Karlsruhe Institute of Technology

Track finding on UT4

Kai Unger – kai.unger@kit.edu

AI Trigger Group at Belle II

KIT ITIV
- Marc Neu
- Kai Unger
- Jürgen Becker

KIT ETP
- Lea Reuter
- Greta Heine
- Stefkova Slavomira
- Torben Ferber

MPI & TUM
- Felix Meggendorfer
- Elia Schmidt
- Christian Kiesling
- Alois Knoll

Agenda

- Track segments for the displaced vertex trigger and graph building for GNN tracking on FPGAs
  - By Marc Neu (ITIV)
- Graph Neural Network based Track Finding in the CDC
  - By Lea Reuter (ETP)
- Object-condensation in the Belle II Electromagnetic Calorimeter
  - By Isabel Haide (ETP)
- 3D Track finding for the Neural z-Trigger
  - By Christian Kiesling
- How to optimize TS-LUTs and deep learning perspectives for the STT
  - By Felix Meggendorfer
- Optimization of the DVT algorithms for economic Hardware Implementation
  - By Elia Schmidt

Outline

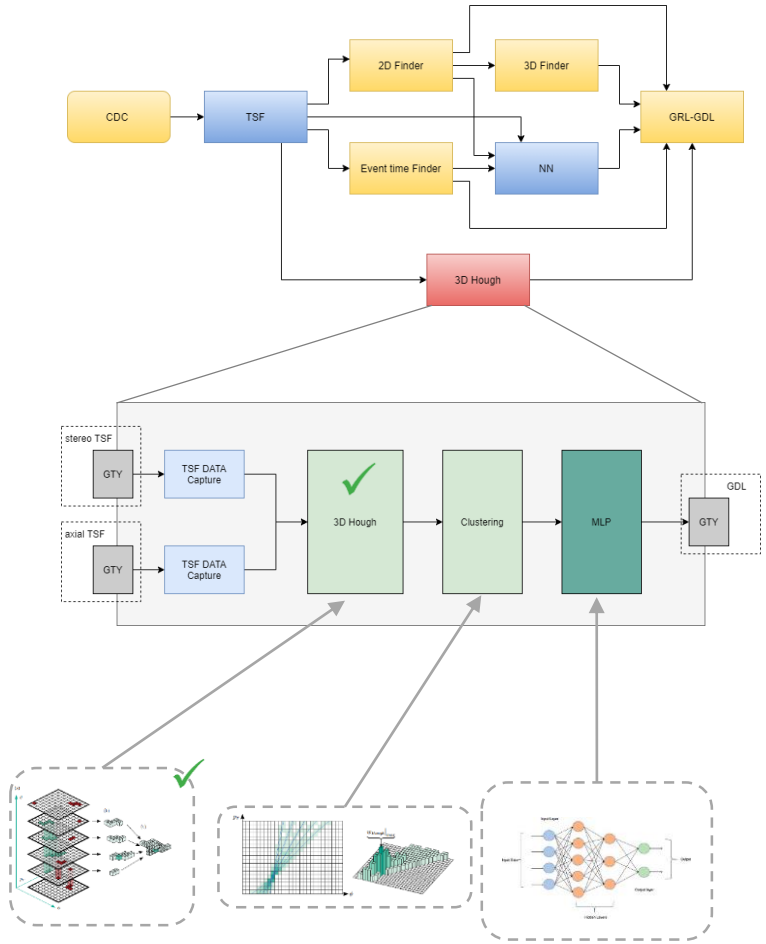
- 3D Hough
- Multi Hough Displaced Vertex Trigger
- CNN Displaced Vertex Trigger

3D Hough (Jan)

- Higher efficiency than the 2D finder through 3D Hough preprocessing
- Better fake track suppression
- More latency due to own preprocessing
- **Goal: an improved z-Vertex resolution and strong suppression of fake tracks**
- More detailed in Christians talk

3D Hough

- 3D Hough implementation done (HLS)
- Clustering implementation ongoing (VHDL)
- MPL need to be trained
  - More hidden layer
  - Implementation with hls4ml



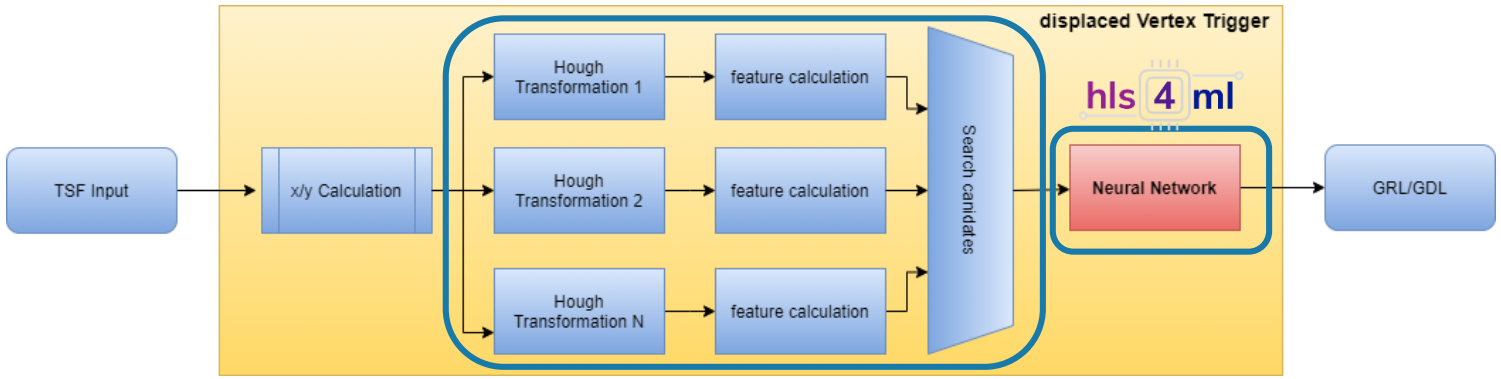
Todos

- First prototype with old 2D Finder as preprocessing
- Clustering
- Neural Network implementation

Multi Hough Displaced Vertex Trigger (Tiancheng)

- Uses parallel Hough transformation with a hypothetical vertex
- Calculate track feature from the Hough map
- Use neural network to estimate true track origin
- Get the displacement from two-track vertex

- More detailed in Elias talk



Multi Hough Displaced Vertex Trigger

- Implemented:
  - 2D Hough transformation ✓
  - 2 peak finding ✓
  - Clustering ✓
  - Cluster parameter
  - Neural Network

	First Implementation	Optimised Version
LUT	1039%	78%
DSP	100%	79%

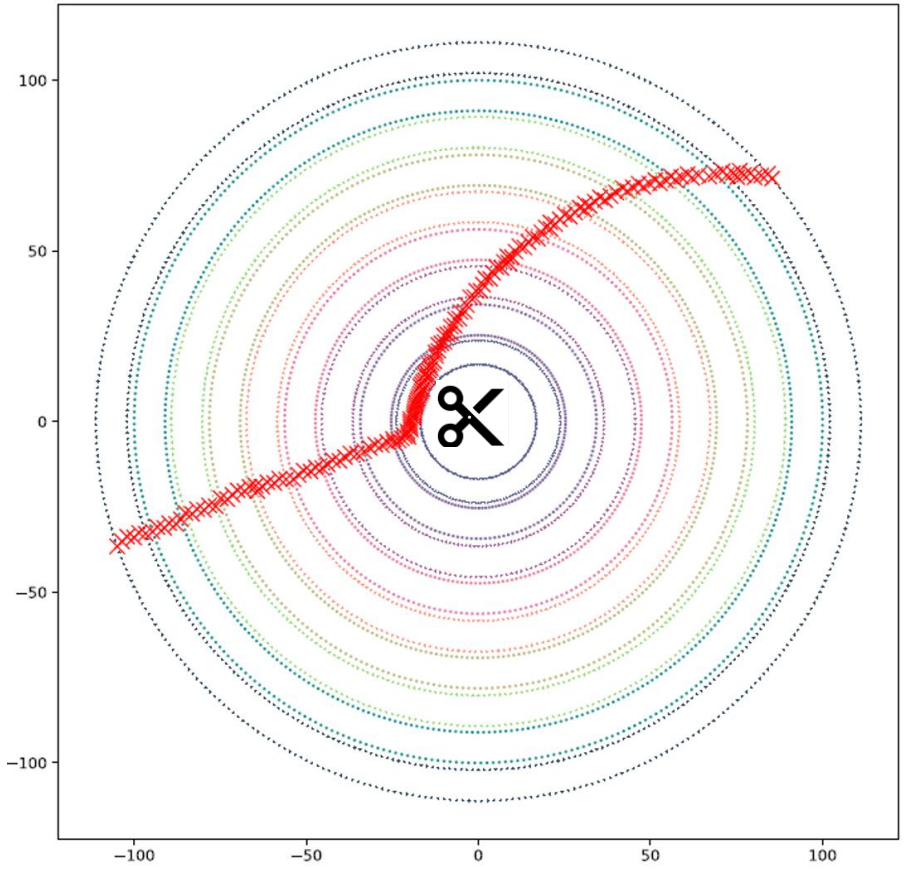
Todos

- Improve Hough transformation with Bram tables -> no DSPs
- Improve resource used
- Implementing cluster parameter
- Implementing NN

CNN Displaced Vertex Trigger (Yichuan)

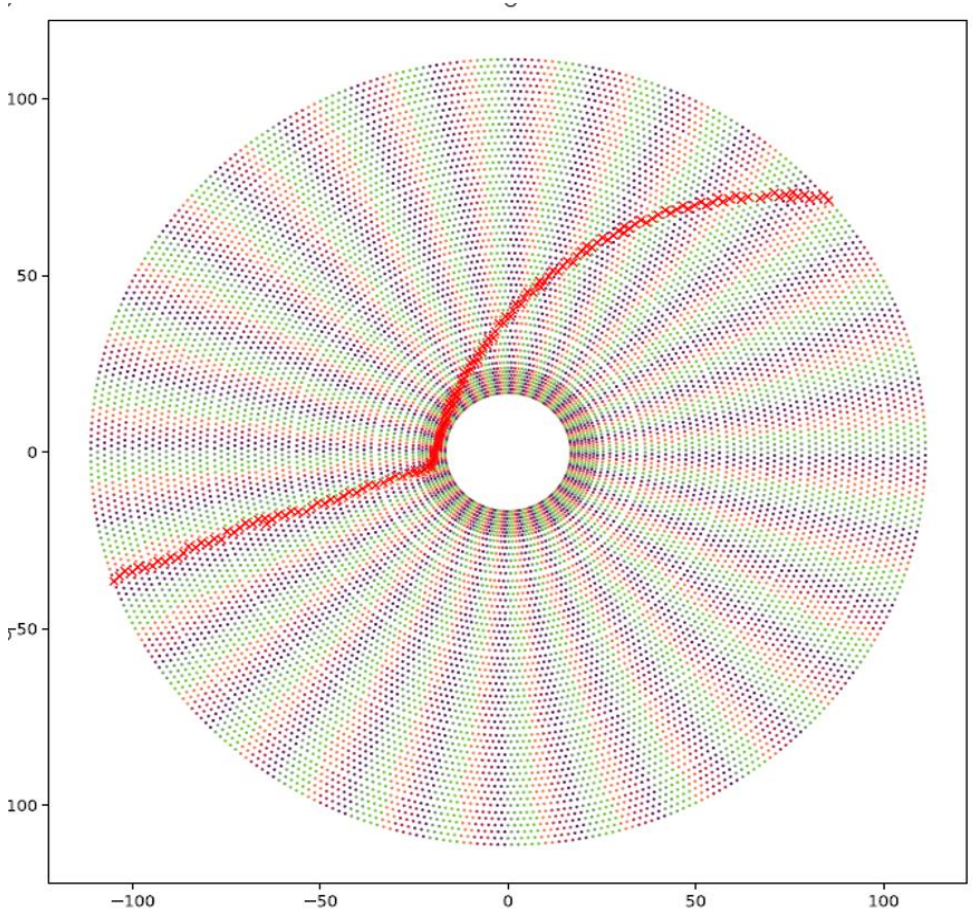
- Mastertesis Yichuan Ma finished
- Goal:
  - Proof that implementation is possible on UT4
  - Use state of the art tools
- Requirements:
  - Need to run on UT4
  - Need to handle experiment data rate

Methodology
Hit image cutting



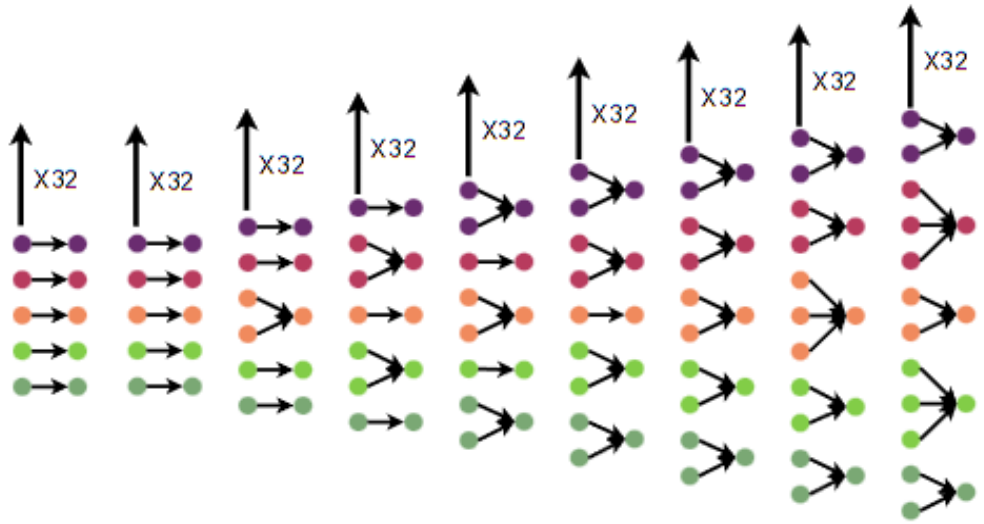
SL	0	1	2	3	4	5	6	7	8
Sense wires	160	160	192	224	256	288	320	352	384
	/32	/32	/32	/32	/32	/32	/32	/32	/32
	5	5	6	7	8	9	10	11	12

Methodology
Hit image aglining schematic



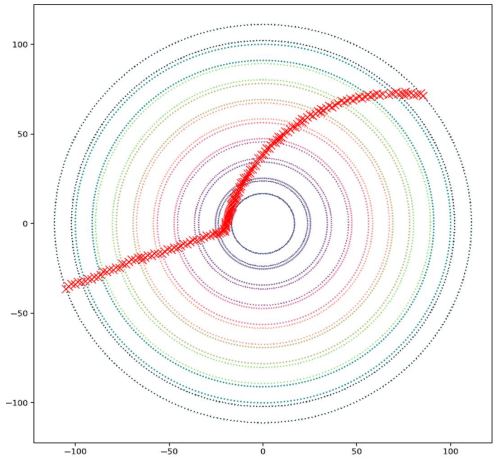
- **Aglining methode ploted on the CDC cross-section**

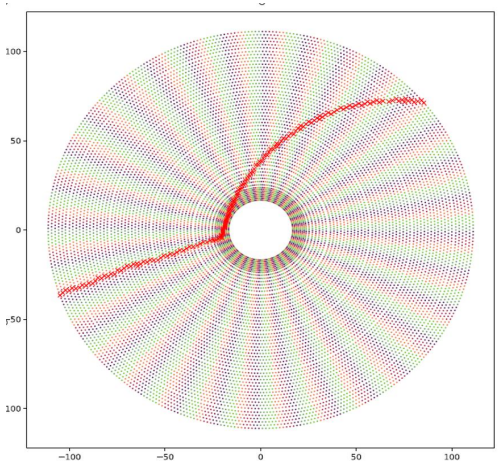
SL	0	1	2	3	4	5	6	7	8
Sense wires	160	160	192	224	256	288	320	352	384
/ 32	5	5	6	7	8	9	10	11	12



- **Proposed aglining methode with a agning width of 5 within each segment**

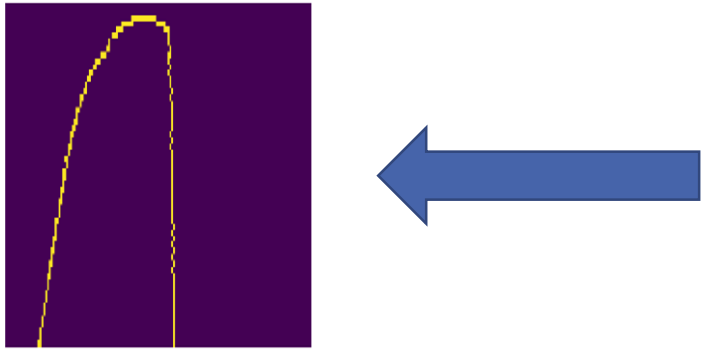
Methodology
Aglined Hit image





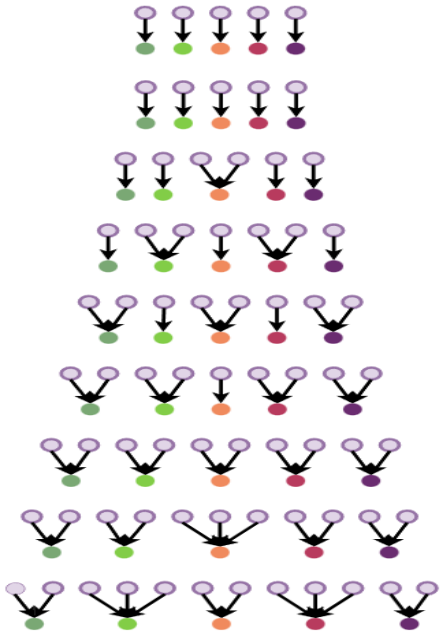
- Unsqueezed Hit images
  - 14336 pixels

- Squeezed Hit images
  - 8960 pixels, 37.5 % reduced

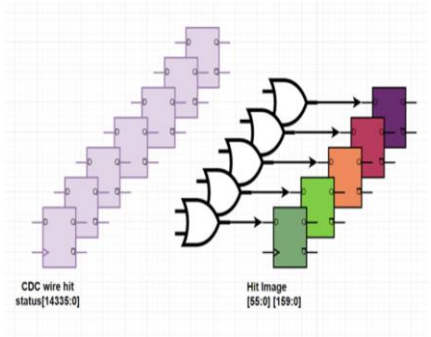


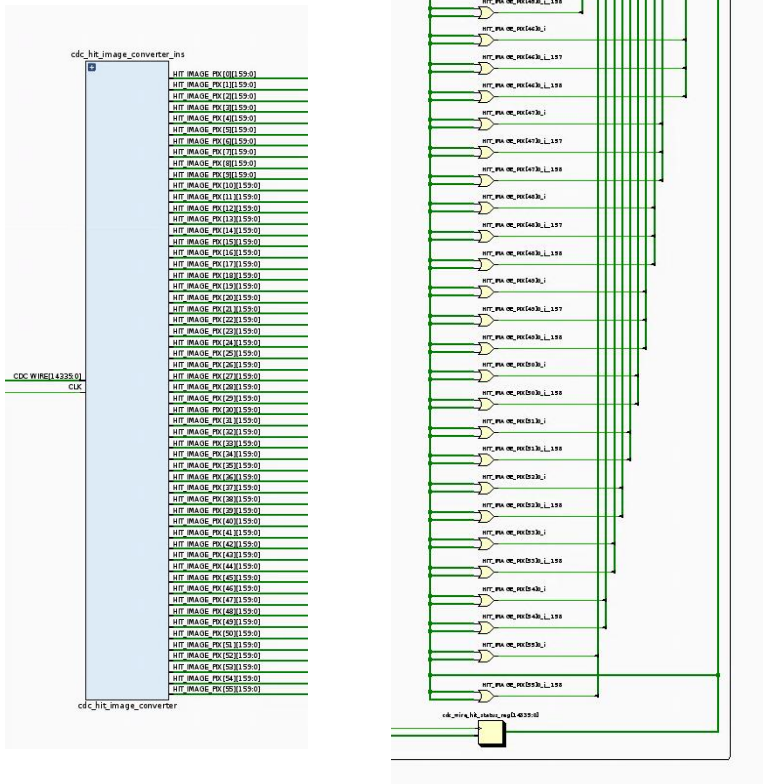
Methodology

Automatic hardware generation with Python



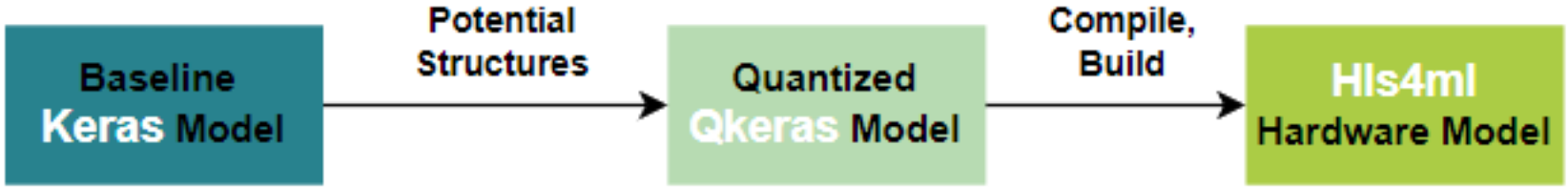
Code generator





- Hardware Design File automatic generated by a self-developed Python Script

Methodology
Keras + Qkeras + Hls4ml Workflow



Keras

- An open-source software libraray provides Python interface for artificial neural networks.

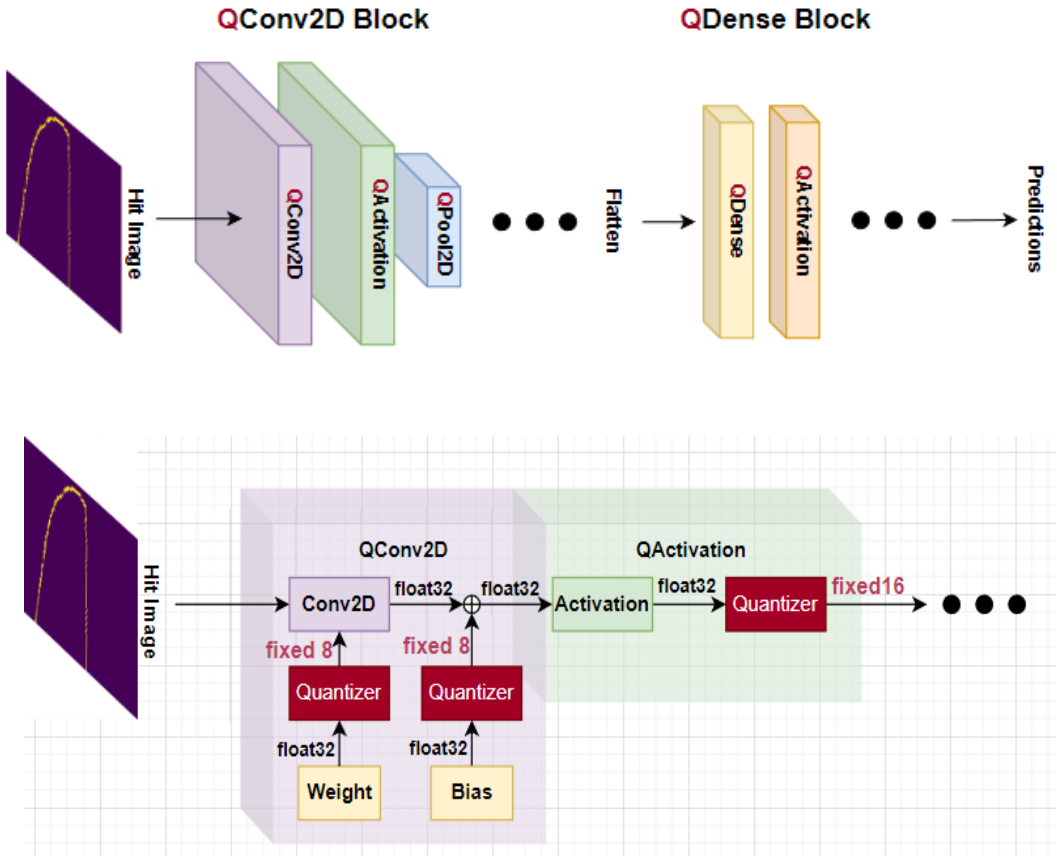
QKeras

- A quantization extension to Keras, supports parameter quantization and quantization aware Training

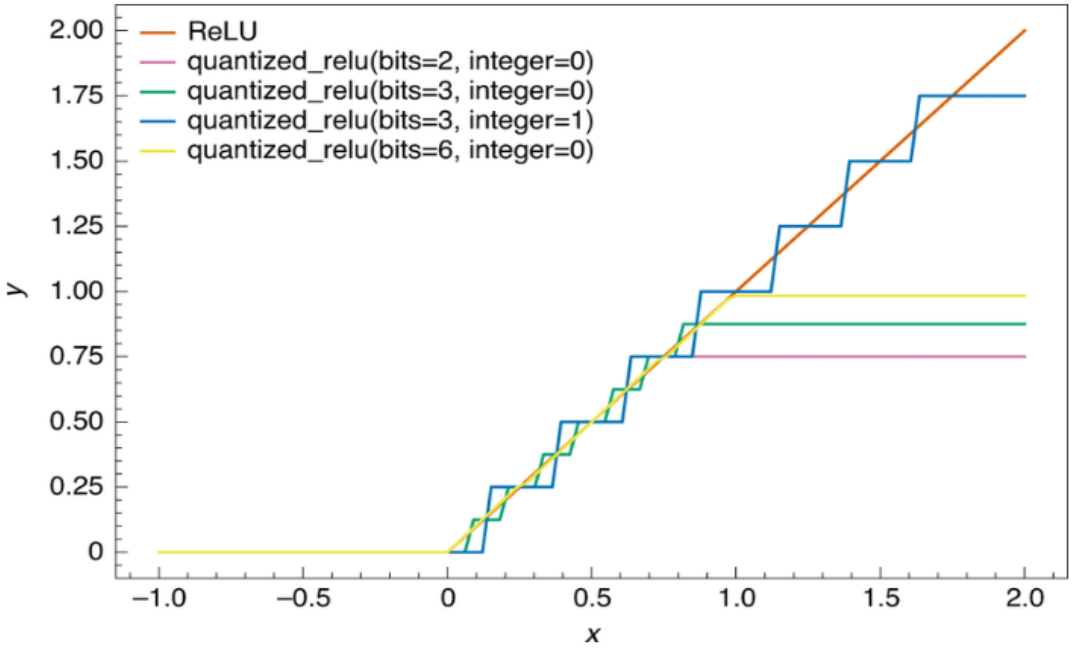
hls4ml

- A Python package for machine learning inference in FPGAs.
- Configurable FPGA firmware using HLS

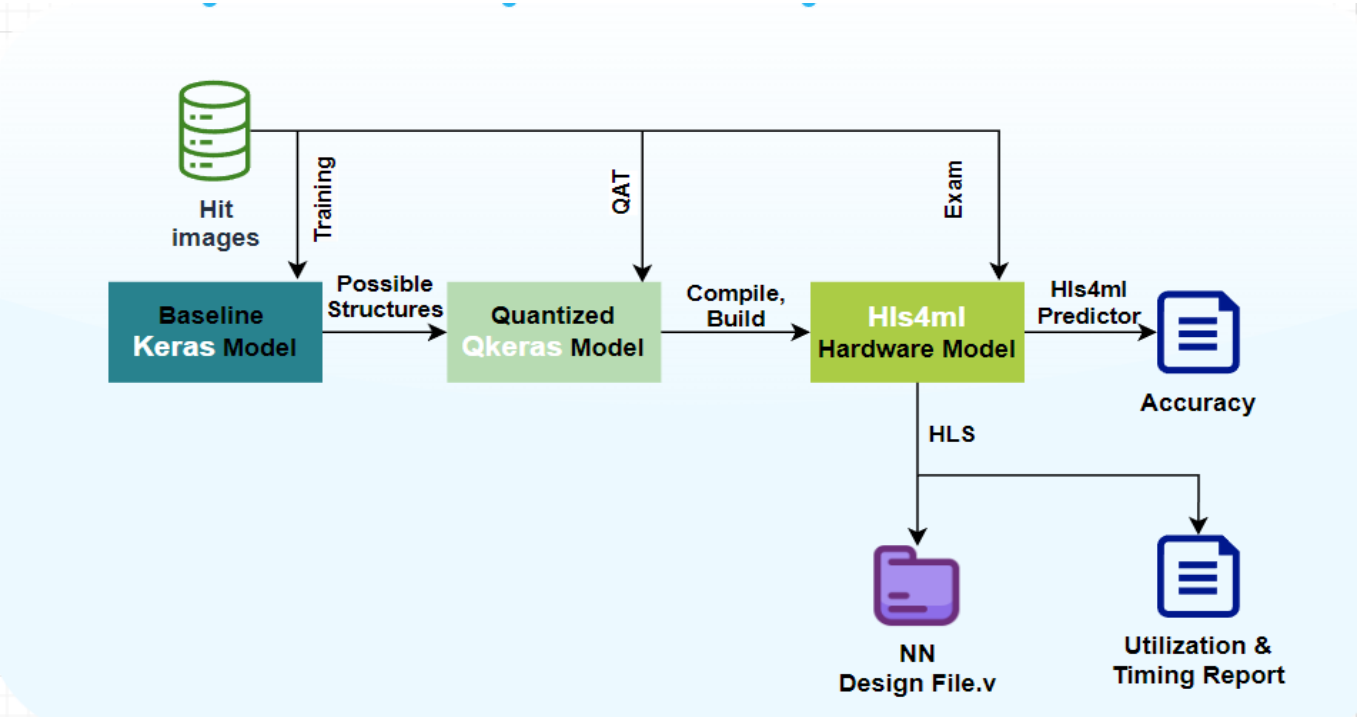
Methodology
Quantized model



- The same structure as the baseline model
- Each Layer is replced with its correspoding Qlayer in Qkeras.
- Qkeras provides various configurable Quantizer in the Qlayer



Methodology
Vertex CNN Model design flow



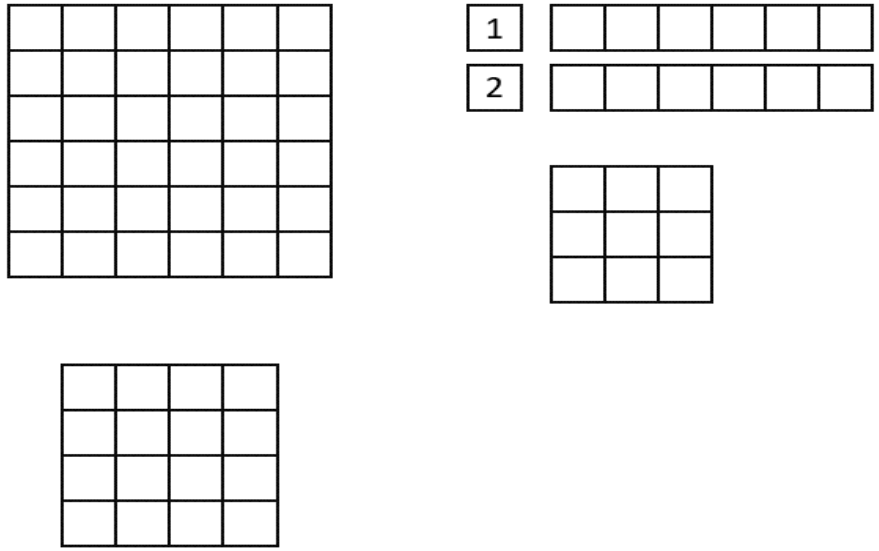
- Training
  - Loss function: Mean Squre Error(MSE)
  - Accuarcy Metric: Euclidean distance
  - Optimizer: Adam
- Timing Report
  - Total latency under 100 clock cycles?
  - Pipline stange lantecy under 31.75 Mhz?

Methodology
Hls4ml Conv2D Hardware implementation

- Total latency must under **100 clk cycles**.

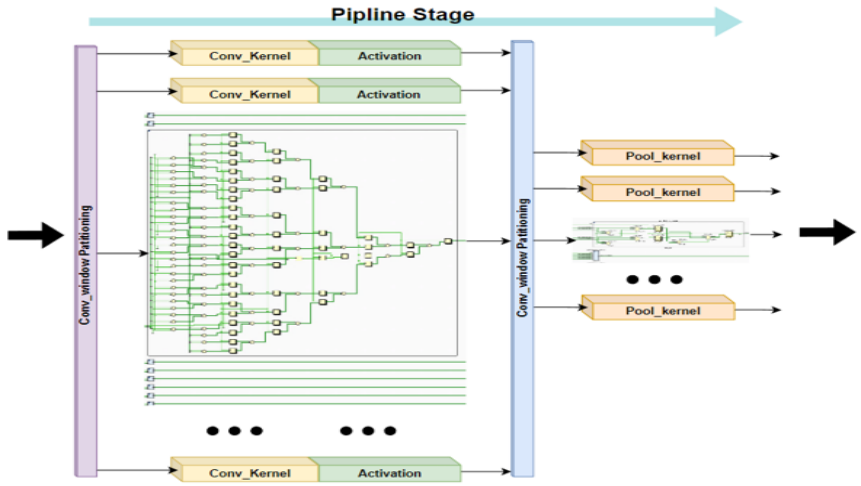
```
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  | 5.00 ns | 5.094 ns |   0.62 ns  |
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+----------+----------+------+------+----------+
    | Latency (cycles) |   Latency (absolute)  |   Interval  | Pipeline |
    |   min   |   max   |    min   |    max   |  min |  max |   Type   |
    +---------+---------+----------+----------+------+------+----------+
    |     8978|     8978| 45.734 us | 45.734 us |  8967|  8967| dataflow |
    +---------+---------+----------+----------+------+------+----------+
```

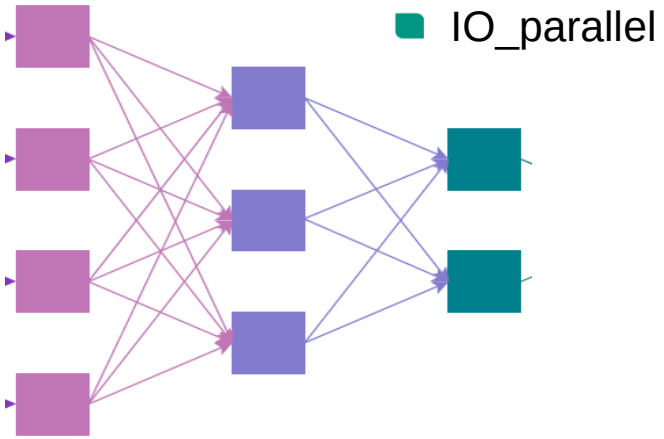
- The CNN implementation in Hls4ml is based on Stream, synthesed into **FIFO**



Methodology
New Strategie with Parallel ConvBlock

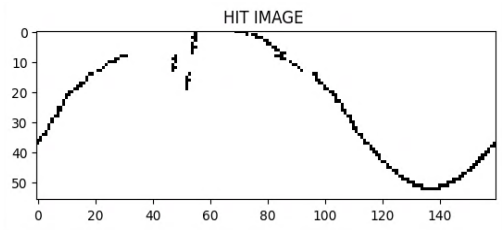
Parallel Conv2d Block

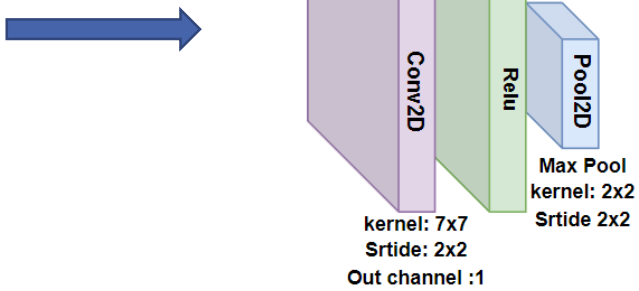


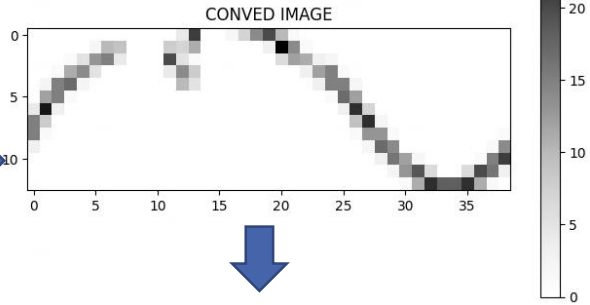


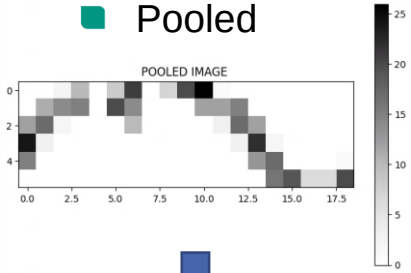
- Parallel ConvBlock is reponsiable for the Convolution portion of the Vertex CNN
  - The parameter is set to some constent value, accuarcy will drop.
- Hls4ml(IO_Parallel) is reponsiable for the MLP portion of the Vertex CNN
  - Requires a new dataset to train.

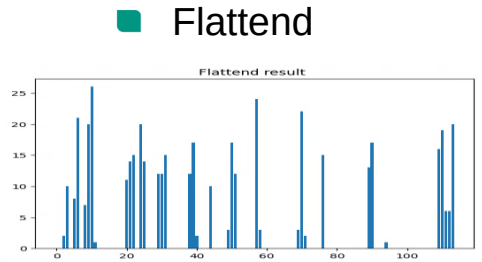
Implementation
Flattend Conved Hit images

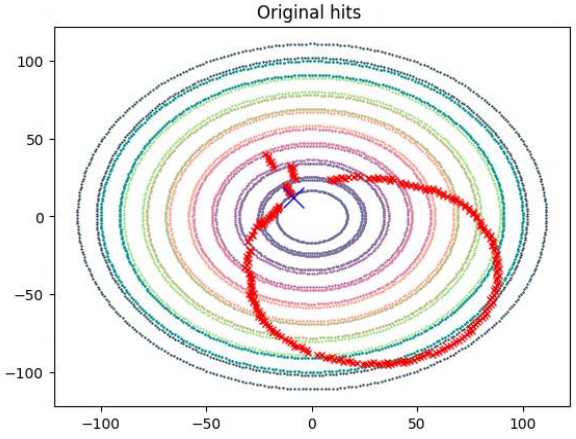


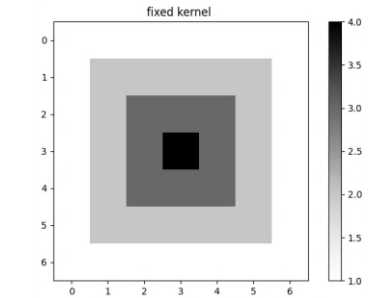




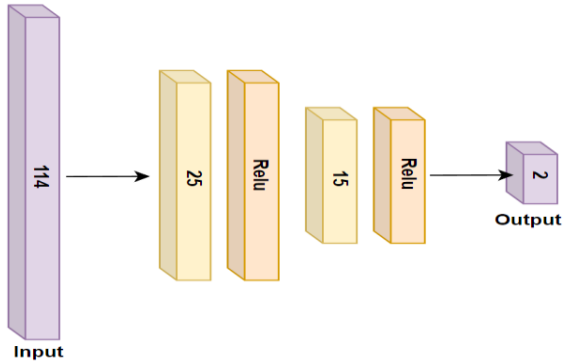








Implementation
Vertex MLP model



- Timing Report

```
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  | 5.00 ns | 4.270 ns |   0.62 ns  |
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+-----------+-----------+-----+-----+----------+
    | Latency (cycles)  |   Latency (absolute)  | Interval | Pipeline |
    |   min   |   max   |    min    |    max    | min | max |   Type   |
    +---------+---------+-----------+-----------+-----+-----+----------+
    |       14|       14| 70.000 ns | 70.000 ns |    1|    1| function |
    +---------+---------+-----------+-----------+-----+-----+----------+
```

- Compare to the hls4ml CNN model

	Presicion	Accuarcy	Max Stage Latency	Total latency
Qkeras MLP Model	**fixed<16,8>**	**10.83cm**		
Hls4ml MLP Model	**fixed<16,10>**	**10.85cm**	**1**	**14**
Qkeras CNN Model	**fixed<16,8>**	**7.25cm**		
Hls4ml CNN Model	**fixed<16,8>**	**7.25cm**	**8967**	**8978**

- 50% accuarcy drop, but much faster !!

- Keyword

Result
Full System with Gigabyte transceiver

- Timing and hardware utilization on Xilinx UT4

	Hit image converter	Parallel ConvBlock	Vertex MLP	Full system
Stage Latency	1	1	1	1
Total latency	1	9	14	24
DSP	0%	0%	75%	76%
BRAM	1%	0%	0%	8%
LUTRAM	0%	1%	0%	1%
LUT	1%	5%	19%	22%