Hough transform implementation on FPGA for event filtering of HL-LHC

Kazuki Todome (INFN Bologna)

Fabrizio Alfonsi (INFN Bologna)

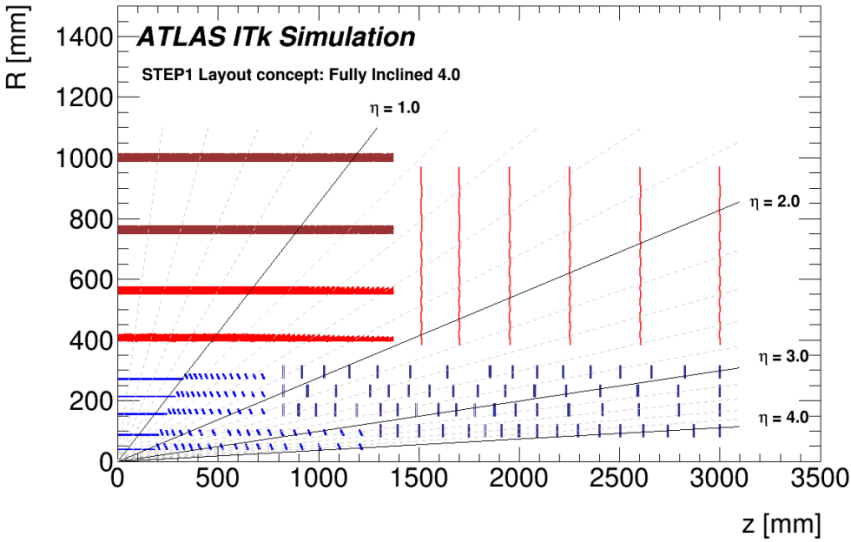
On behalf of the ATLAS collaboration

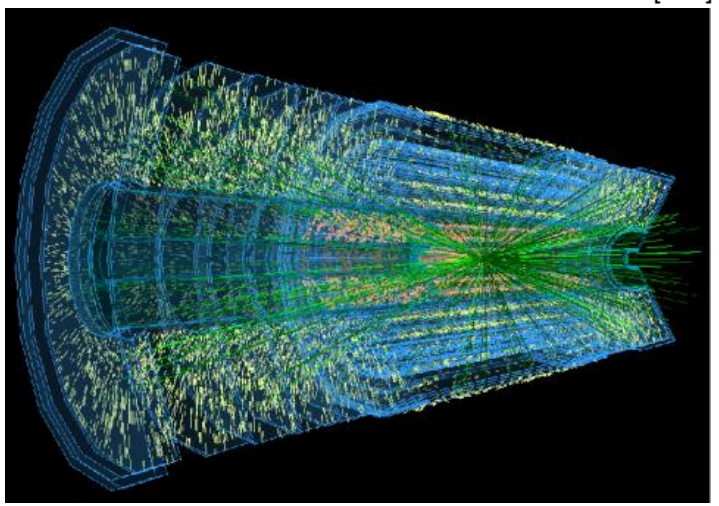
International Conference on High Energy Physics 2022

8th Jul 2022 Bologna

Introduction – ATLAS in HL-LHC

- Challenging environment of HL-LHC (7.5x luminosity)
  - More collisions (40 to 200 pileup)
  - Higher granularity detectors → More hit clusters
  - Data taking trigger using calorimeter + Muon spectrometer @ 1 MHz
  - Event Filtering system skims the events using data including Inner Tracker (ITk) down to 10 kHz
  - Large number of cluster evaluation in short time
- Event Filter system works as follows:
  1. Construct online clusters
  2. Group clusters which may construct track: "Pattern Recognition" (Scope of this talk)
  3. Perform online fitting of the track
  4. Evaluate fitted track for the event skim
     - See Viviana's talk (link)
- We need to reduce fitting process time ~ possible combination of clusters





Introduction – Hough Transform

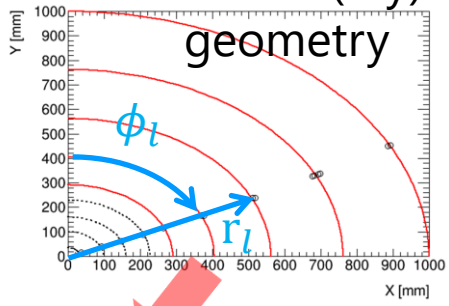
- Inputs: a bunch of clusters with format of ALTAS geometry {(r_l, ϕ_l, η_l, l =layer)}

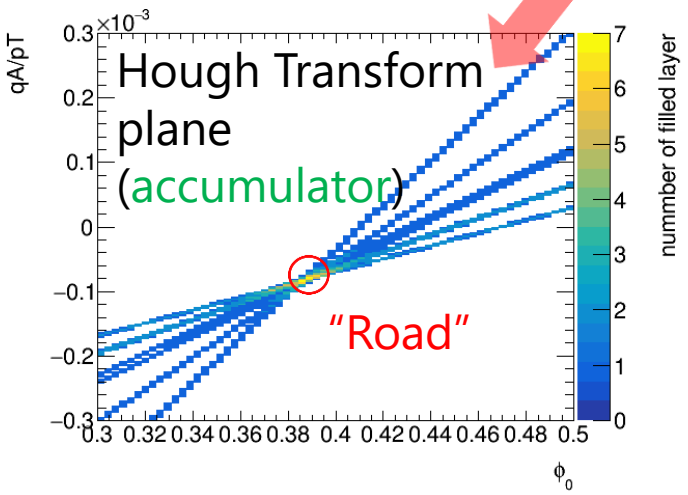
A is constant value
In the ATLAS,
A=3 × 10^{-4} GeV/mm

- Idea of Hough Transform: Convert cluster valuables to phase space of track ($\frac{qA}{p_T}$, ϕ_0)
  using relation in the transverse plane: $\frac{qA}{p_T} = \frac{\phi_0 - \phi_l}{r_l}$

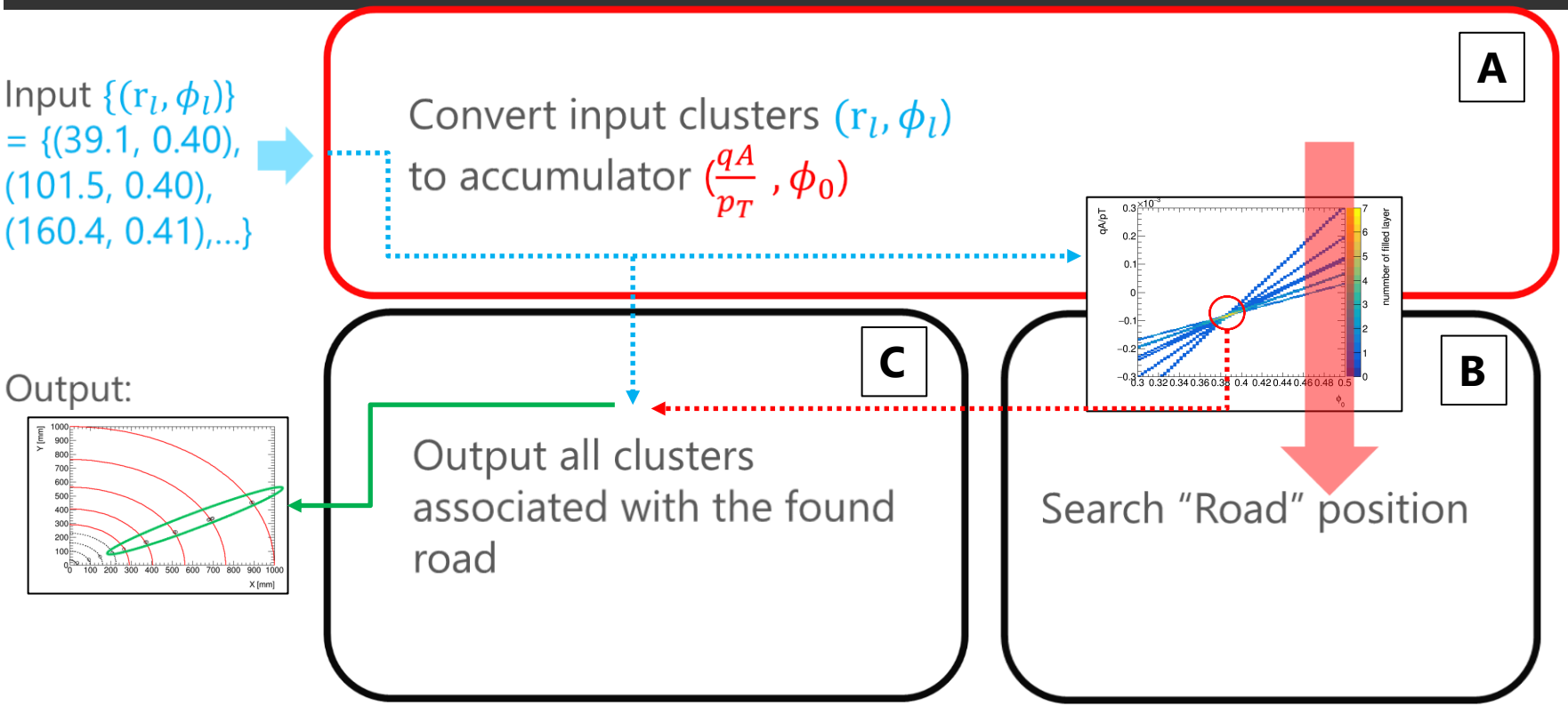
- Steps of Hough Transform
  1. Draw a line for each cluster, on HT plane layer by layer (**accumulator**)
  2. Search more probable track parameters ("Road")
  3. Output all clusters associated with the found road
     → extract only minimum clusters

- Requirements on Hough Transform in Event Filtering
  - Low latency
    - Hardware based, like FPGA implementation is ideal
  - High efficiency/Low number of roads and clusters
  - Low resource (critical!) → serval logic blocks in a board
    - Reduce number of boards and power consumption





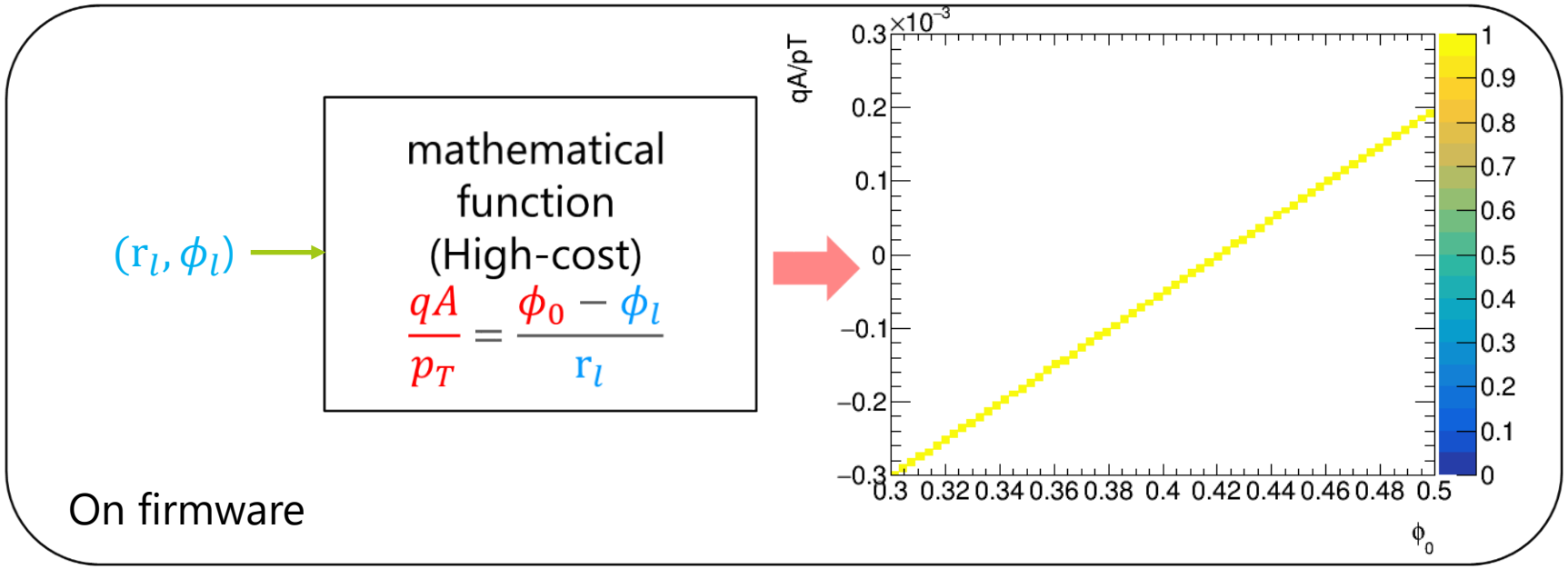
Architecture overview



- Most resource-consuming block "A" takes 400 k LUTs
  ~ more than 90% of available resources (VC709 case)

Previous implementation and limitation

- Previous way of implementation
  - Mathematically evaluate $\frac{qA}{p_T} = \frac{\phi_0 - \phi_l}{r_l}$ for given (r_l, ϕ_l)
  - Point the accumulator bin to be fired
- Due to the large possibility of the operations, this block consumes a lot of resources
  - In contrast to the mathematical operation cost in firmware, it is not a high cost for software
  - But parallel process, such as finding roads, is a high cost for software

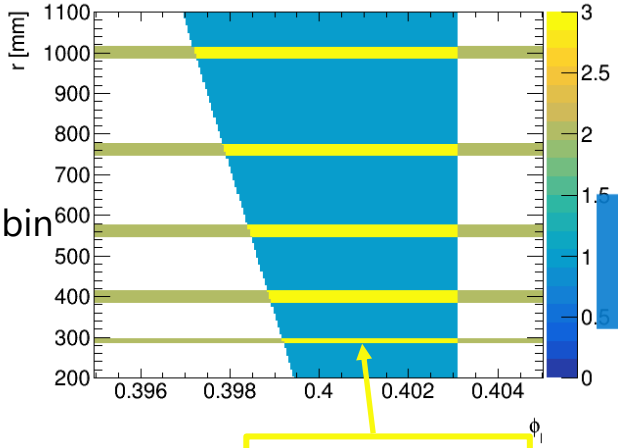


How to reduce resource utilization

- Idea: perform all possible mathematical operations in software, then implement results in firmware
  1. For a each $(\frac{qA}{p_T}, \phi_0)$ bin, evaluate the possible range of (r_l, ϕ_l) to fire the given bin in software
     - Effect of r_l fluctuation is small enough → evaluate only range of ϕ_l
     - This step takes more than a few hours → software itself cannot be implemented as an online system
  2. On firmware, just check if the input ϕ_l is in the range given by software for each $(\frac{qA}{p_T}, \phi_0)$ bin
     - Parallel process: FPGA has strong advantages
     - Some $(\frac{qA}{p_T}, \phi_0)$ bins have exactly same ϕ_l range → additional resource reduction is possible

evaluate range of (r_l, ϕ_l) to fire a $(\frac{qA}{p_T}, \phi_0)$ bin

$(\frac{qA}{p_T} = \frac{\phi_0 - \phi_l}{r_l})$



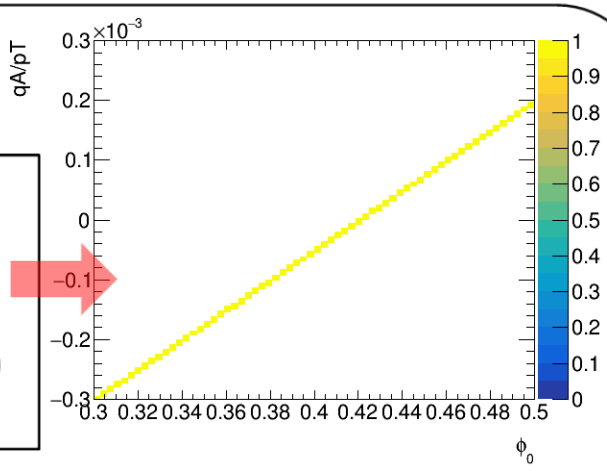
Possible range

On software

Only range

ϕ_l

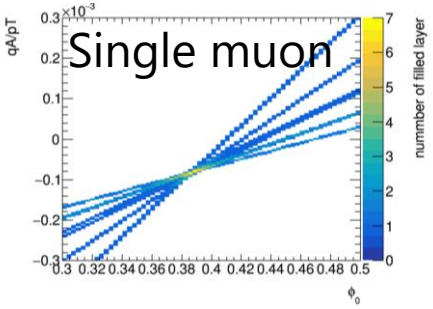
Check range (low cost)

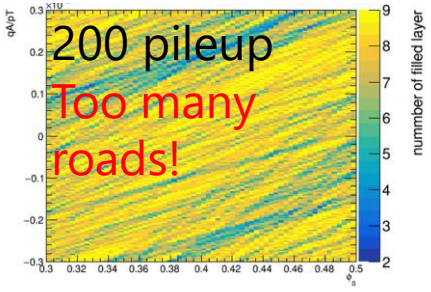


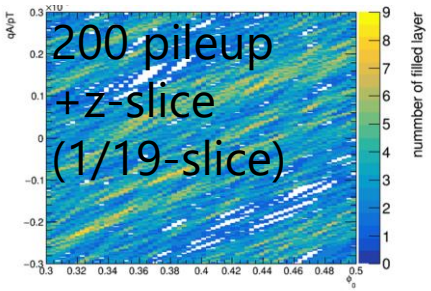
On firmware

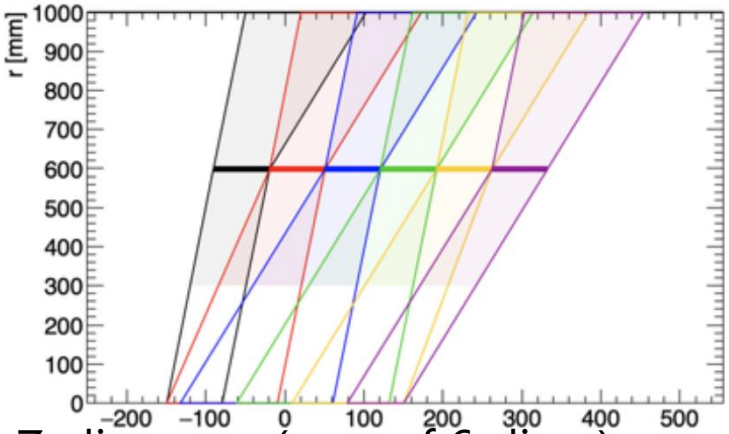
Test setup

- Accumulator size:
  $(\frac{qA}{p_T}, \phi_0)$= ([-3×10^{-3}, 3×10^{-3}] /mm in 216 bins, [0.3, 0.5] in 64 bins)
- Test dataset: single muon events under 200 of the pileup events interpreted as 18 bits of ϕ_l for each layer
- Used layers: subset of ITk layers (in total 8)
- Used board: Xilinx Virtex-7 VC709 (433k LUTs, 866k FFs available)
- Consider "z-slice":
  - At a particular r position, every track passes a position of z
  - By limiting the z position, the range of targeted tracks is also limited
  - Select only clusters that may compose such limited tracks
  - We assume 19 z-slice









Z-slice map (case of 6 slices)

First evaluation: Software base

- Purpose: evaluate the performance with operations that are logically compatible with firmware idea
- Procedure:
  1. Evaluate the range of ϕ_l to fire each $(\frac{qA}{p_T}, \phi_0)$ bin and save this, independently from the test dataset
     - Two ideas were tested: r is fixed to the center, or scan all possible r range
  2. Based on saved file, generate accumulator for each single muon + 200 pileup file
  3. Extract roads and associated clusters
  4. Compare the performance and resource utilization with the original implementation and purely mathematical operations (without digitalization of input ϕ_l)
- Results
  - Fix r option
    - LUTs utilization has been reduced to **less than 8%** of previous implementation
    - Inefficiency is slightly increased
    - Demonstrated in the next slide
  - Scan r option
    - LUTs utilization is ~¼ of previous implementation
    - Inefficiency is quite low
    - #fit and #road are quite large due to duplicated roads
      - Could be reduced by duplication remover

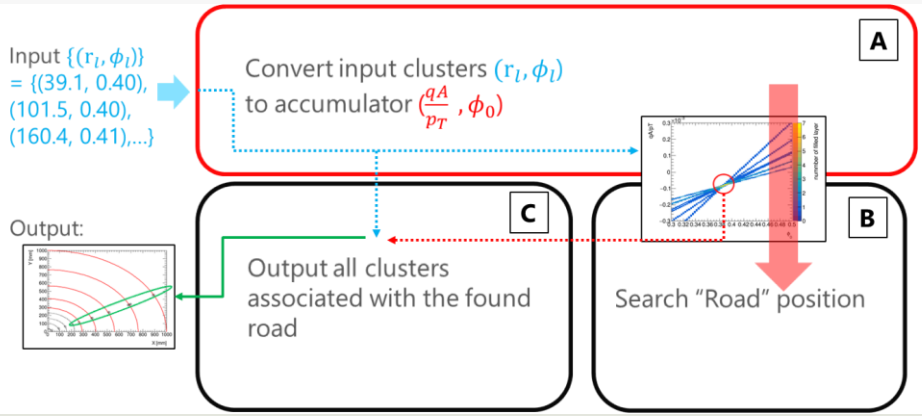
	Previous	math	Fix r	Scan r
#LUTs	400k	---	31k	108k
<#fit>	---	2375	2709	292k
<#road>	---	242	270	4881
Inefficiency	---	4.5%	6.5%	0.36%

Second evaluation: Firmware implementation

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	82131	433200	18.96
LUTRAM	690	174200	0.40
FF	180584	866400	20.84
BRAM	154	1470	10.48
DSP	104	3600	2.89
IO	13	850	1.53
GT	8	36	22.22
BUFG	9	32	28.13
MMCM	3	20	15.00
PCIe	1	3	33.33



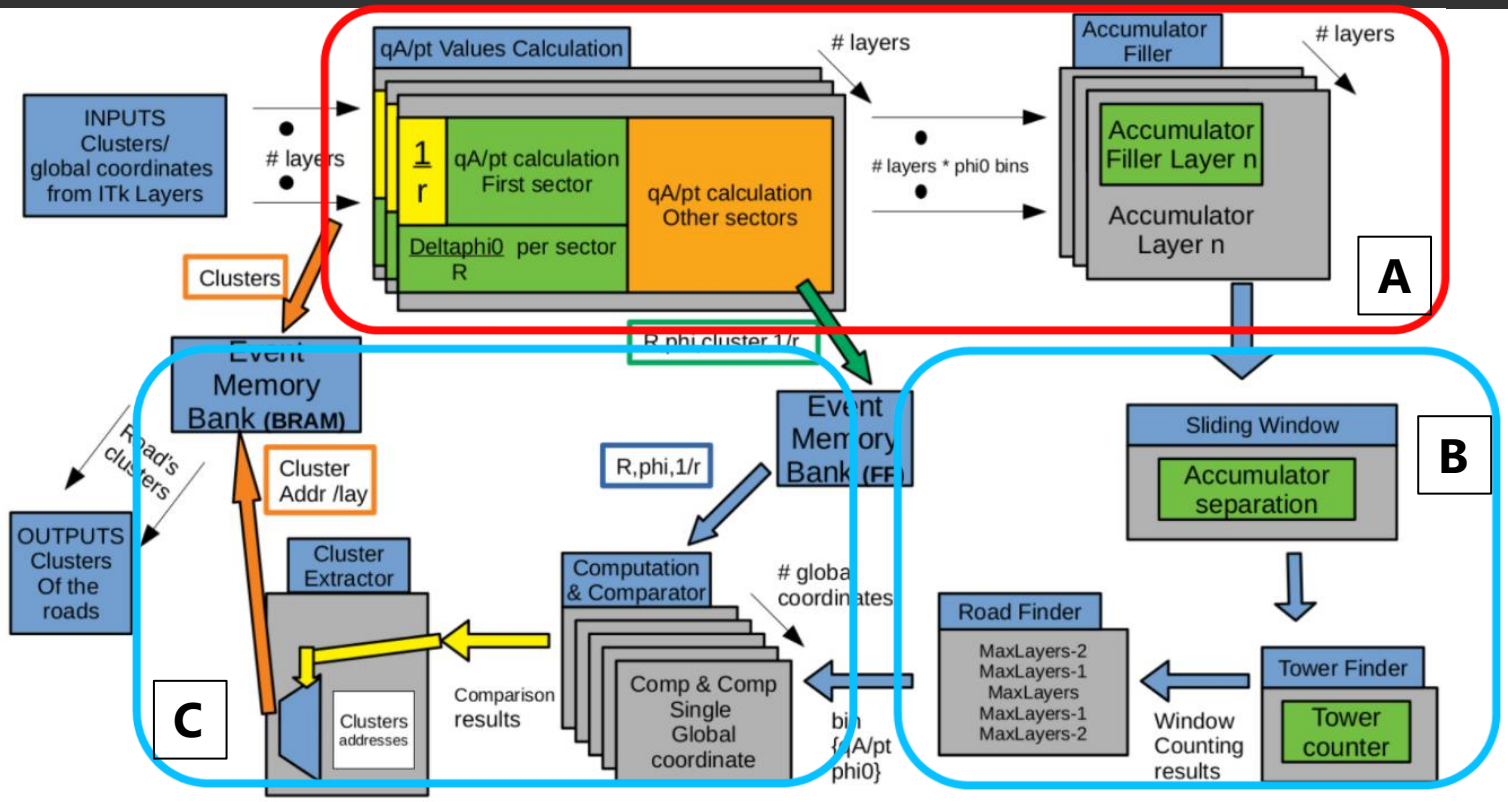
- Evaluated range of ϕ_l is called by firmware code and implemented as LUTs
  - Confirmed that the implemented block works as designed by software
- All blocks work in 250 MHz clk domain
- For the concerned block "A", implementation is achieved with exactly the expected number of LUTs
- The size of resources consumed by block "A" became compatible with ones consumed by block "B"
  - Space for optimization in the block "B"

Summary

- In the challenging environment of HL-LHC, “Pattern Recognition” is one of the important functions in Event Filtering to reduce total processing time and Hough Transform is one of the promising algorithms for this function
- In the previous studies of Hough Transform, the function block to build accumulator consumes 400 k LUTs, which is more than 90% of available resources
- By pre-calculating logic in software, the resource utilization of the concerned block has been reduced to **less than 8%** of the original implementation while keeping its inefficiency low enough
  - This idea is feasible also for any other mathematical operations on the firmware to reduce resource utilization
- The pre-calculated logic has been implemented on the firmware targeting Xilinx Virtex-7 VC709 working with 250 MHz CLK
  - **The implemented block works as designed**
- Plans for possible improvement:
  - Optimization of other logic blocks
  - Investigate a better method to define the range of ϕ_l for each accumulator bin

Backup

Detailed architecture overview



A) Convert input clusters (r_l, ϕ_l) to accumulator $(\frac{qA}{p_T}, \phi_0)$

B) Search "Road" position

C) Wrap up all clusters associated with the found road and output them

- Most resource consuming block "A" takes 400 k LUTs ~ more than 90% of available resource (VC709 case)