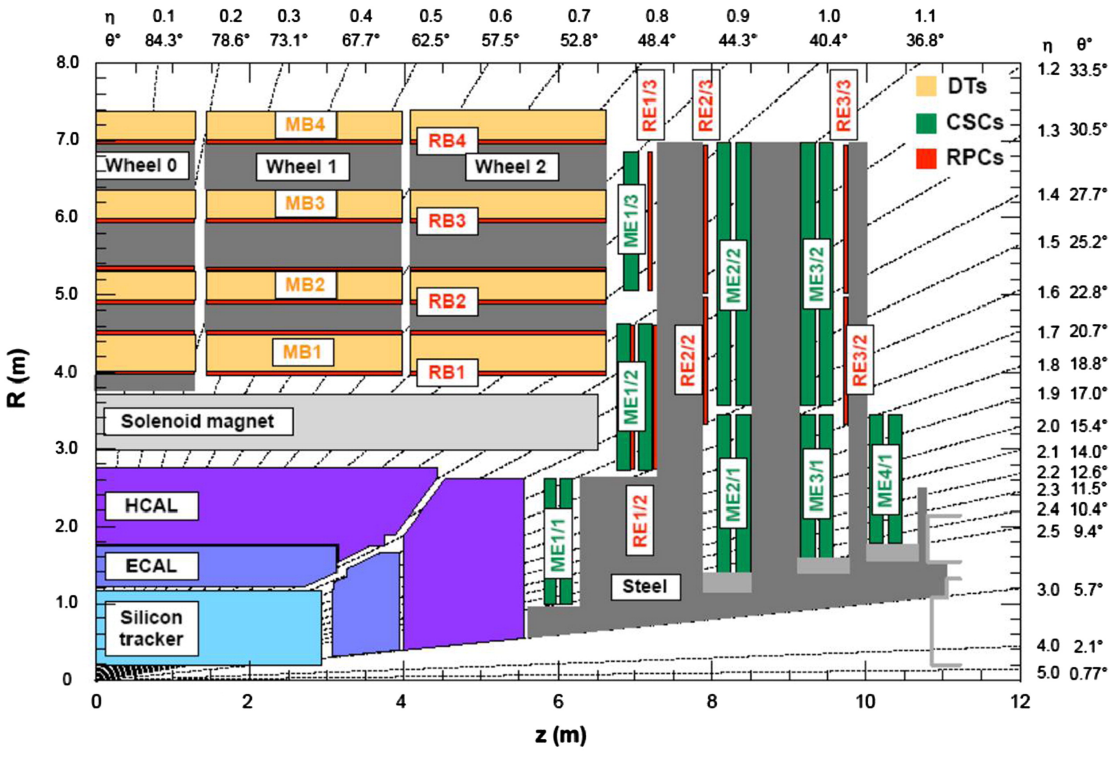
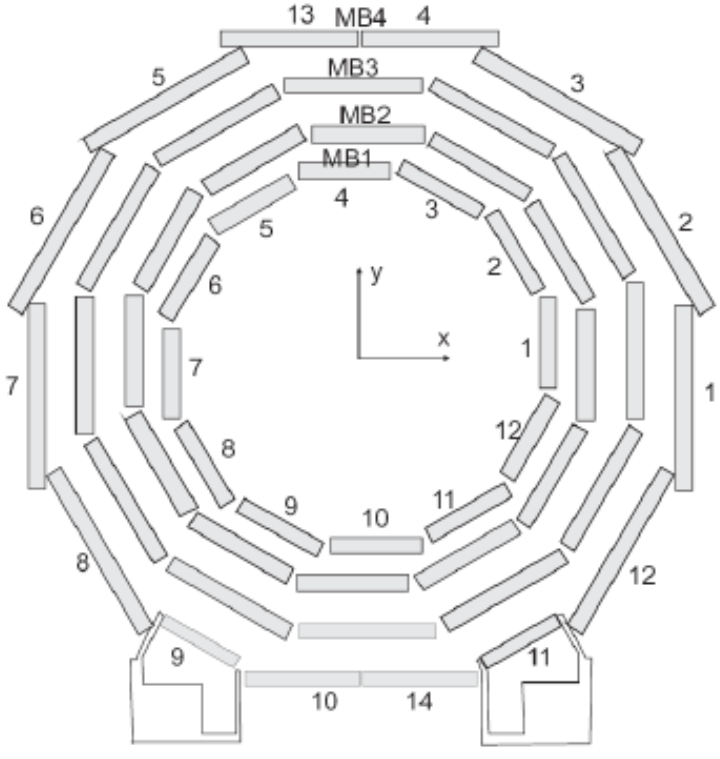
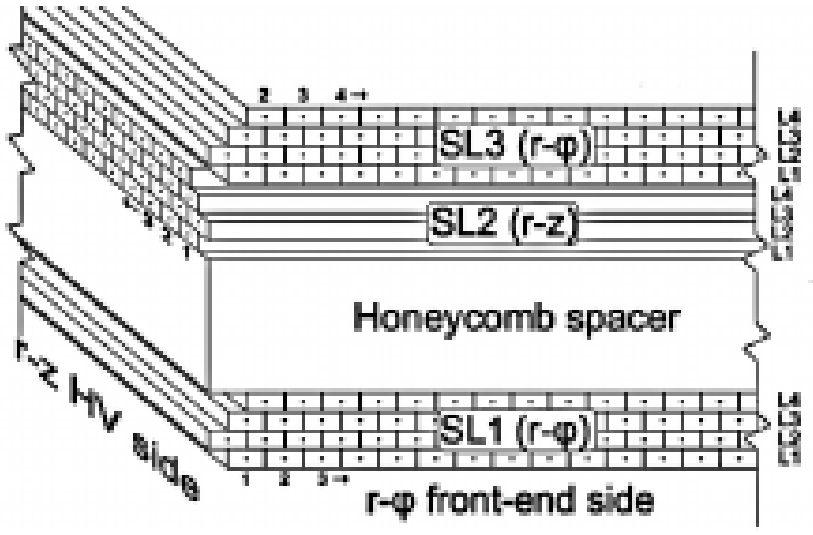
Using the Outer Hadron Calorimeter in the TwinMux emulator for Level 1 muon trigger

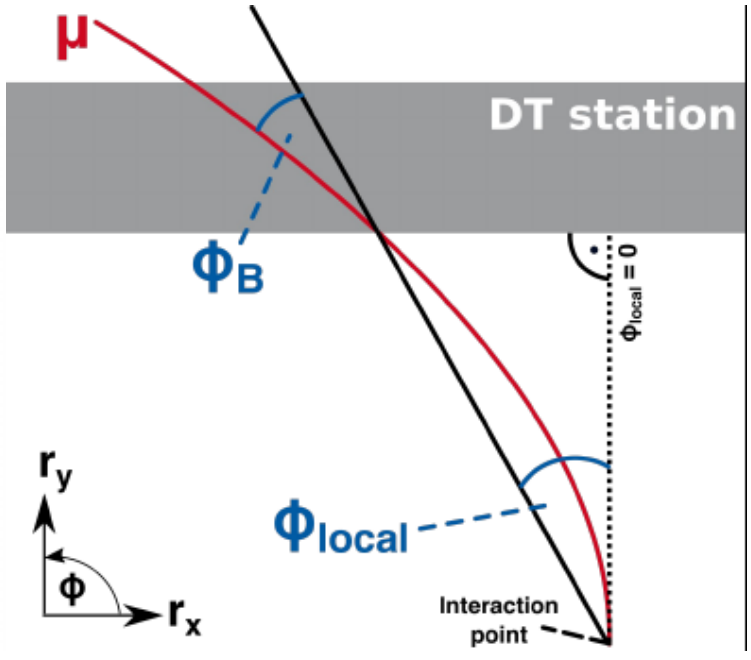
The Muon Drift Tube (DT) system [1]





The Muon Drift Tube (DT) system [2]



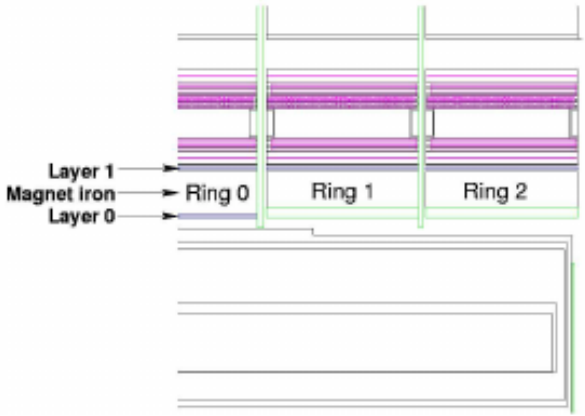


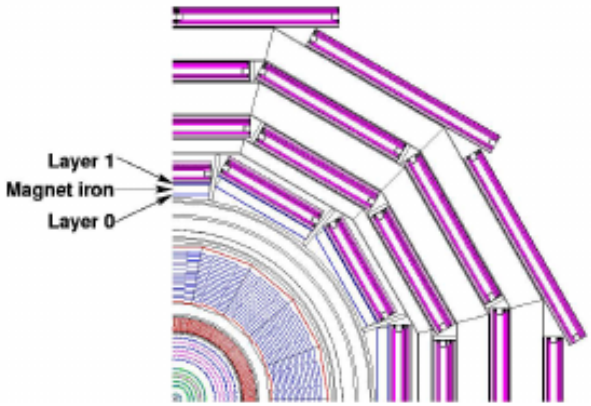
- 4 staggered layers form 1 SuperLayer (SL).
- $SL_{r\text{-}\phi}$ have wires parallel to the beamline, and measure quantities in the r-ϕ plane.
- SL_z have wires perpendicular to the beamline, and measure quantities in the r-z plane.
- In MB1/2/3, one chamber is formed by 2 $SL_{r\text{-}\phi}$ and 1 SL_z.

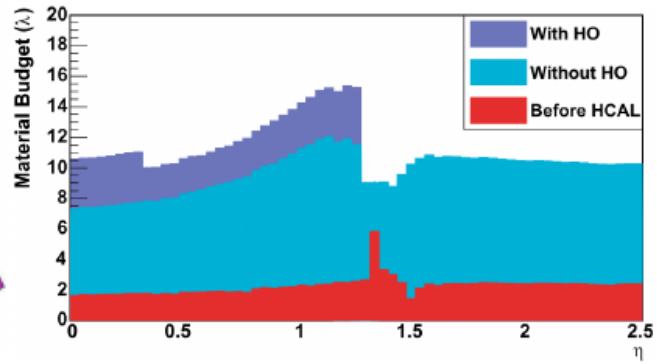
The Muon Drift Tube (DT) system [3]

- In MB4, one chamber is formed by only 1 $SL_{r\text{-}\phi}$.
- The DT chambers provide Trigger Primitives (TPs) which store information about the location of the hit, number of aligned DT-hits, the bending angle ϕ_B etc. It also contains a quality code which indicates the number of SL hits and the how well aligned they are.

The Outer Hadron (HO) calorimeter







- $|\eta| < 1.4$.
- Central ring has 2 layers. The others have 1.
- Segmentation: $\Delta\eta \times \Delta\phi = 0.087 \times 0.087$.
- The HO provides Trigger Primitives (TPs) which among other things, store the η-ϕ location information, and also a mip-bit which stores whether the hit is mip-like (within certain thresholds) or not.

Need for the HO

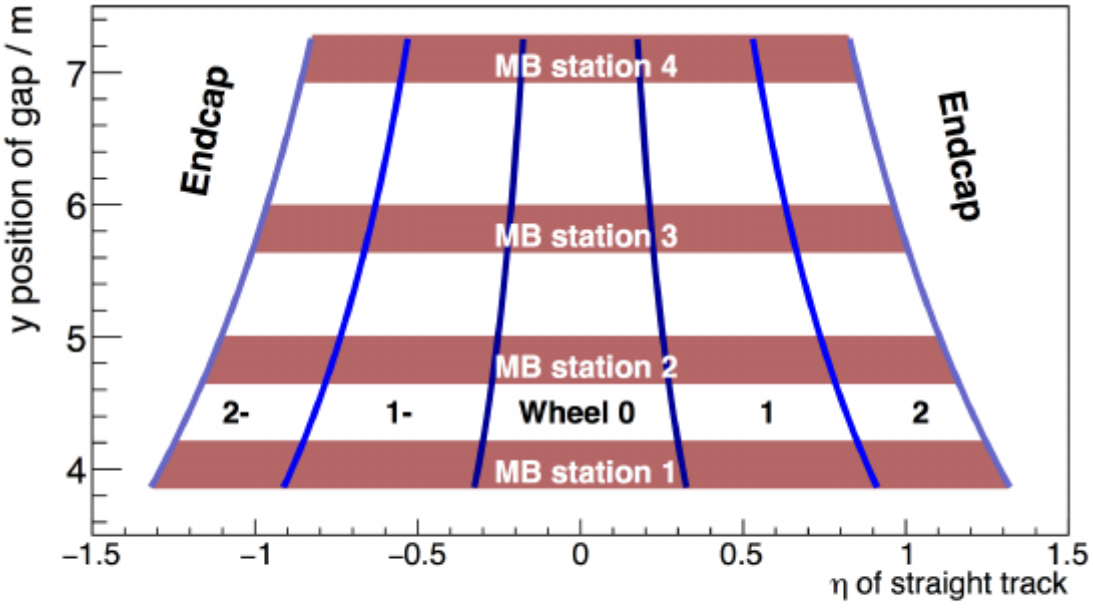
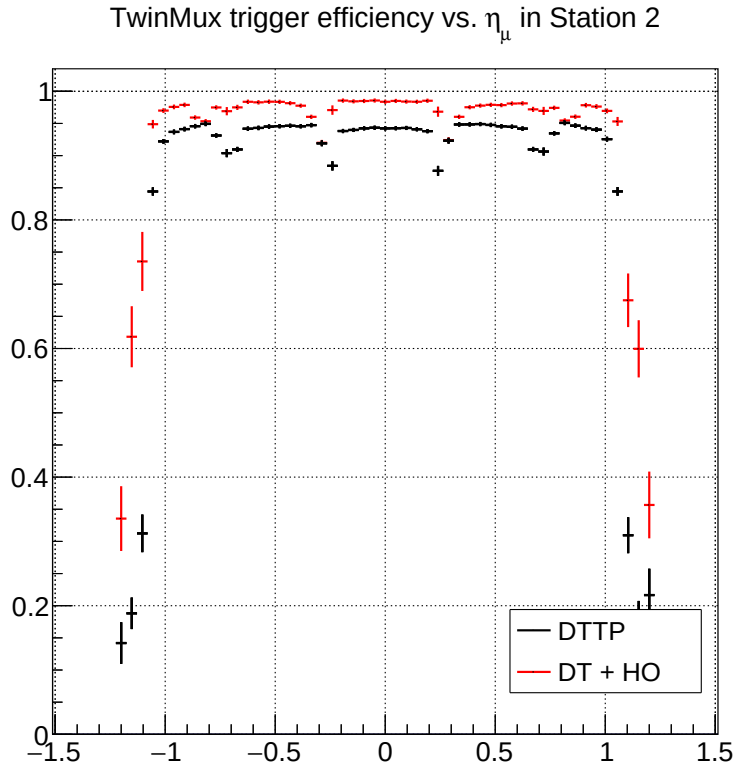
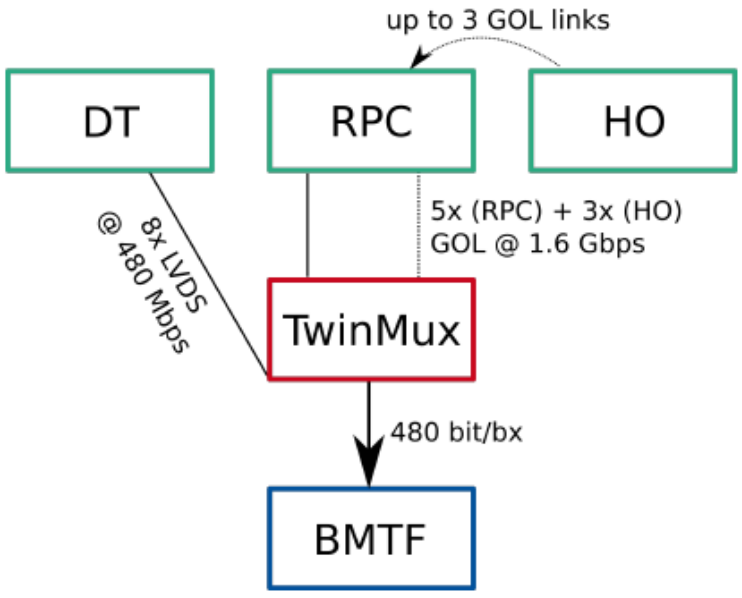


Figure 2.11: y projection of the muon chambers (MB station 1–4). A straight muon track has a fixed η value on the x-axis. This plot allows to easily see which MBs are hit by a muon with a straight track. The blue lines mark the wheel borders.



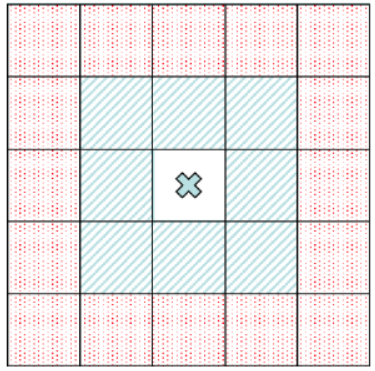
- Efficiency in muon barrel gap regions is much lower.
- Scenario where a DT station fails.
- Since the HO covers the muon barrel gap regions (along constant η), it can be used to aid the muon track finder.

The TwinMux



- The DT-TPs and HO-TPs are collected by the TwinMux (a μTCA board), along with information from the RPC (mainly for timing), and sent to the Barrel Muon Track Finder (BMTF).
- The TwinMux emulator is a piece of code that emulates this behaviour, and can be used for stuff like testing new algorithms.
- Currently, the HO-TPs are not sent by the TwinMux to the BMTF.
- I am implementing an algorithm to decide if and when will an HO-TP be used to support the DT-TPs.

The algorithm [1]



Loop over all the DT-TPs in the following manner:

1. Take a DT-TP in say the first station/barrel.
2. Decide whether the DT-TP is Low Quality ($0 < LQ < 4$) or High Quality ($3 < HQ < 7$).
3. For LQ, try to find a matching HO-TP in the same wheel as the DT-TP such that $\Delta i\phi < 1$.
4. For HQ, try to find a matching HO-TP such that $\Delta i\eta \times \Delta i\phi < 1 \times 1$, i.e. within a 3×3 tile window.
5. If a matching HO-TP is found, then the quality code of the DT-TP is modified to indicate that this DT-TP has support from the HO.

The algorithm [2]

6. This modified DT-TP, say DT-TP' (primed) will be treated differently in the BMTF.

The modified DT-TPs can be used by the BMTF to:

- recover muons in case of a muon barrel failure.
- recover muons in the barrel gap regions where the efficiency is low.

Datasets

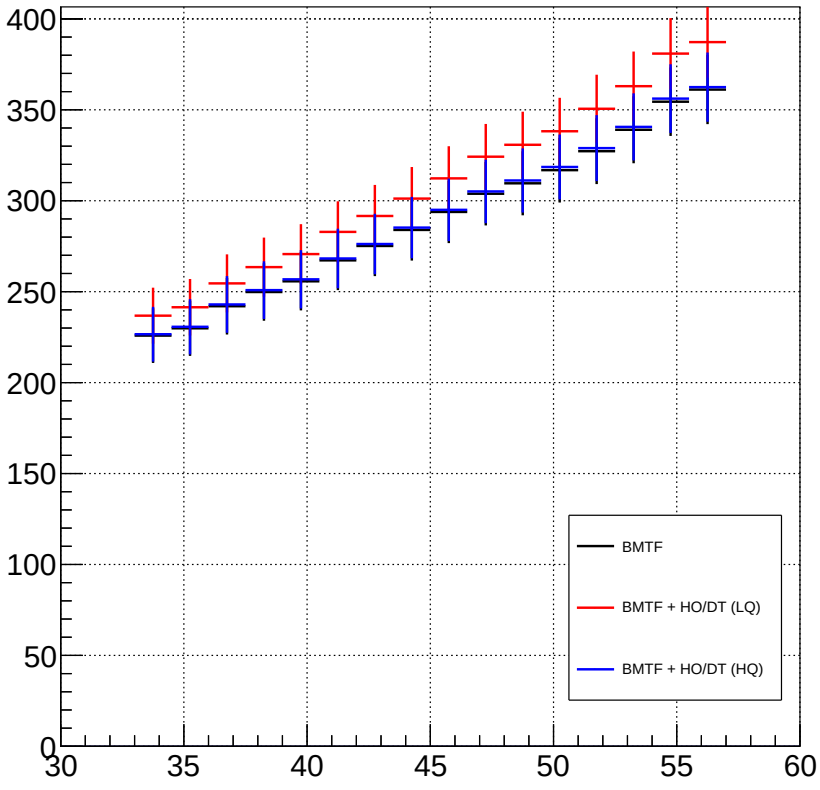
- `/ZeroBiasBunchTrains[0-5]/Run2016H-v1/RAW`

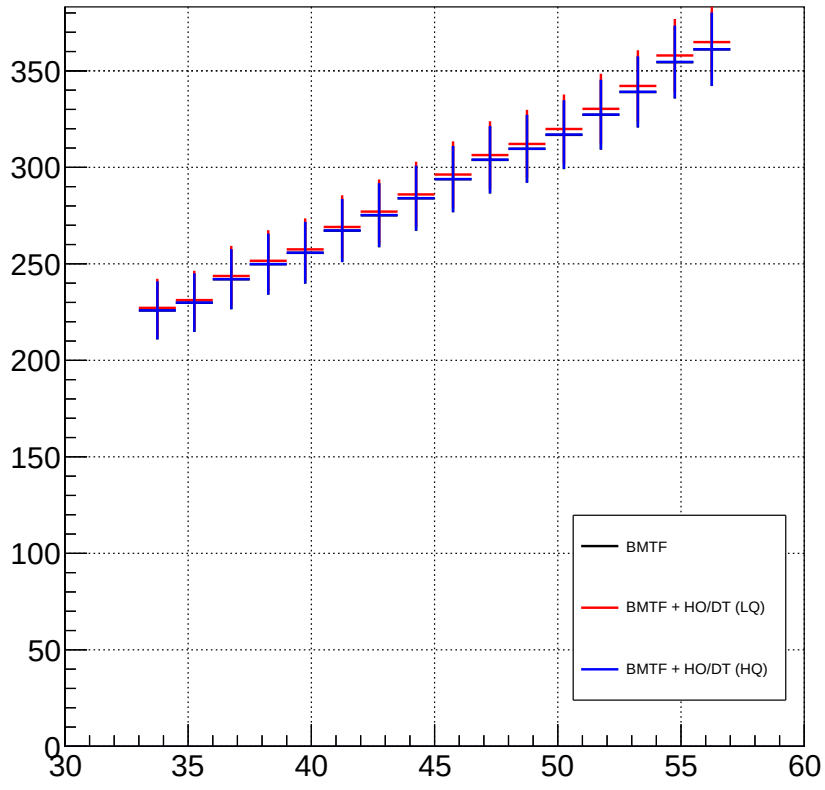
Algorithm performance: L1 trigger rate [1]

- Have to check that the rate increase of the L1 trigger is not too high.
- Unprescaled means $nBunch = 1$; can be scaled to any arbitrary bunch filling. The p_T cut is 3 GeV.
- As can be seen, the rate increase is tolerable.
- Have to resolve issue with HOTP mip-bit.

Algorithm performance: L1 trigger rate [2]

Unprescaled rate [Hz] vs. PU





The HOTP mip-bit [1]

- Accessing mip-bit (suggested by Christopher West):

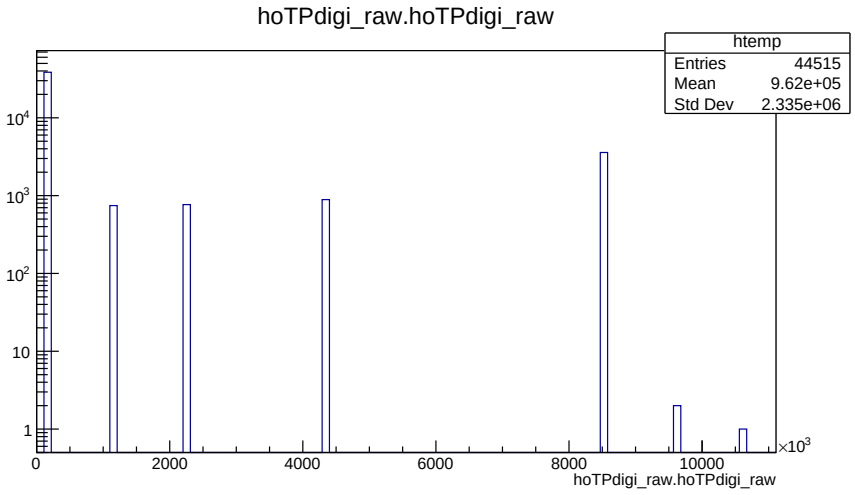
```
(hoTP->bits()>>hoTP->whichSampleTriggered())&0x1
```

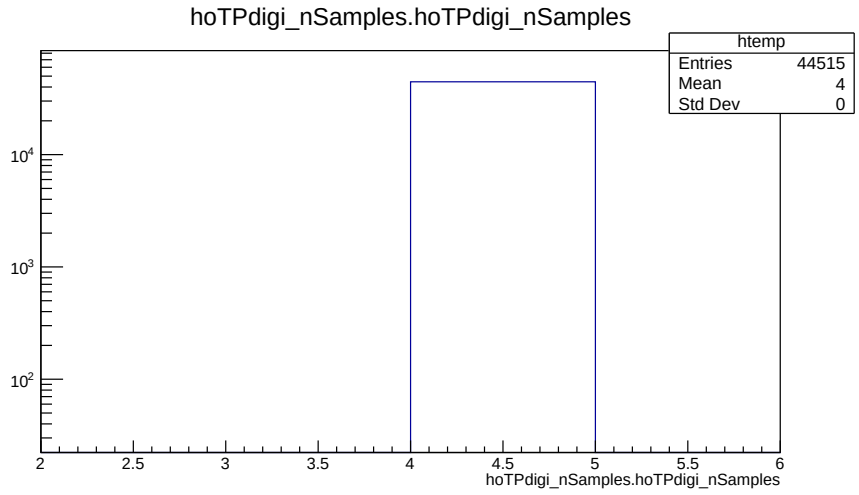
- HOTP unpacker is `HOTriggerPrimitiveDigi.h`; will use the TwinMux unpacker once it's ready.
- Naively, hotpDigi.bits() can take values from 0 to 1023 ($03FF = 1023$); will depend on the value of the raw (packed) Trigger Primitive (uint32_t theHO_TP).

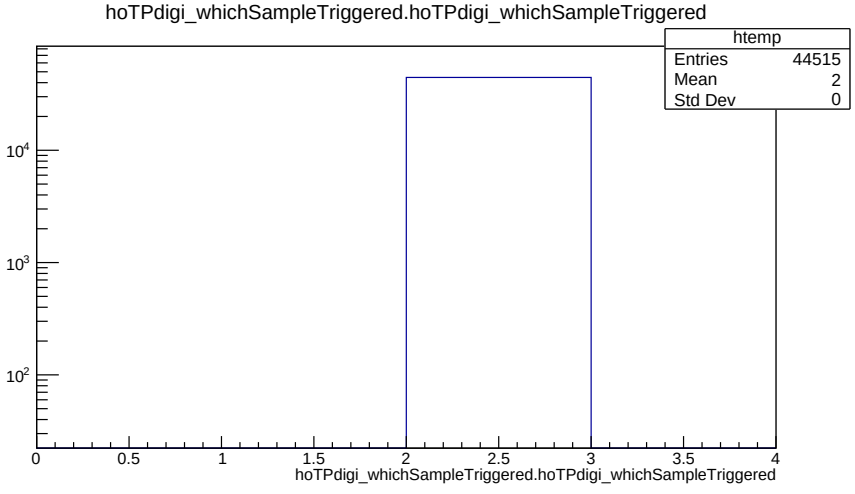
```
/// get the raw (packed) Triger Primitive
uint32_t raw() const { return theHO_TP; }
/// get the number of samples used to compute the TP
int nsamples() const { return (theHO_TP>>12)&0x000F; }
/// get the number of the triggering sample
int whichSampleTriggered() const { return (theHO_TP>>16)&0x000F; }
/// get the single-bit data
int bits() const { return (theHO_TP>>20)&0x03FF; }
```

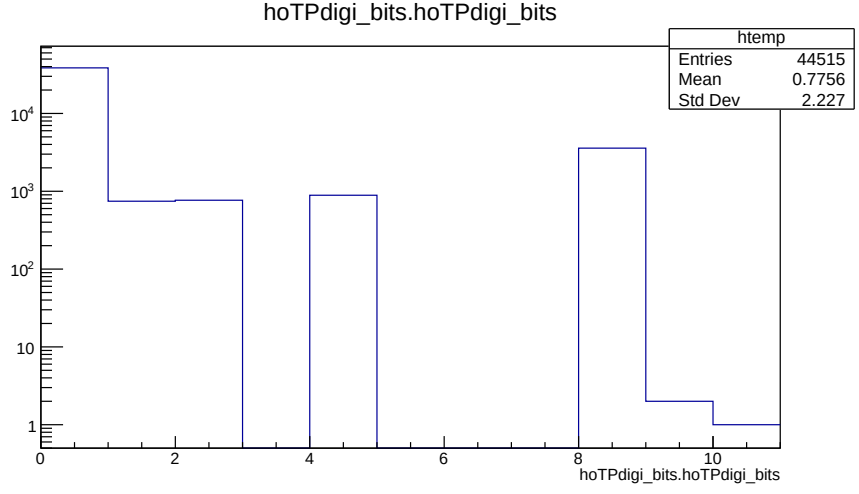
- Discussion with HO-experts - may be Pooja can elucidate on that.

The HOTP mip-bit [2]









Algorithm performance: DTTP Efficiency [1]

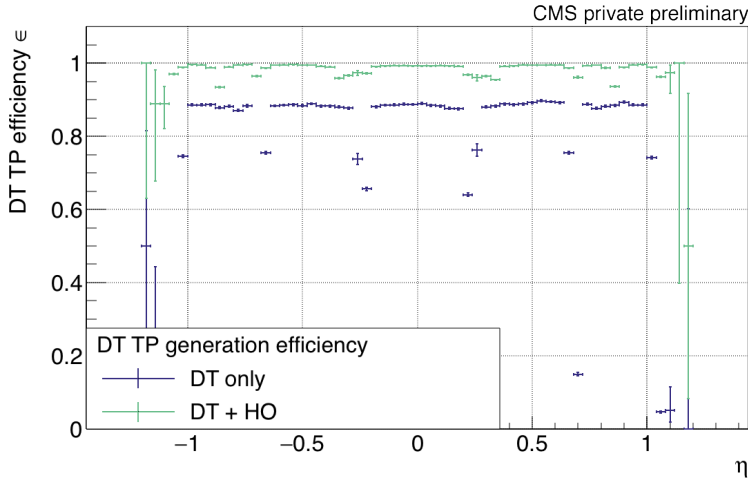
- A DT track segment is reconstructed from a single DT chamber.
- In-time means that the TP must have been recorded at the correct bunch-crossing.
- The efficiencies are computed separately for each muon station (barrel).
- The DTTP efficiency is defined as [1]:

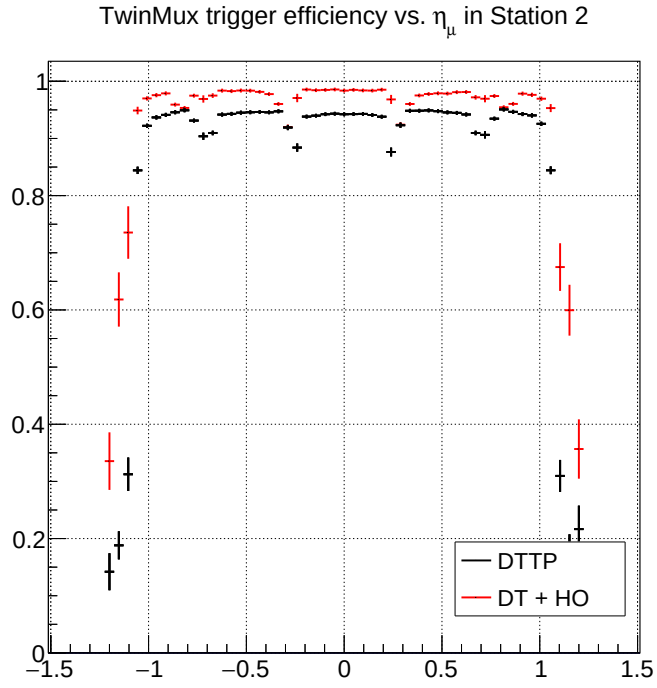
$$\epsilon^{MBi}_{DTTP}\ (\text{in a bin}) = \frac{\#\text{ of in-time DTTPs in MBi matched to the denominator}}{\#\text{ of DT track segments in MBi matched to reconstructed muon tracks}}$$

- Say a DTTP is missing in MB2. Then the MB1 and HO information can be used. The efficiency with HO support is defined as:

$$\epsilon^{MB2}_{DTTP+HOTP}\ (\text{in a bin}) = \frac{\#\text{ of HOTPs matched to in-time MB1 DTTPs matched to the denominator}}{\#\text{ of DT track segments in MB2 matched to reconstructed muon tracks}}$$

Algorithm performance: DTTP Efficiency [2]





- Trend similar to Florian's result.
- One can play around with the cuts used to match the objects to get higher/lower efficiencies.

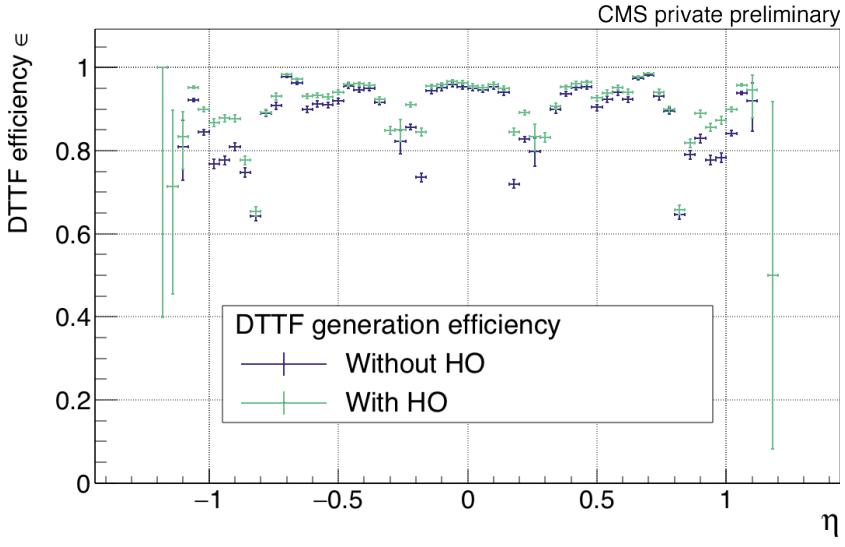
[1] Stefano Marcellini

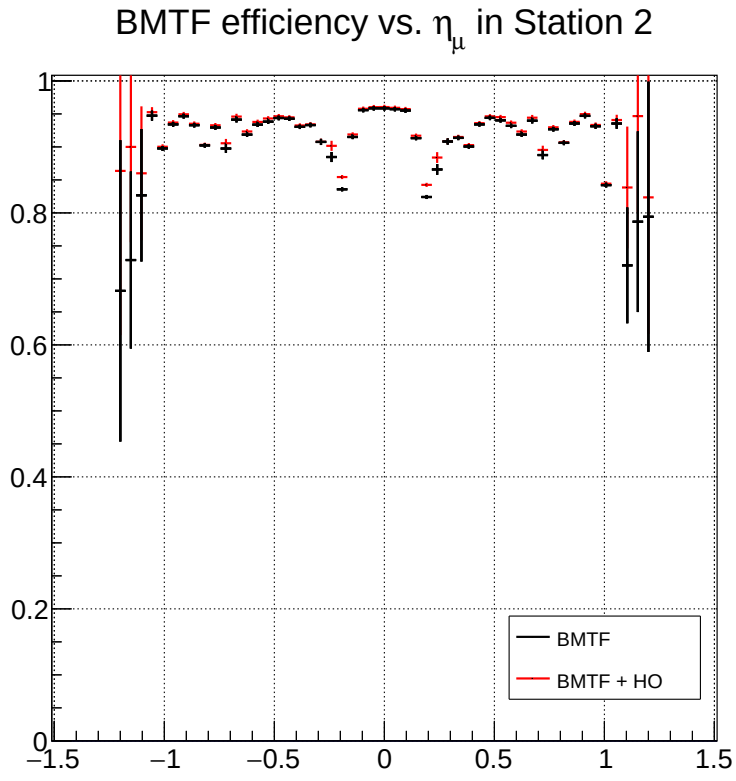
Algorithm performance: BMTF Efficiency [1]

- The BMTF constructs tracks out of DTTPs.
- The efficiencies are computed separately for each muon station (barrel).
- A genuine DTTP is one that is matched to a DT track segment which was in turn matched to a reconstructed muon track.
- The BMTF efficiency is defined as: given there is a genuine DTTP, what fraction of times was it used to create a track.

$$\epsilon_{BMTF}^{MBi} \text{ (in a bin)} = \frac{\#\text{ of BMTF tracks matched to the denominator}}{\#\text{ of genuine DTTPs in MBi}}$$

Algorithm performance: BMTF Efficiency [2]





- Trend similar to Florian's result - mainly the wheel gap regions are affected,
- My result seems to have slightly lower increase in efficiency - as before, can play around with the cuts.

Summary

- Will use the TwinMux unpacker once it's ready.
- Issue with the HO mip-bit needs to be resolved.
- Have partially implemented the change in the TwinMux emulator - however, cannot check actual performance unless the BMTF algorithm is changed to use the modified DTTPs.