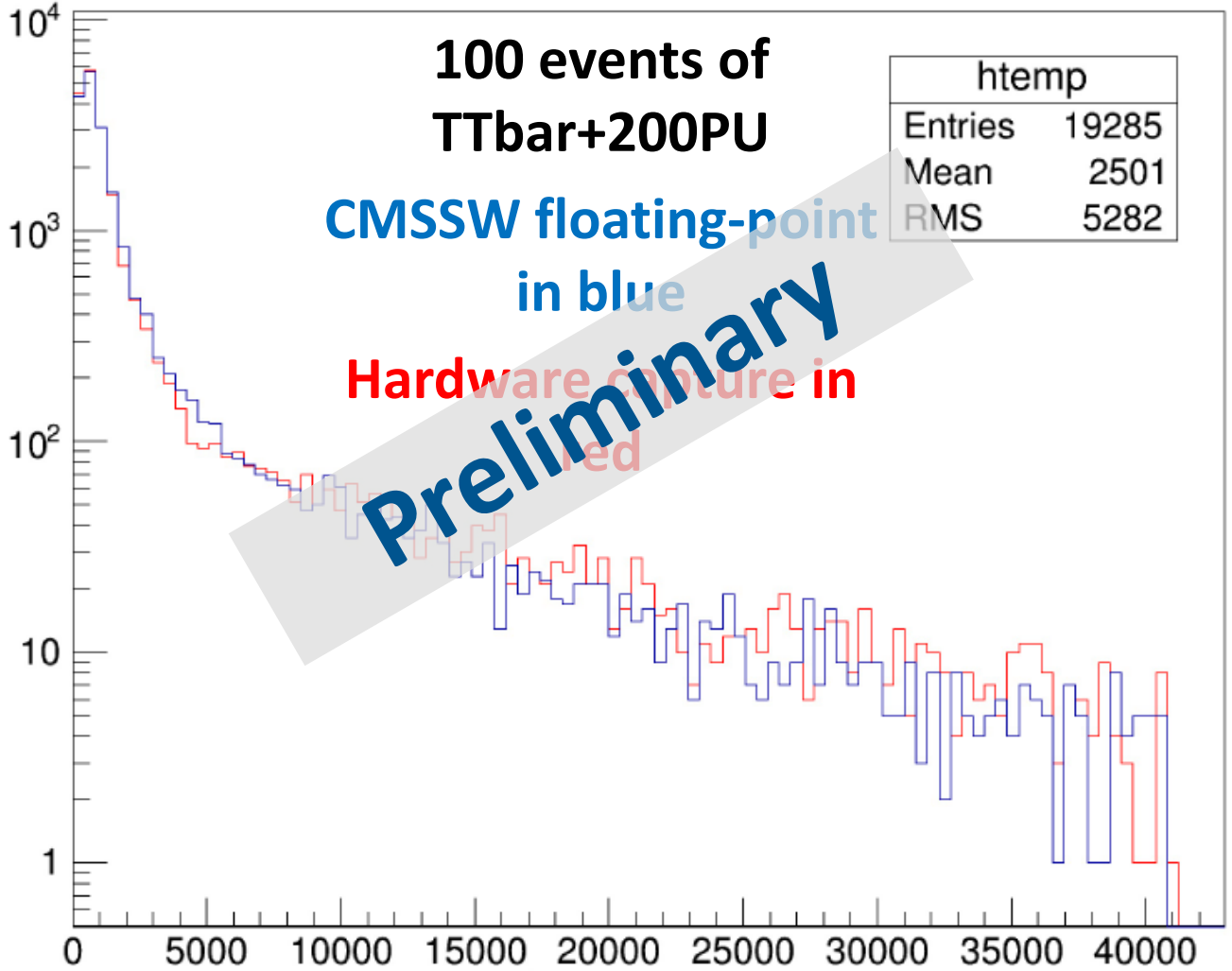


Relative φ_0 distribution (HWU)



χ^2 distribution

Please note logarithmic axis



q/p_T resolution (1/GeV)