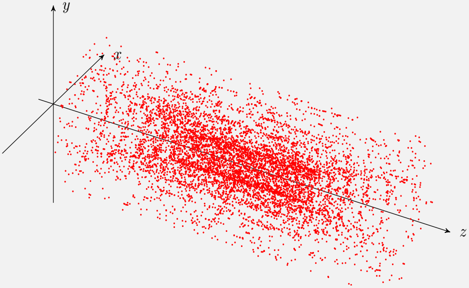
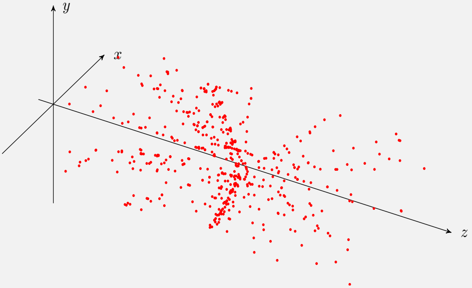
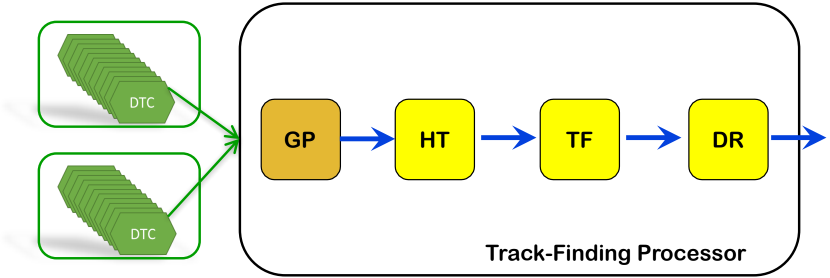
Track Finding: Geometric Processor & Hough Transform

Thomas Schuh | 08.12.2016
(KIT+RAL)

ON BEHALF OF TMTT GROUP



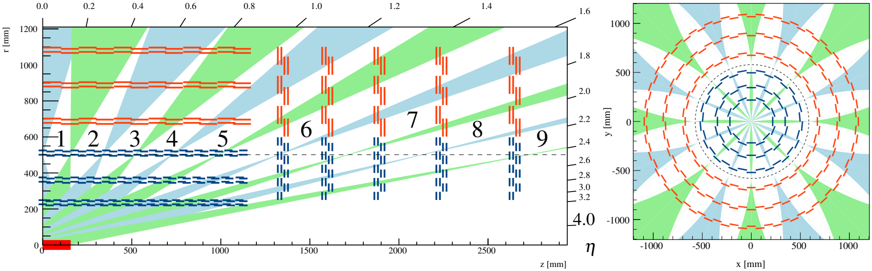



DTC
DTC
GP
HT
TF
DR
Track-Finding Processor

Geometric Processor (GP)

Divide and Conquer

- we subdivide each tracker octant in $2\,\phi \times 18\,\eta$ sectors
- and perform subsequently track-finding independently in each sector in parallel



The Geometric Processor:

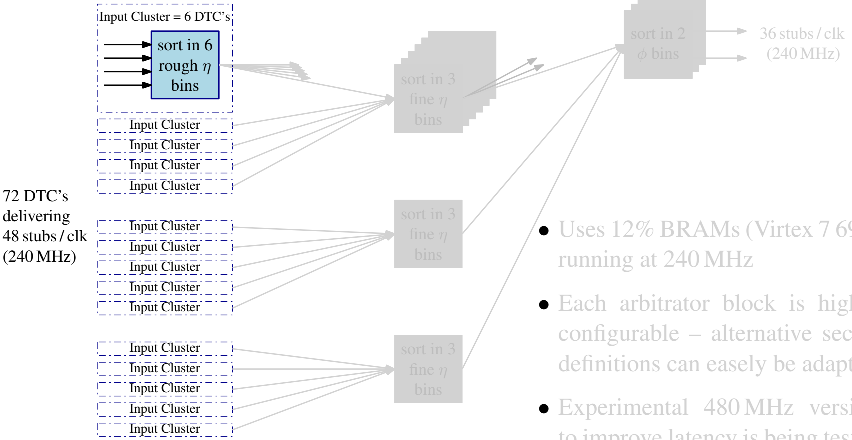
- assigns stubs to sectors
  - also divides each η sector into 2 virtual sub-sectors, and records which one(s) each stub is in
  - ensure tracks found are $\sim$consistent with line in r-z plane
- formats the stub data in a way that is convenient for the subsequent track-finding
- routes all stubs in a given sector to dedicated output links

FPGA-Based GP Implementation – Routing Block

- the GP routes stubs from 72 inputs (one per DTC) to 36 outputs (one per sector)
- Routing happens in three steps:

  rough η sorting $\rightarrow$ fine η sorting $\rightarrow$ ϕ sorting

(Each arrow corresponds to a connection which transports 1 stub / clk)

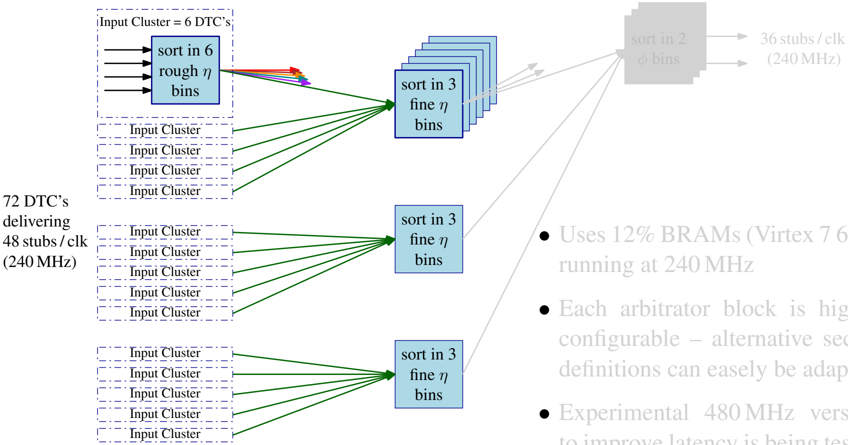


- Uses 12% BRAMs (Virtex 7 690) running at 240 MHz
- Each arbitrator block is highly configurable – alternative sector definitions can easely be adapted
- Experimental 480 MHz version to improve latency is being tested

FPGA-Based GP Implementation – Routing Block

- the GP routes stubs from 72 inputs (one per DTC) to 36 outputs (one per sector)
- Routing happens in three steps:
  rough η sorting $\rightarrow$ fine η sorting $\rightarrow$ ϕ sorting

(Each arrow corresponds to a connection which transports 1 stub / clk)

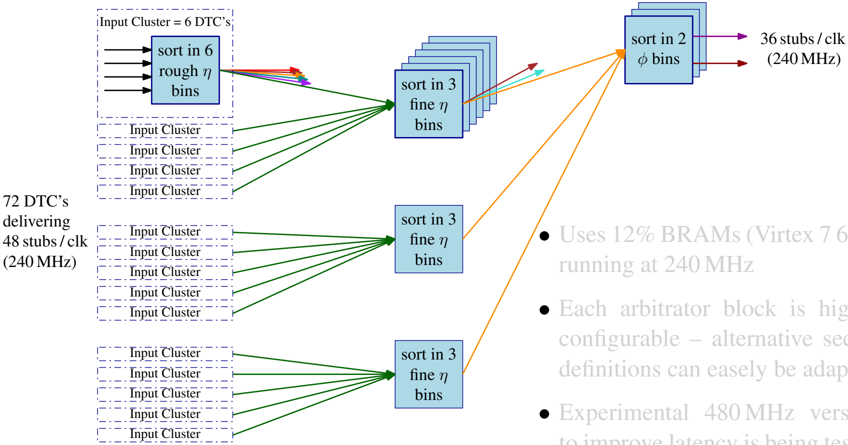


- Uses 12% BRAMs (Virtex 7 690) running at 240 MHz
- Each arbitrator block is highly configurable – alternative sector definitions can easely be adapted
- Experimental 480 MHz version to improve latency is being tested

FPGA-Based GP Implementation – Routing Block

- the GP routes stubs from 72 inputs (one per DTC) to 36 outputs (one per sector)
- Routing happens in three steps:
  rough η sorting $\rightarrow$ fine η sorting $\rightarrow$ ϕ sorting

(Each arrow corresponds to a connection which transports 1 stub / clk)

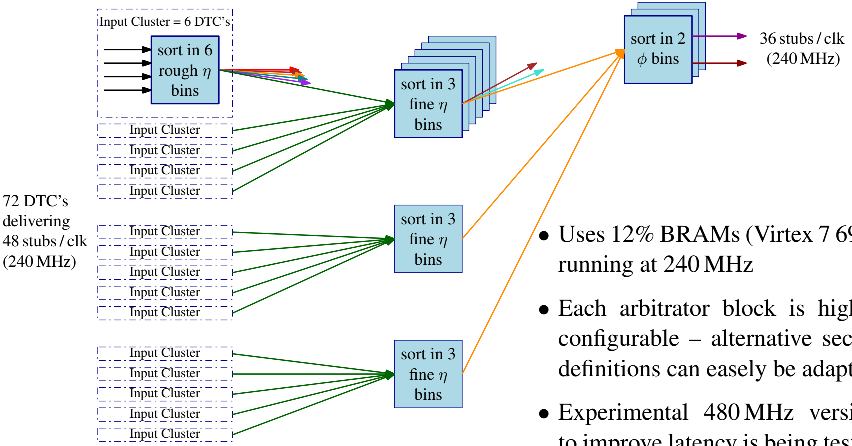


- Uses 12% BRAMs (Virtex 7 690) running at 240 MHz
- Each arbitrator block is highly configurable – alternative sector definitions can easely be adapted
- Experimental 480 MHz version to improve latency is being tested

FPGA-Based GP Implementation – Routing Block

- the GP routes stubs from 72 inputs (one per DTC) to 36 outputs (one per sector)
- Routing happens in three steps:
  rough η sorting $\rightarrow$ fine η sorting $\rightarrow$ ϕ sorting

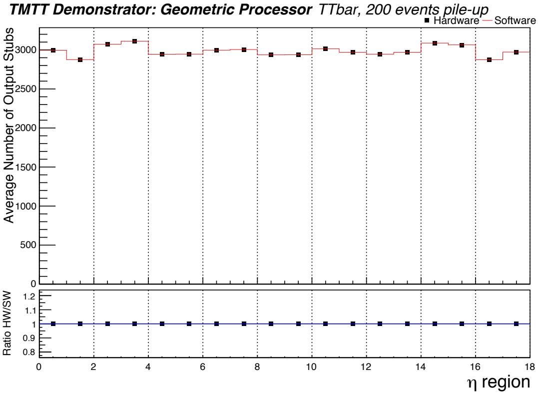
(Each arrow corresponds to a connection which transports 1 stub / clk)

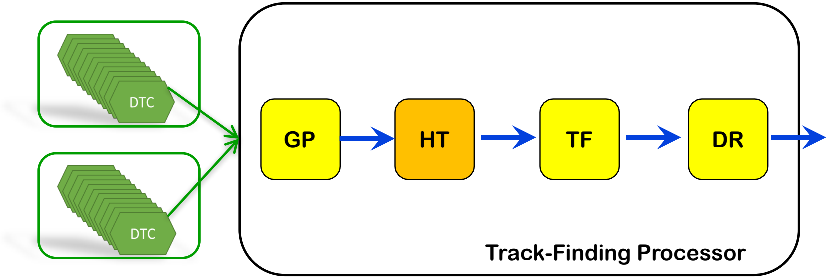


- Uses 12% BRAMs (Virtex 7 690) running at 240 MHz
- Each arbitrator block is highly configurable – alternative sector definitions can easely be adapted
- Experimental 480 MHz version to improve latency is being tested

FPGA-Based GP Implementation – Results

- plot compares # tracks / η region in h/w (black dots) vs s/w (red lines)
- 200 $t\bar{t}$@200PU events were used
- first in to first out latency is constant at 310 ns

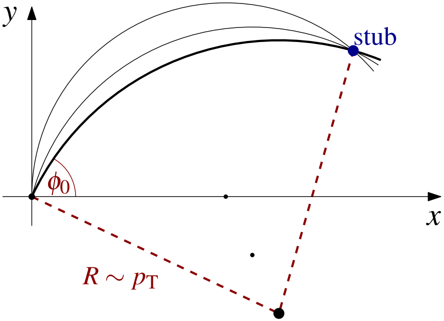



DTC
DTC
GP
HT
TF
DR
Track-Finding Processor

Hough Transform (HT)

Hough Transform – Theory

- search for primary tracks in the r-ϕ plane
- infinite number of circles $\left(\phi_0, \frac{q}{p_T}\right)$ consistent with beam-line & any individual stub position (r, ϕ)

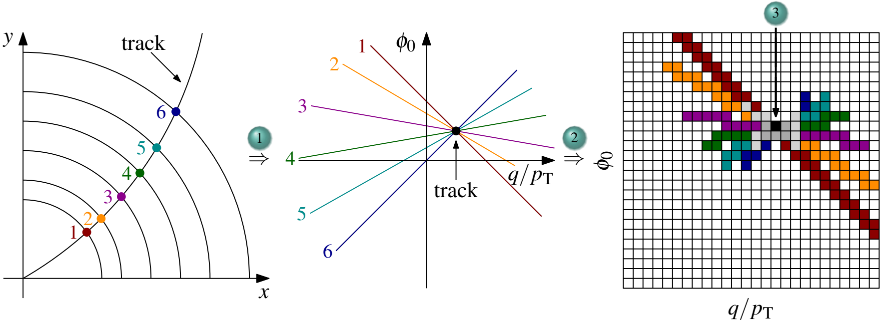


- they must obey constraint:

$$\phi_0 \quad \approx \quad \phi + \frac{q}{p_T} \times r$$

- **stub positions corresponds to straight lines in the track parameter plane**

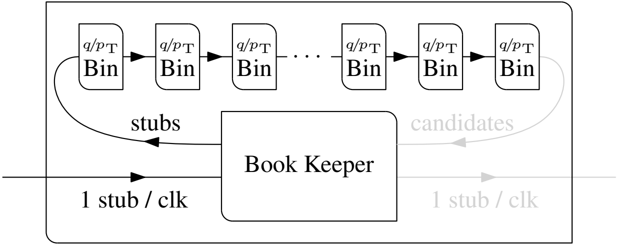
Hough Transform – Algorithm

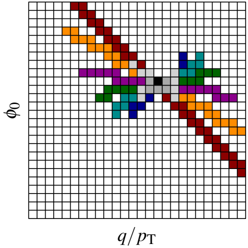


1. for each stub calculate ϕ_0 for each q/p_T
2. fill the stub into corresponding cells of an array with 32×64 cells in $q/p_T \times \phi_0$
   - ignore q/p_T values inconsistent with the p_T estimate of the stub
3. define cells with stubs in at least 4 or 5 tracker layers as track candidates
   - 4 layer threshold used to cope with dead layer (cooling loop failure) or barrel-endcap transition region (where a track can not cross more then 5 layers)

FPGA-Based HT Implementation – Overview

- array is implemented as a pipeline, it processes one stub per clock cycle (240 MHz)
- first step is the filling of the array
- second step is the readout of track candidates

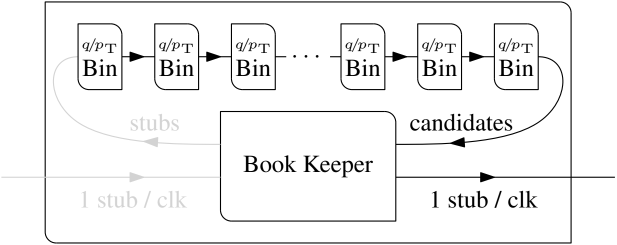


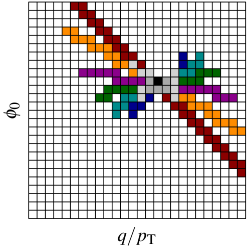


- Book Keeper unpacks stub data from input link, which then propagate to each q/p_T Bin in turn
- track candidates found by the Bins propagate back to the Book Keeper, which transmits them over output link

FPGA-Based HT Implementation – Overview

- array is implemented as a pipeline, it processes one stub per clock cycle (240 MHz)
- first step is the filling of the array
- second step is the readout of track candidates





- Book Keeper unpacks stub data from input link, which then propagate to each q/p_T Bin in turn
- track candidates found by the Bins propagate back to the Book Keeper, which transmits them over output link

FPGA-Based HT Implementation – Bin

- each Bin represents a q/p_{T} column in HT array

Hough Transform

- gets ϕ_0 at left boundary
- calculates ϕ_0 at right boundary

ϕ_0 Buffer

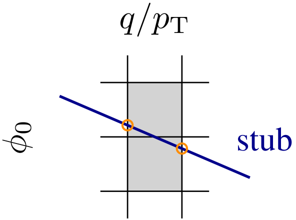
- duplicates stubs if it belongs to two cells

Track Builder

- sorts stubs in ϕ_0 cells
- marks ϕ_0 cells with stubs from at least 4 or 5 layers within one η subsector for read-out

Hand Shake

- controls the read-out of track candidates



FPGA-Based HT Implementation – Bin

- each Bin represents a q/p_{T} column in HT array

Hough Transform

- gets ϕ_0 at left boundary
- calculates ϕ_0 at right boundary

ϕ_0 Buffer

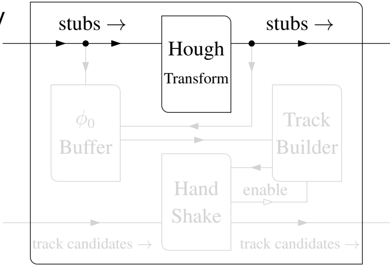
- duplicates stubs if it belongs to two cells

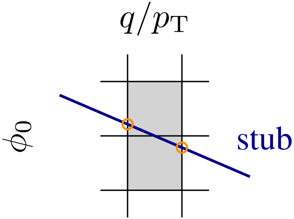
Track Builder

- sorts stubs in ϕ_0 cells
- marks ϕ_0 cells with stubs from at least 4 or 5 layers within one η subsector for read-out

Hand Shake

- controls the read-out of track candidates





FPGA-Based HT Implementation – Bin

- each Bin represents a q/p_{T} column in HT array

Hough Transform

- gets ϕ_0 at left boundary
- calculates ϕ_0 at right boundary

ϕ_0 Buffer

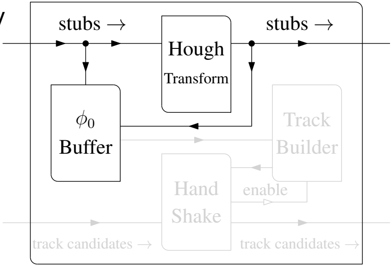
- duplicates stubs if it belongs to two cells

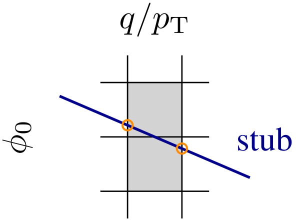
Track Builder

- sorts stubs in ϕ_0 cells
- marks ϕ_0 cells with stubs from at least 4 or 5 layers within one η subsector for read-out

Hand Shake

- controls the read-out of track candidates





FPGA-Based HT Implementation – Bin

- each Bin represents a q/p_T column in HT array

Hough Transform

- gets ϕ_0 at left boundary
- calculates ϕ_0 at right boundary

ϕ_0 Buffer

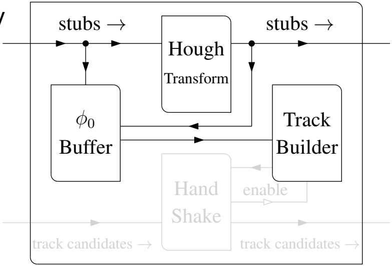
- duplicates stubs if it belongs to two cells

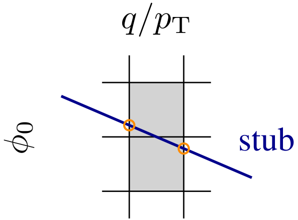
Track Builder

- sorts stubs in ϕ_0 cells
- marks ϕ_0 cells with stubs from at least 4 or 5 layers within one η subsector for read-out

Hand Shake

- controls the read-out of track candidates





FPGA-Based HT Implementation – Bin

- each Bin represents a q/p_T column in HT array

Hough Transform

- gets ϕ_0 at left boundary
- calculates ϕ_0 at right boundary

ϕ_0 Buffer

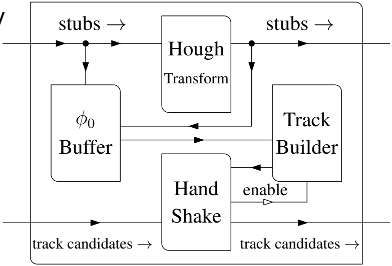
- duplicates stubs if it belongs to two cells

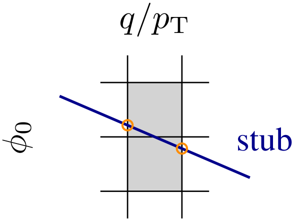
Track Builder

- sorts stubs in ϕ_0 cells
- marks ϕ_0 cells with stubs from at least 4 or 5 layers within one η subsector for read-out

Hand Shake

- controls the read-out of track candidates

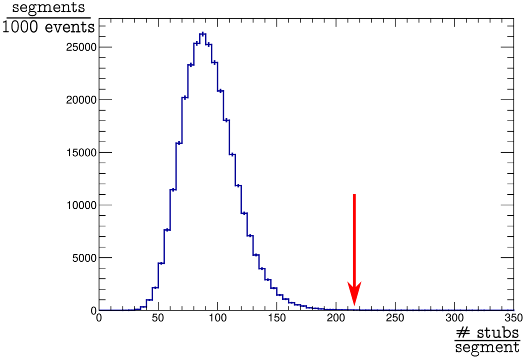




FPGA-Based HT Implementation – Truncation

Input Truncation

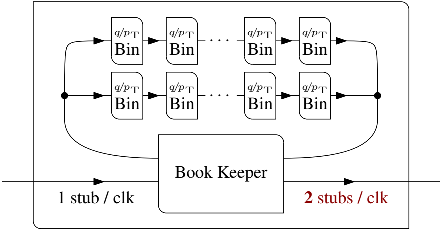
- **one** HT array processes **one new stub per clock cycle** (240 MHz)
- that implies a lot of arrays working in parallel
  per octant: 2 $\phi \times 18\ \eta$ = 36 independent arrays
- 2 MP7s needed, since 18 arrays fit into one MP7
- each array has 900 ns processing time (36 BX), that corresponds to 216 stubs
- lost stubs due to input truncation in $t\bar{t}$@200PU measured to be at per mille level



FPGA-Based HT Implementation – Truncation

Output Truncation

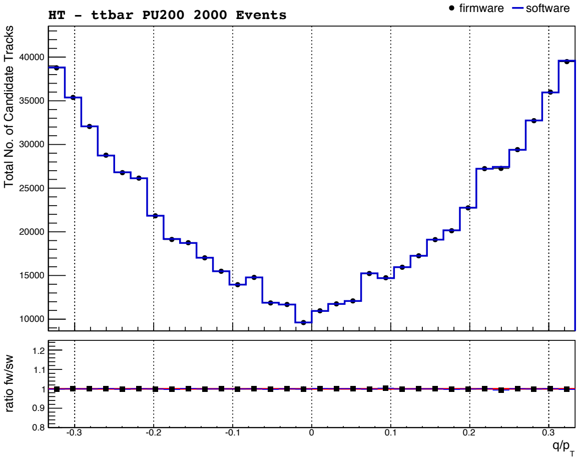
- on average the HT reduces the number of stubs by one order of magnitude
- problematic are only local fluctuations, mainly caused by jets
- therefore we balance the output load
- output bandwidth can easily be increased by splitting the chain of bins



- we split the bins in 6 chains
- and interleave the chains of 3 different not neighbouring not opposite η sectors
- efficiency loss due to truncation in $t\bar{t}$@200PU measured to be at per mille level

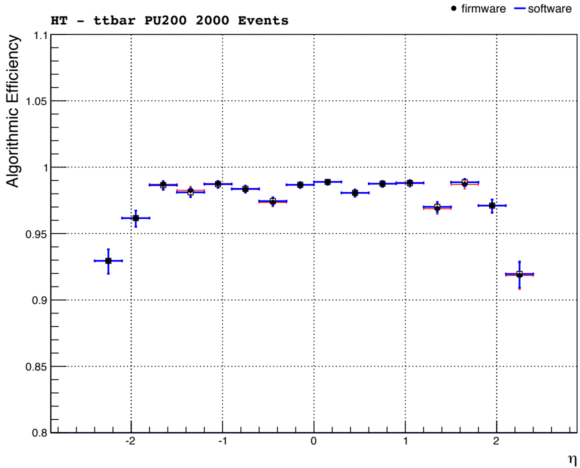
FPGA-Based HT Implementation – Results

- plots compares # tracks / q/p_{T} bin in h/w (black dots) vs s/w (blue lines)
- 2000 $t\bar{t}$@200PU events were used
- first in to first out latency is constant at 1025 ns (dominated by the 900 ns TMP)

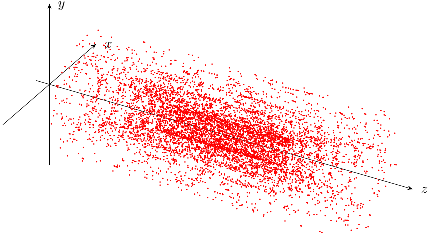


FPGA-Based HT Implementation – Results

- plots compares algorithmic efficiency in h/w (black dots) vs s/w (blue lines)
- 2000 $t\bar{t}$@200PU events were used
- hardware achieved 97.94 % algorithmic efficiency



Summary



- high track finding efficiency within 1.5 µs achieved using the Hough Transform (GP first in to HT first out)
- Capability to perform L1 track finding under high luminosity conditions has been demonstrated with current technology.