Dec 9, 2018 • CPAD Instrumentation Frontier Workshop 2018

DNN-based algorithm for the CMS Level-1 muon reconstruction

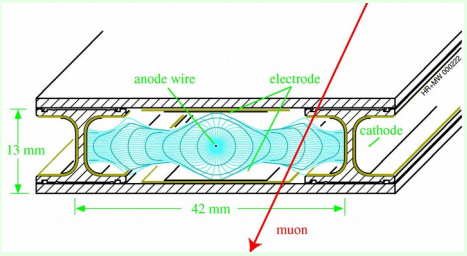
Jia Fu Low (University of Florida)
on behalf of the CMS collaboration

Overview

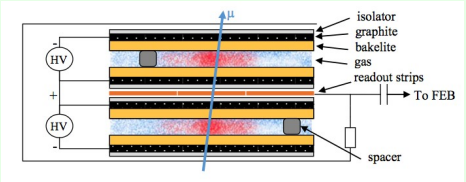
- Introduction
  - CMS muon system & L1 trigger system
  - Endcap Muon Track Finder (EMTF)
- Phase 2 EMTF++ algorithm (under R&D)
- DNN implementation in the FPGAs
- Summary

CMS Muon System

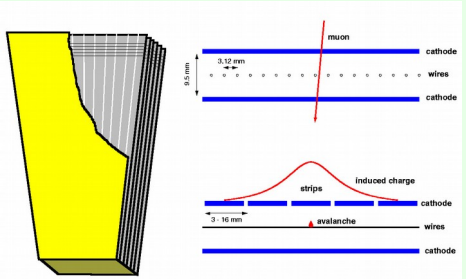
Drift Tubes

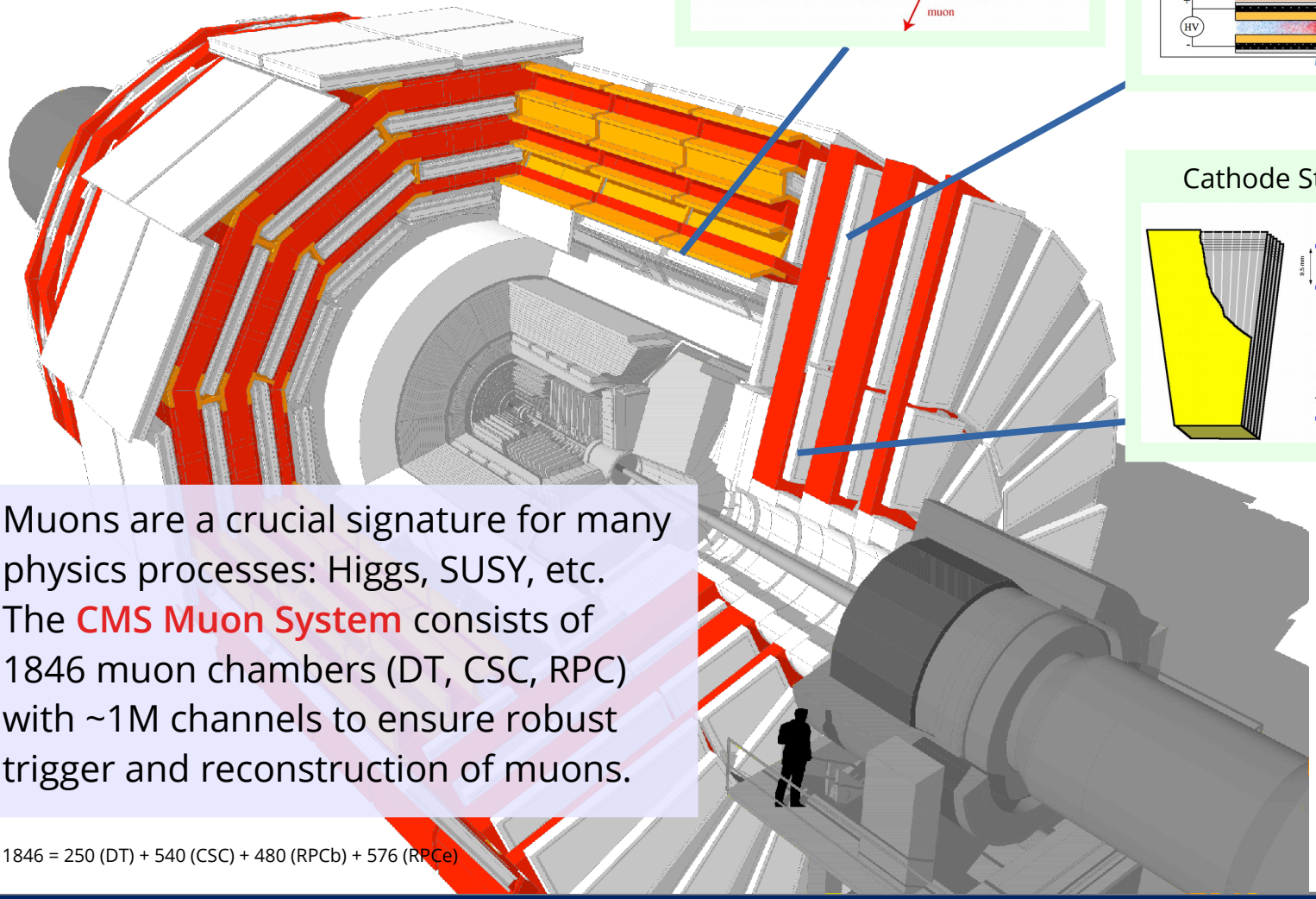


Resistive Plate Chambers



Cathode Strip Chambers

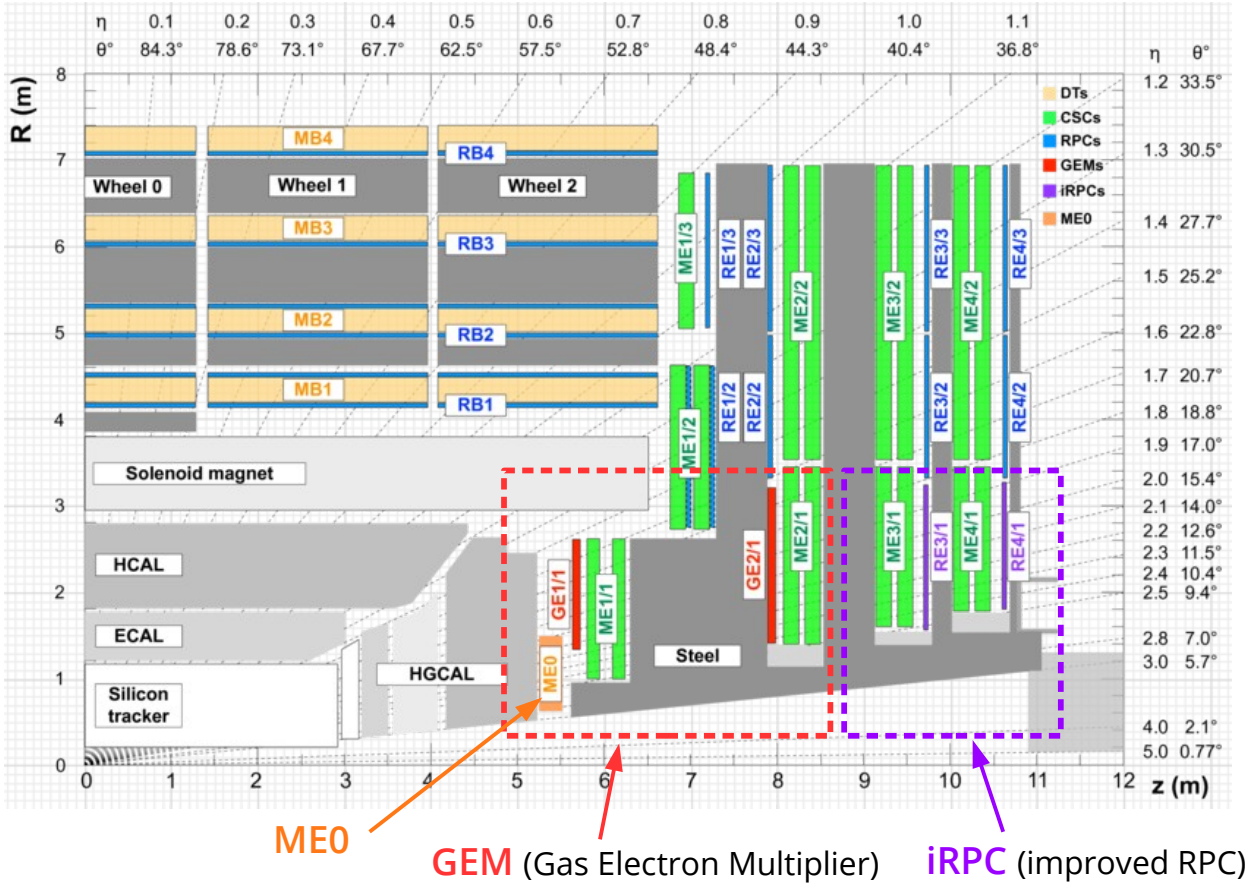




Muons are a crucial signature for many physics processes: Higgs, SUSY, etc. The **CMS Muon System** consists of 1846 muon chambers (DT, CSC, RPC) with ~1M channels to ensure robust trigger and reconstruction of muons.

1846 = 250 (DT) + 540 (CSC) + 480 (RPCb) + 576 (RPCe)

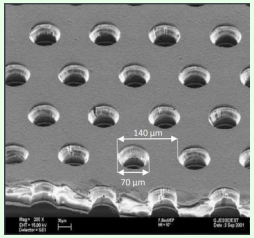
Phase-2 Muon Upgrade

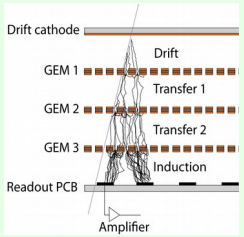


HL-LHC (a.k.a. Phase-2, starting in 2026) inst lumi could reach 7.5×10^{34} $cm^{-2}s^{-1}$ with <PU> ~200 (today ~40).

To maintain excellent trigger performance in the harsher conditions, the forward muon system will be enhanced with new muon detectors (GEM, iRPC, ME0).

Some electronics of the existing muon detectors will also be replaced.

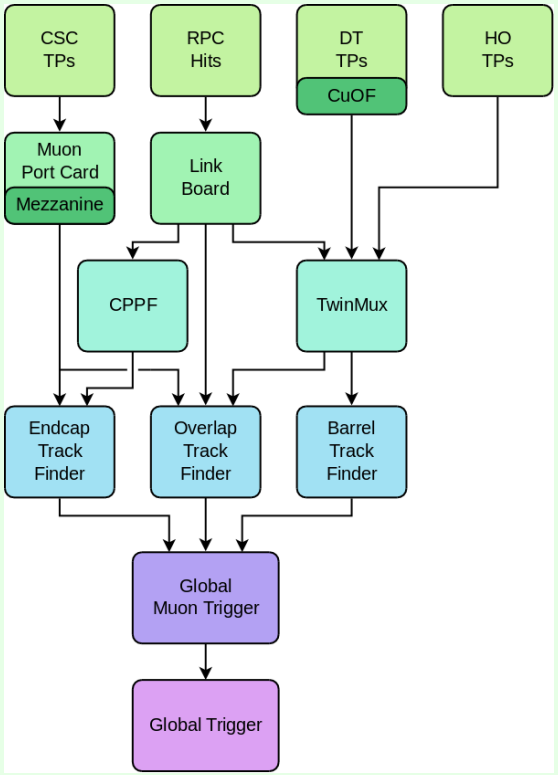




L1 Trigger System

- CMS two-level trigger system:
  - **Level-1 (L1)**: custom electronics (e.g. FPGAs) used to reduce event rate of 40 MHz to 100 kHz within 4 μs latency.
  - **High Level Trigger (HLT)**: large CPU farm used to run software algorithms to further reduce event rate to 1 kHz.
- **L1 muon trigger system**:
  - Muon detectors have dedicated electronics that create the Trigger Primitives (TPs).
  - The TPs are collected and sent to the regional **Track Finders** (barrel, overlap, endcap) which find muons and measure their transverse momenta p_T.
  - The muons are sent to the Global Muon Trigger, and then to the Global Trigger, which makes the trigger decision (typically $p_T > 22$ GeV in Run 2).
- L1 muon trigger algorithm needs to be executed **very quickly and efficiently.**
- For Phase-2, the upgraded L1 trigger will have a larger bandwidth (~750 kHz) and a longer latency (12.5 μs).
  - But the architecture will look completely different with the addition of the track trigger at L1. A lot of discussion are still ongoing…

L1 muon trigger system (2017-18)

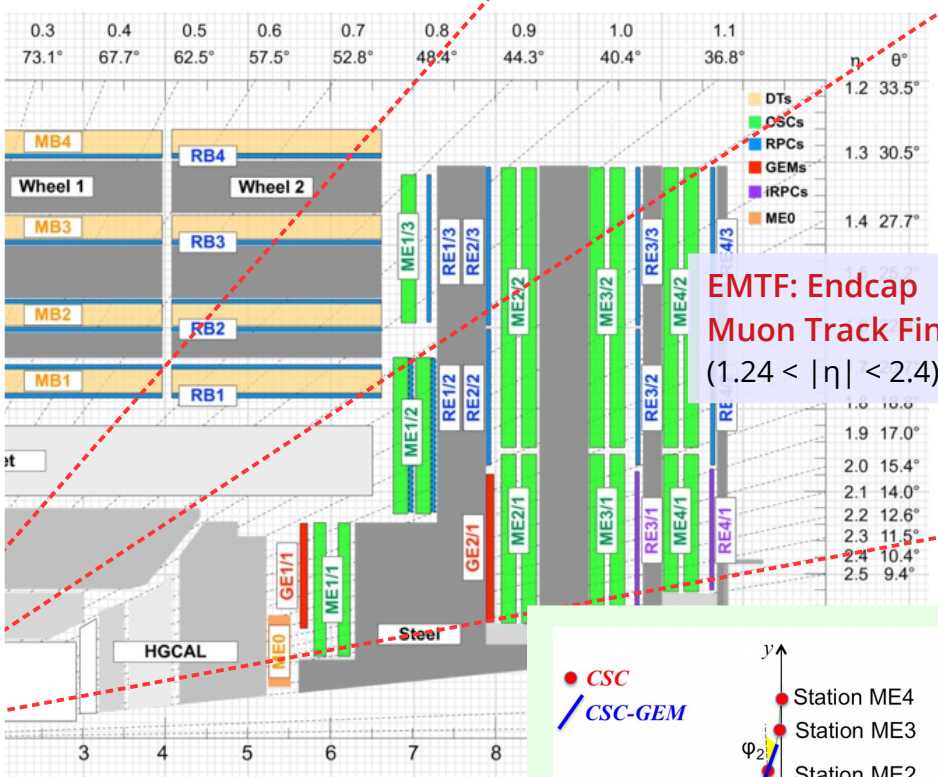


EMTF

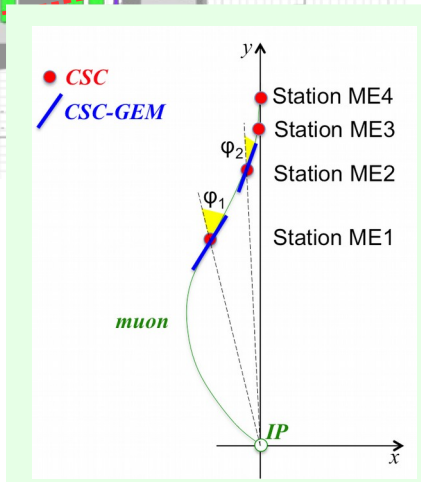
- Challenges:
  - Highly **non-uniform magnetic field** with very little magnetic bending in the very forward region.
  - **Large background** from low p_T muons, punch-thrus, neutrons, etc that could lead to non-linear PU dependence.
- For Phase-2, EMTF has to evolve to
  - Incorporate the **new muon detectors**.
  - Improve **efficiency, redundancy, p_T resolution, timing**.
  - Maintain the same trigger threshold at a reasonable **rate** at <PU>=200 to remain sensitive to electroweak scale physics.

BMTF: Barrel Muon Track Finder
(0 < |η| < 0.8)

OMTF: Overlap Muon Track Finder
(0.8 < |η| < 1.24)



EMTF: Endcap Muon Track Finder
(1.24 < |η| < 2.4)



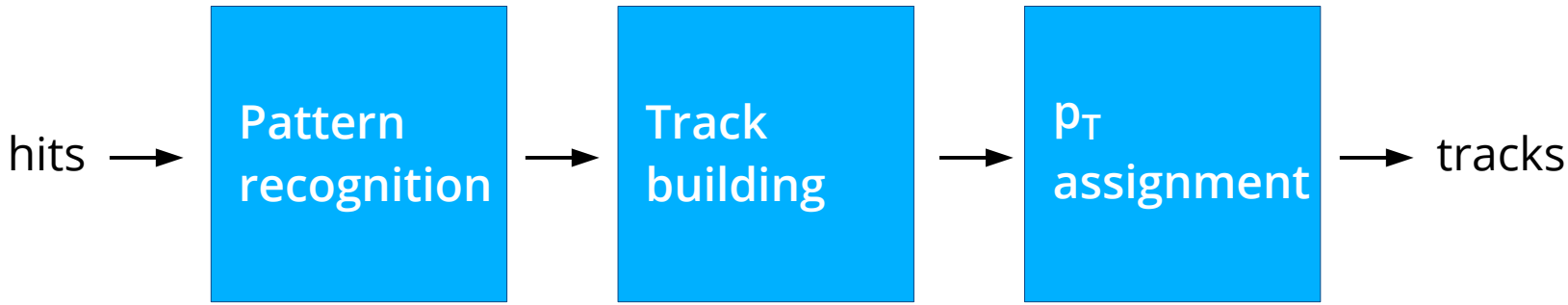
GEM-CSC "bending angles" help improve p_T resolution and reduce trigger rate.

Trigger primitives

	CSC				RPC	iRPC	GEM		ME0
	ME1/1a	ME1/1b	10° chmb (non-ME1/1)	20° chmb (non-ME1/1)			GE1/1	GE2/1	
η	2.1-2.4	1.6-2.1	0.9-1.8	1.6-2.4	0-1.9	1.8-2.4	1.6-2.2	1.6-2.4	2.0-2.8
chamber size	10°	10°	10°	20°	10°	20°	10°	20°	20°
# of strips or segmentations	96	128	160	160	32	96	192	384	384
ϕ resolution (°)	0.1042	0.0781	0.0625	0.1250	0.3125	0.2083	0.0521	0.0521	0.0213
relative ϕ resolution	1.6667	1.2500	1.0000	2.0000	5.0000	3.3333	0.8333	0.8333	0.3402
# of wires or partitions	varies	varies	varies	varies	3	5	8	8	8
θ resolution (°)	?	?	?	?	?	?	?	?	?

- Different muon detectors have different spatial resolution (ϕ, θ). Want to combine them in the most optimal way to improve muon p_T measurement.

EMTF++ algorithm basics



- EMTF++ is an extension of the current EMTF algorithm, and it follows the same design.
- **Pattern recognition**
  - Use patterns to find hits in the 4 (or more) muon stations that are consistent with muons of different p_T.
- **Track building**
  - Build tracks by picking a unique hit from each muon station.
- **p_T assignment**
  - Use machine learning algorithm (e.g. BDT, NN) to determine the muon p_T using all the discriminating variables: $\Delta\phi$'s, $\Delta\theta$'s, bend, η, etc.
  - Current EMTF implements the BDT algorithm with a large LUT. Multiple input variables are encoded into 30-bit address space, which is used to store the p_T in the LUT. For Phase-2, the LUT address space will be increased to 37-bit.
  - For Phase-2, also want to investigate running NN directly in FPGAs.

Pattern recognition

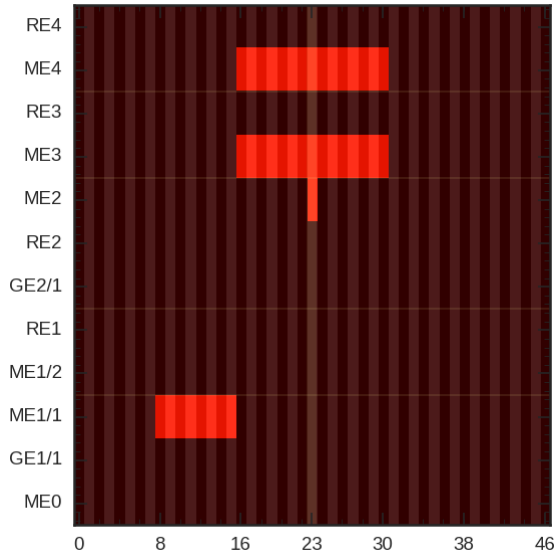
- The patterns have been reworked to accommodate the new detectors
  - Current EMTF patterns use 4 windows for the 4 muon stations.
  - Expanded to have windows for the following 12 muon stations:

ME1/1	ME1/2	ME2	ME3	ME4	RE1	RE2	RE3	RE4	GE1/1	GE2/1	ME0

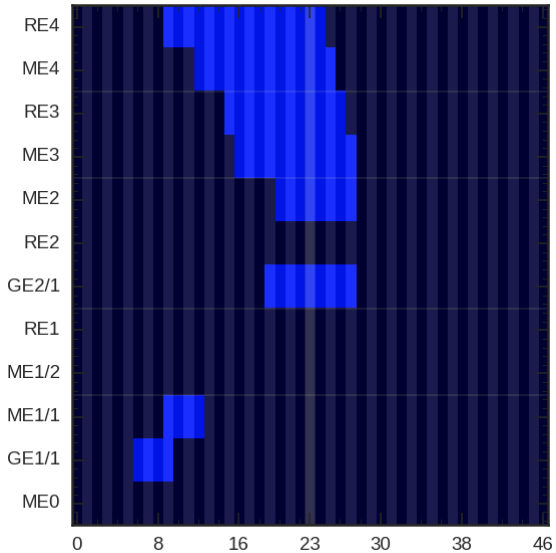
(ME1/1 and ME1/2 are split because they are actually very different due to different magnetic bending)

  - Patterns have different straightness (from 0 to 4) for muons of different p_T (low to high). The windows are tuned for each station using a muon gun.
  - Also increased the number of zones (η partitions) from 4 to 6.

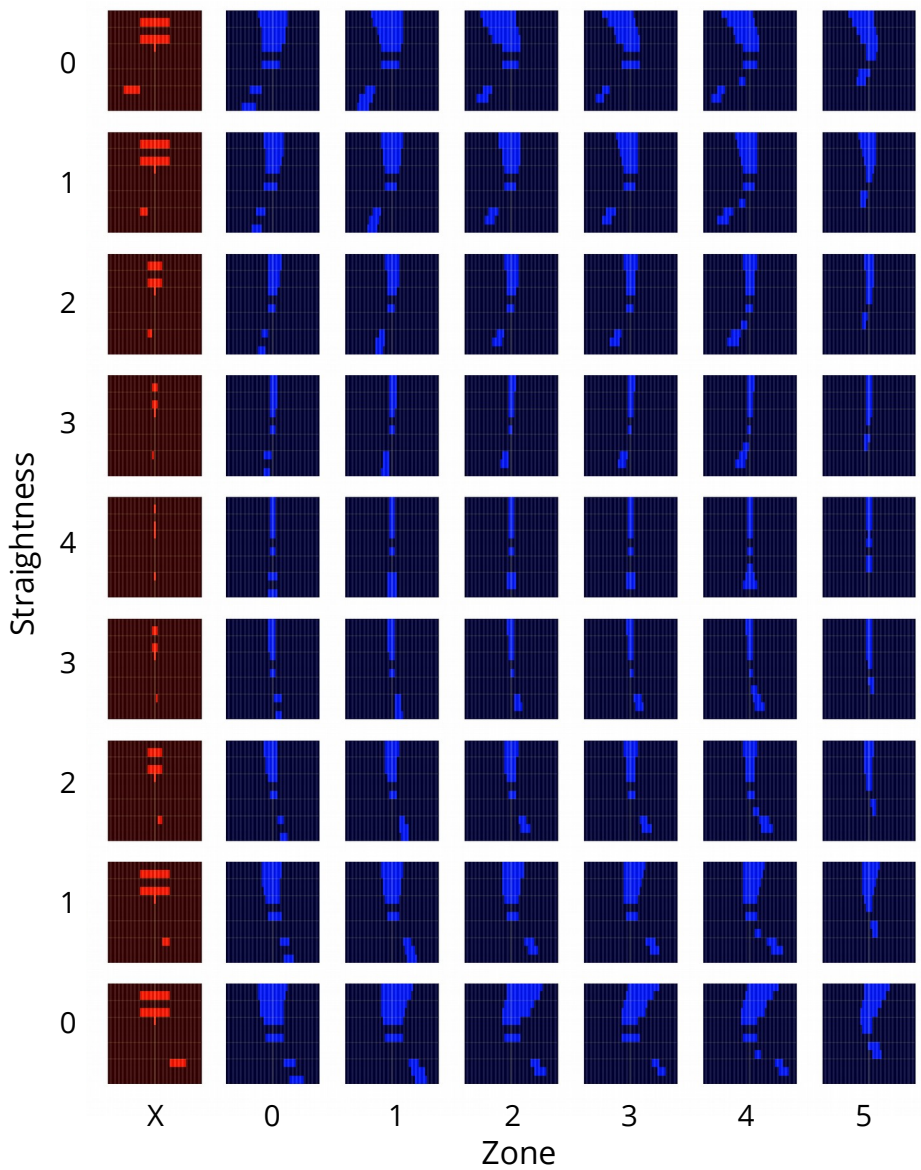
EMTF pattern
(straightness=0, charge=-1)



EMTF++ pattern (zone 1)
(straightness=0, charge=-1)



Pattern recognition

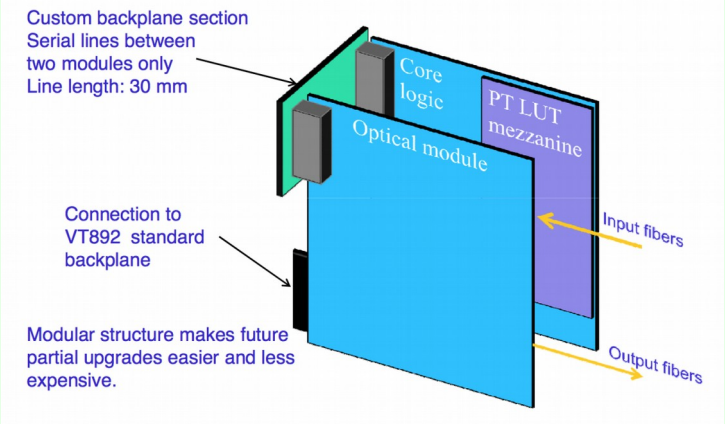


If there are multiple hits in a station, the hit with the most compatible ϕ & θ is selected by comparing to the track ϕ & θ.

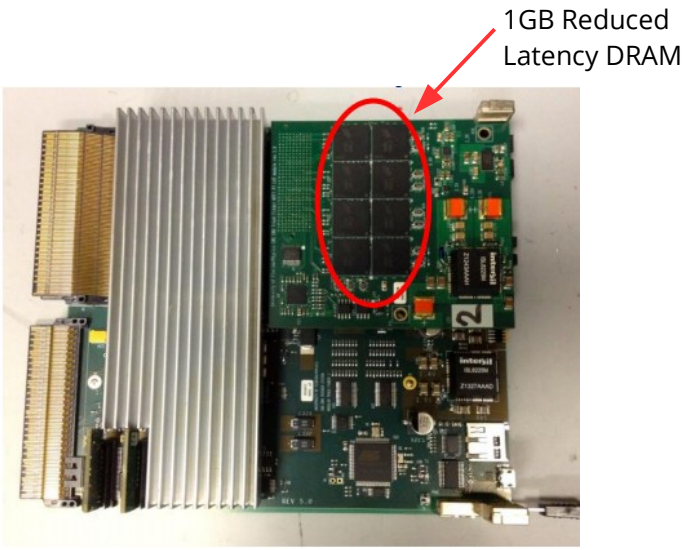
p_T assignment

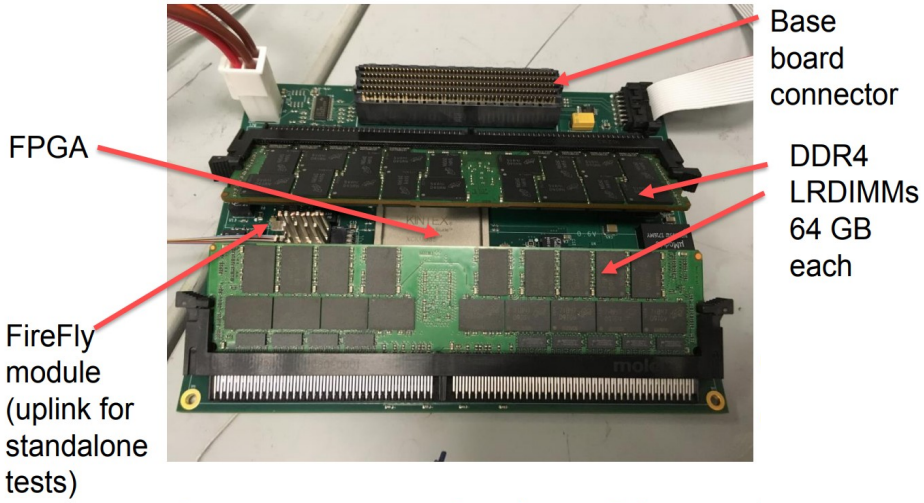
- Current EMTF p_T assignment based on **Boosted Decision Tree (BDT)** regression.
  - About 25 discriminating variables: Δϕ's, Δθ's, bend, track θ, etc.
  - Offline, train the BDT and evaluate it for every possible combination of the input variables. Encode/compress the input variables into the address space of the LUT, and "memorize" the BDT output.
  - During running, FPGA does some preprocessing to compress the input variables and extract the p_T from the LUT.
- MTF7 board has a **very large, fast LUT** for p_T assignment
  - 1GB PTLUT mezzanine: 30-bit address space, ~50ns random address lookup
  - Core FPGA: Xilinx Virtex-7
- Phase-2 APd board will have a larger PTLUT: 128 GB with 37-bit address space, based on DDR4 RAM.

MTF7 conceptual diagram



The hardware





- **Phase-1 MTF7 board**
  - 1GB PTLUT mezzanine: 30-bit address space, ~50ns random address lookup
  - Core FPGA: Xilinx Virtex-7
  - 84 input links (up to 10 Gb/s)
  - 24 output links (up to 10 Gb/s)
  - Installed in 2016

- **Phase-2 APd board**
  - 128GB PTLUT mezzanine: 37-bit address space, ~50ns random address lookup
  - Core FPGA: Xilinx Virtex Ultrascale+ VU9P
  - ~100 input links (up to 28 Gb/s)
  - ~100 output links (up to 28 Gb/s)
  - Currently under R&D

p_T assignment

- For Phase 2, we want to investigate p_T assignment using **neural network (NN)**
  - Alleviates bottleneck of # of address bits to LUT by using logic and DSPs in FPGA.
  - Can use many variables without heavy compression.
  - Possible to implement NN inference in FPGAs with low latency
    - See HLS4ML paper: arXiv:1804.06913

See Dylan Rankin's talk

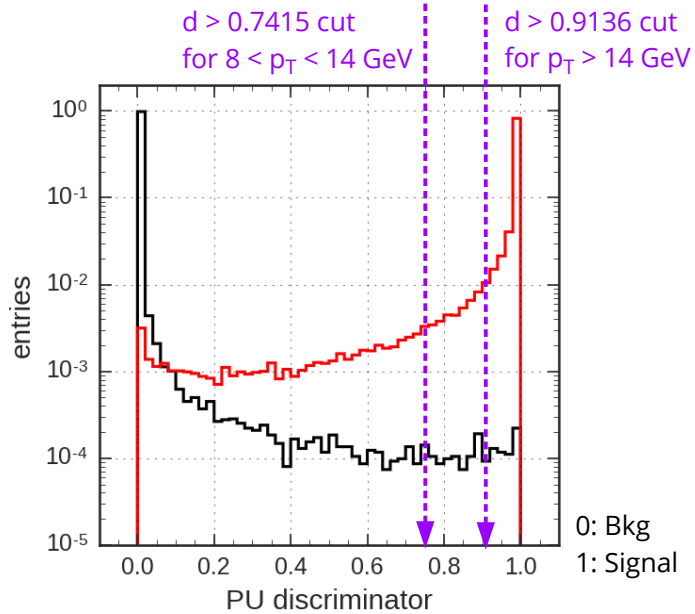
- At the moment, consider **39 features**
  - Can easily add more, provided enough FPGA resources (more on this later).

	ME1/1	ME1/2	ME2	ME3	ME4	RE1	RE2	RE3	RE4	GE1/1	GE2/1	ME0
ϕ	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
θ	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
bend	✔	✔	✔	✔	✔							✔
F/R	✔	✔										✔
ring			✔	✔	✔							

+ pattern straightness
+ zone
+ median theta

Neural network

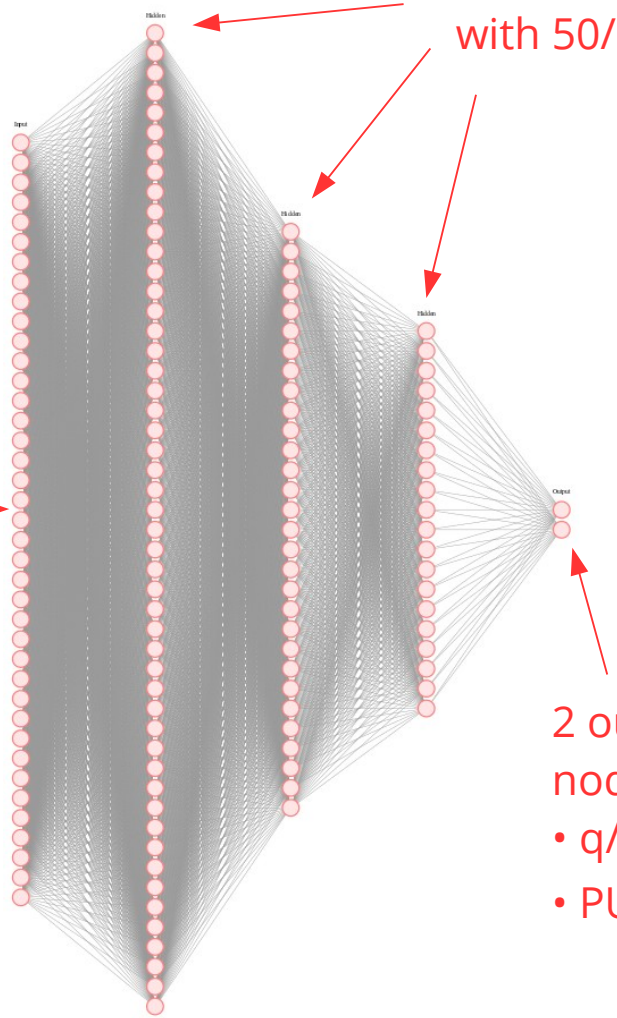
- Using **Keras**, create a feedforward NN with **3 hidden layers** (50, 30, 20 nodes respectively). Batch normalization is applied at all the hidden layers.
- Two output nodes: **q/p_T** and **PU discriminator**, the latter is used to reject PU tracks and bad quality tracks.
  - q/p_T is a regression trained with muons to assign momentum
  - PU discriminator is a classification trained with:
    - S = muon $p_T > 8$ GeV
    - B = any track from 200 PU. Veto applied to any events with $p_T > 8$ GeV.
- PU discriminator is used as a cut:
  - If q/p_T output gives $p_T > 8$ GeV, a cut is applied to PU discriminator to accept or reject the muon. A tighter cut is applied to for $p_T > 14$ GeV muons.
- From simulation studies, we have achieved roughly 4x rate reduction with the addition of the new muon detectors. The overall efficiency has also been improved.
  - Still work in progress.



Neural network

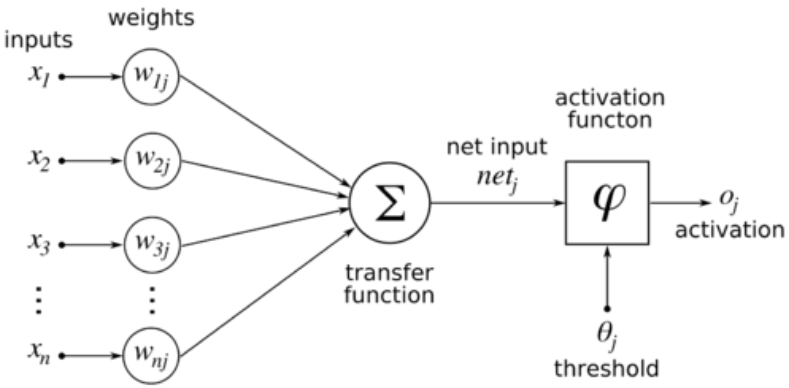
3 hidden layers
with 50/30/20 nodes

39 input
nodes



2 output
nodes:
- q/p_T
- PU discr

Each node computes $o_j = \varphi\left(\sum_{i=1}^{n} w_{ij} \cdot x_i + \theta_j\right)$

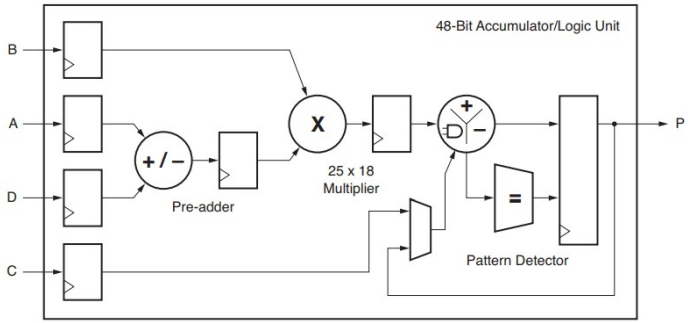


Pruning can be applied to remove unimportant synapses to reduce resource usage.

NN firmware implementation

- hls4ml toolkit
  - A toolkit to implement fast neural network inferences in FPGAs using Vivado High-Level Synthesis (HLS).
  - Can convert NN models from popular ML libraries (Keras, Tensorflow, etc) into VHDL codes, which can be used to generate the firmware.
    - Optimize usage of DSPs in the modern FPGAs for multiply-accumulate operations
  - See paper: arxiv:1804.06913

Basic DSP48E1 Slice functionality



Device	# of DSPs
Kintex-7 325T	840
Virtex-7 690T	3600
Kintex UltraScale KU115	5500
Virtex UltraScale+ VU9P	6800

NN firmware resource usage

- Converted an earlier version of NN using HLS4ML
  - 80 input nodes, 3 hidden layers with 64/32/16 nodes, and 1 output node.
  - Pruning is applied to remove 50% of the unimportant synapses
  - Use <18,8>-precision for inputs & output. 8 integer bits, 18 total bits
- Xilinx Virtex-7 target FPGA (as in the MTF7 board)
  - Clock of 250 MHz
- Currently re-optimizing the NN structure for less resource usage.
  - A smaller version with 30/25/20 nodes uses only 50% DSP resource.

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	6
FIFO	-	-	-	-
Instance	56	2822	315515	112745
Memory	-	-	-	-
Multiplexer	-	-	-	36
Register	-	-	4689	-
Total	56	2822	320204	112787
Available	2940	3600	866400	433200
Utilization (%)	1	78	36	26

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00	3.49	0.50

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
40	40	1	1	function

~160ns fixed latency

NN firmware resource usage

- HLS estimates and actual FW implementation resources agree reasonably well
- DSP usage is spot on. LUT and FF are over-estimated in HLS (seen before). Conclusions agree well with earlier observations using generic NNs.

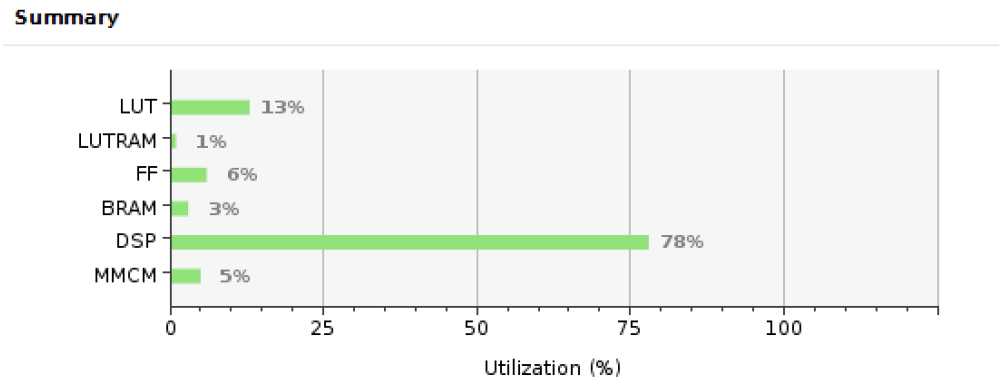
HLS Estimates

Utilization Estimates

Summary

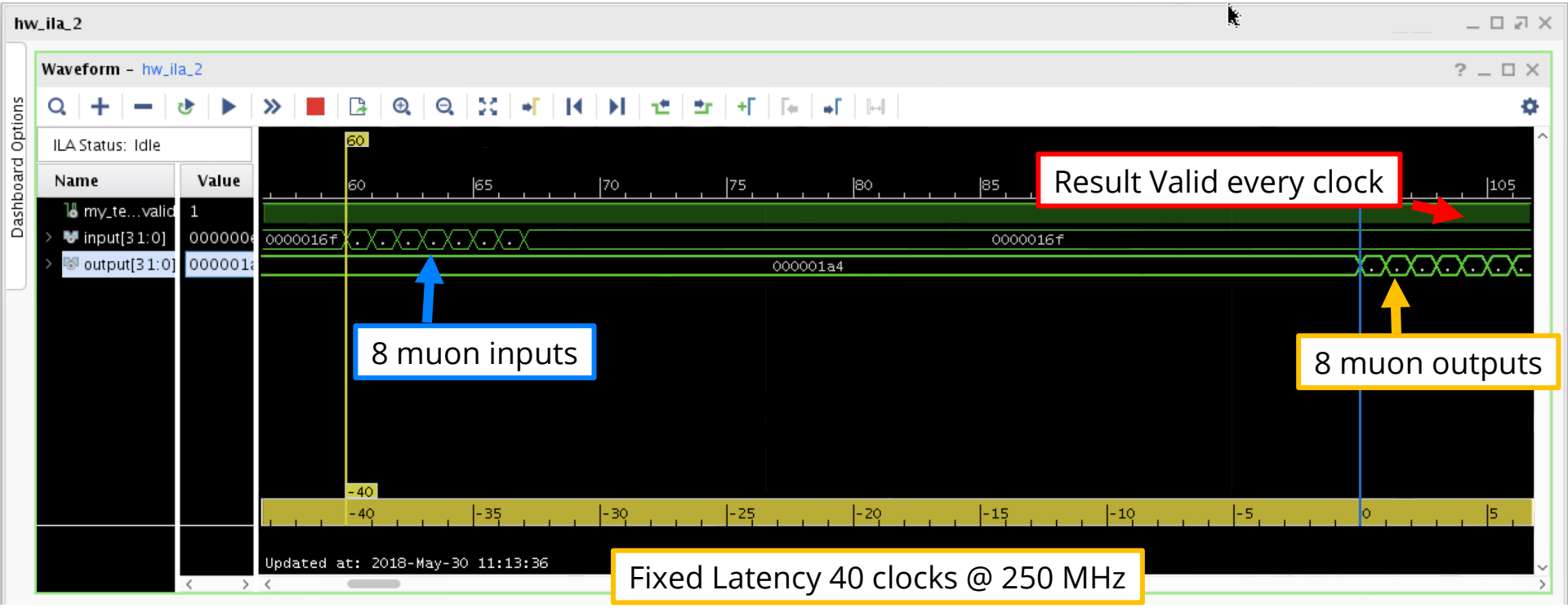
Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	6
FIFO	-	-	-	-
Instance	56	2822	315515	112745
Memory	-	-	-	-
Multiplexer	-	-	-	36
Register	-	-	4689	-
Total	56	2822	320204	112787
Available	2940	3600	866400	433200
Utilization (%)	1	78	36	26

Design synthesis in Vivado 2018.1



NN in MTF7 hardware

- Reading inputs corresponding to 8 simulated muons and calculating their p_T.
- HLS IP core target frequency 250 MHz (which agrees with design frequency)



Expect latency of ~40 + N_{muons} clocks

99% percentile of tracks to fit in both endcaps is 5. Per processor 5/12 = 0.4

Extrapolation to Virtex Ultrascale+

- A look at the future devices:
  - Xilinx Virtex Ultrascale+ VU9P FPGA (device ID: xcvu9p-flga2104-2-i)
  - Targeted by the USCMS APd board
  - Clock of 333 MHz

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+-------+---------+---------+-----+
|       Name      | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+-----------------+---------+-------+---------+---------+-----+
|DSP              |        -|      -|        -|        -|    -|
|Expression       |        -|      -|        0|        6|    -|
|FIFO             |        -|      -|        -|        -|    -|
|Instance         |       56|   2822|    70358|   121761|    -|
|Memory           |        -|      -|        -|        -|    -|
|Multiplexer      |        -|      -|        -|       36|    -|
|Register         |        -|      -|     5249|        -|    -|
+-----------------+---------+-------+---------+---------+-----+
|Total            |       56|   2822|    75607|   121803|    0|
+-----------------+---------+-------+---------+---------+-----+
|Available        |     4320|   6840|  2364480|  1182240|  960|
+-----------------+---------+-------+---------+---------+-----+
|Utilization (%)  |        1|     41|        3|       10|    0|
+-----------------+---------+-------+---------+---------+-----+
```

```
+ Timing (ns):
    * Summary:
    +--------+-------+----------+------------+
    |  Clock | Target| Estimated| Uncertainty|
    +--------+-------+----------+------------+
    |ap_clk  |   3.00|      2.49|        0.38|
    +--------+-------+----------+------------+

+ Latency (clock cycles):
    * Summary:
    +-----+-----+-----+-----+----------+
    |  Latency  |  Interval | Pipeline |
    | min | max | min | max |   Type   |
    +-----+-----+-----+-----+----------+
    |   24|   24|    1|    1| function |
    +-----+-----+-----+-----+----------+
```

Fixed Latency is 24 cycles @333MHz this is 72ns

The algorithm should fit comfortably in FPGAs considered for Phase-2 boards

NN configuration for Run 3

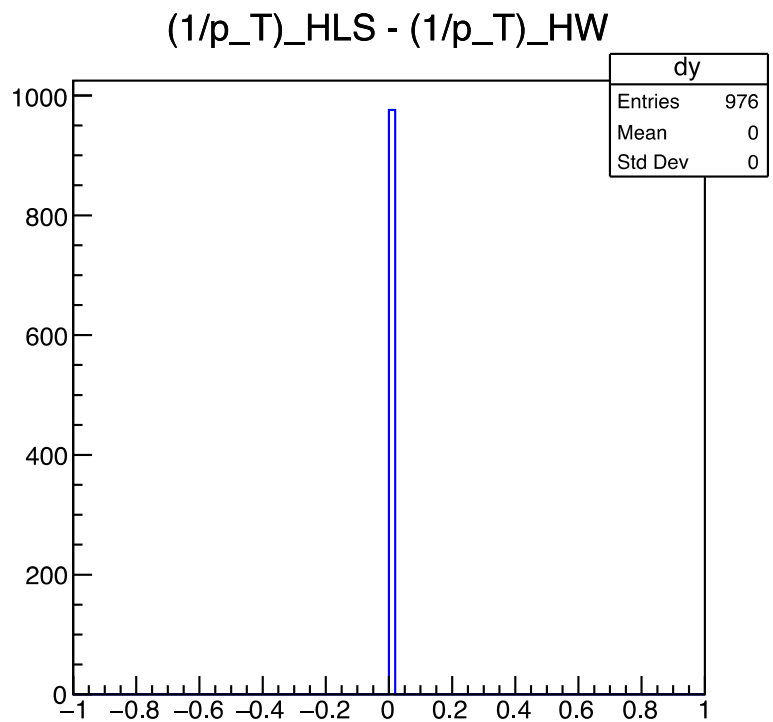
- Plan to develop and run NN in parallel in Phase-1 EMTF for Run 3
  - If able to fit in the MTF7 board
- Develop a Run 3 NN prototype
  - Without GE1/1 for now. It will be added in the future.
  - **Using only CSC and RPC inputs.** Retrain the NN with 31 features.
    - NN size is reduced to 24/16/12 nodes in the 3 hidden layers.
    - Only 28% DSP utilization (with 250 MHz clock)

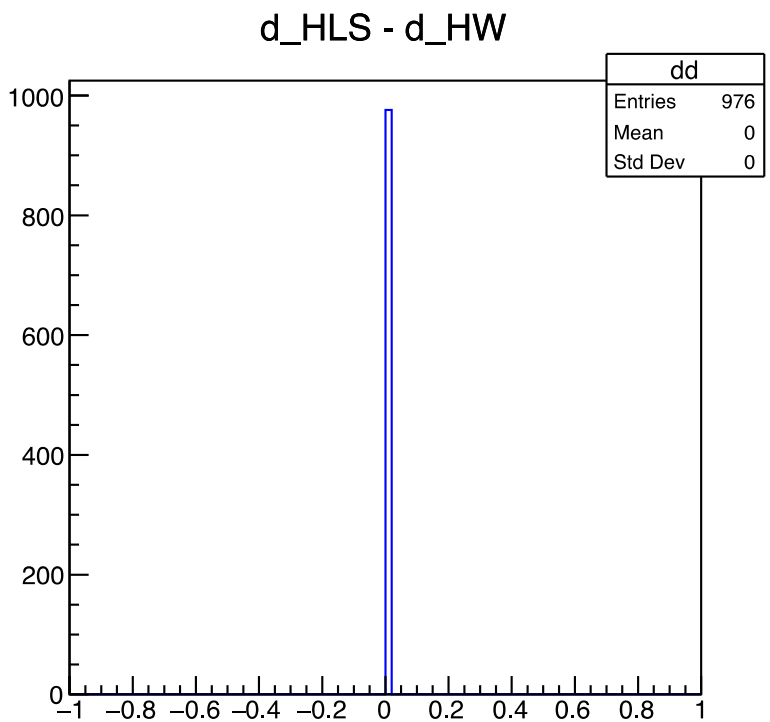
	ME1/1	ME1/2	ME2	ME3	ME4	RE1	RE2	RE3	RE4	GE1/1	GE2/1	ME0
ϕ	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
θ	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
bend	✔	✔	✔	✔	✔							✔
F/R	✔	✔										✔
ring			✔	✔	✔							

+ pattern straightness
+ zone
+ median theta

Large scale HW tests

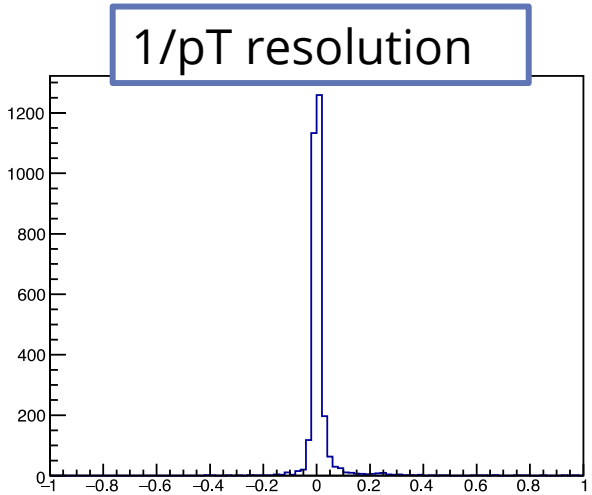
- Inject ~1000 muons in a pipelined burst
- Compare HW outputs (q/p_T and PU discriminator) with Vivado HLS simulation
  - Perfect agreement observed!

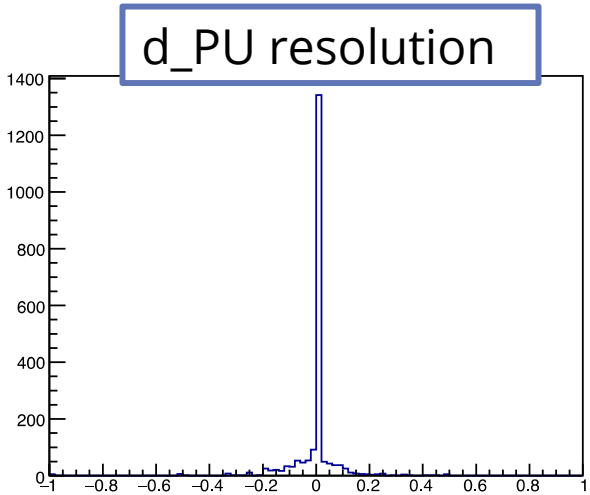


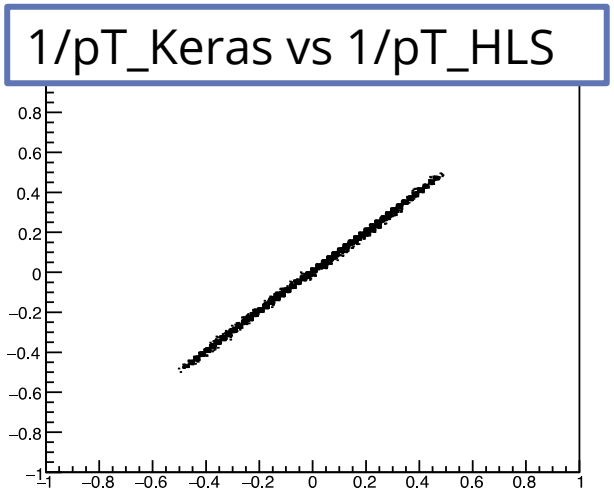


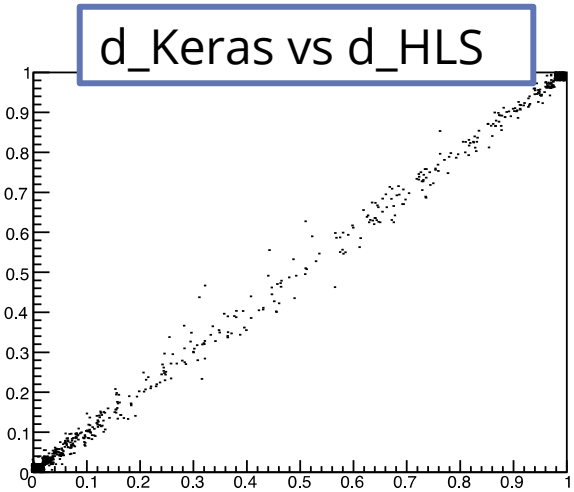
Large scale HW tests

- Compare Keras vs HLS simulation outputs. Keras uses floating-point precision, HLS simulation uses <18,8> precision. Resolution is typically at 10^{-3} level.









Summary

- A new NN has been trained to combine trigger primitives from various muon detectors to determine the muon p_T.
- A lot of progress made in the **NN implementation in FW & HW**.
  - HLS4ML toolkit allows to convert NN model into VHDL codes, which can be used to generate the firmware. HLS estimates of resource usage are accurate.
  - NN prototype firmware has been implemented on the real hardware (MTF7). HW outputs have perfect agreement with HLS simulation using fixed-point precision.
- FW development plans:
  - Plan to **develop and run NN** in parallel in Phase-1 EMTF **for Run 3**.
- Exciting future possibilities:
  - Possible with Convolutional Neural Network? It could potentially be used for the pattern recognition step. There are many more advanced types of DNN developed by the industry (Google, Amazon, autonomous vehicle industry, etc), which we could also try.