DEEP UNDERGROUND
NEUTRINO EXPERIMENT

HORIZONTALMUON ALGORITHM FOR VD COLDBOX TRIGGER

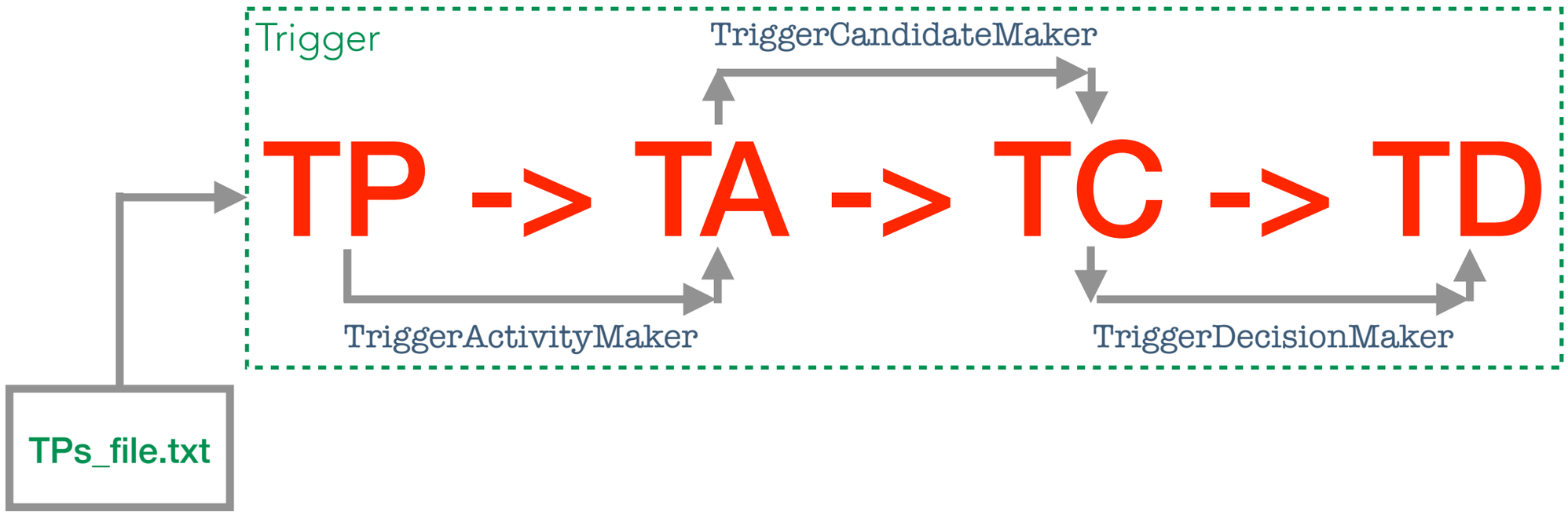
Alexander Booth

FD Data Selection

October 19, 2021

Queen Mary
University of London

Motivation & Implementation



- A simple trigger that can be used to look for horizontal muons at the VD coldbox test stand at CERN - extension of SumADCSimpleWindow.

- Monitor the total number of (collection) channels with a hit of incoming TPs in a given time window (`window_length`), trigger if that total goes above some threshold value (`n_channels_threshold`).
  - Both `window_length` and `n_channels_threshold` are configurable members of the `TriggerActivityMakerHorizontalMuon` class.
  - Defaults obtained based on the data in tps_link_11.txt (obtainable from cernbox).

- All of the logic is implemented in the `TriggerActivityMakerHorizontalMuon` class. A nested class called `Window` monitors incoming TPs (which are time ordered).

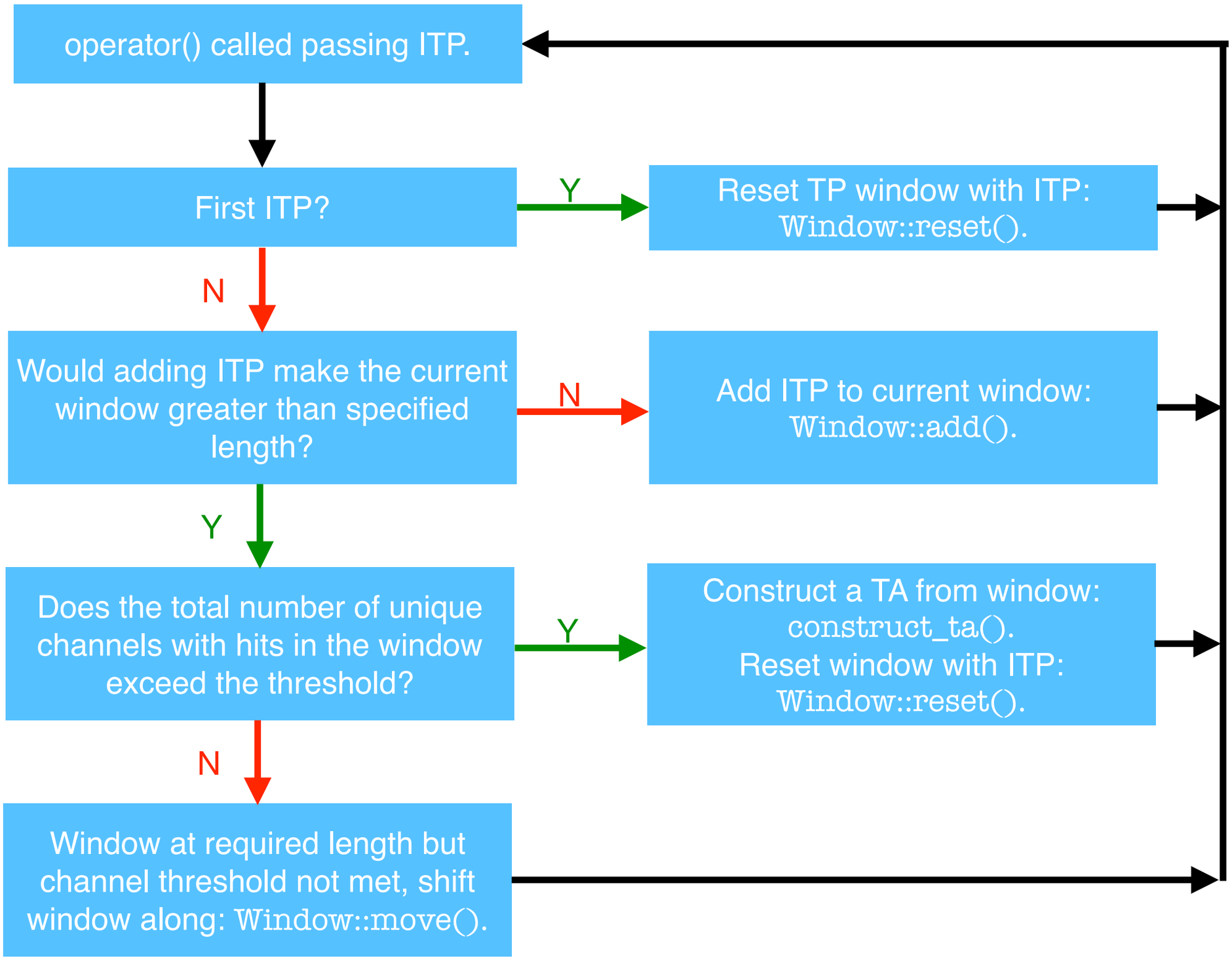
- When a particular window exceeds threshold, `TriggerActivityMakerHorizontalMuon::construct_ta()` constructs a `TriggerActivity`. This is enough (eventually) to request data to be read out. The `TriggerCandidateMakerHorizontalMuon` class simply constructs trigger candidates one for one from the trigger activities.

- HorizontalMuon retains all of the logic of SumADCSimpleWindow.

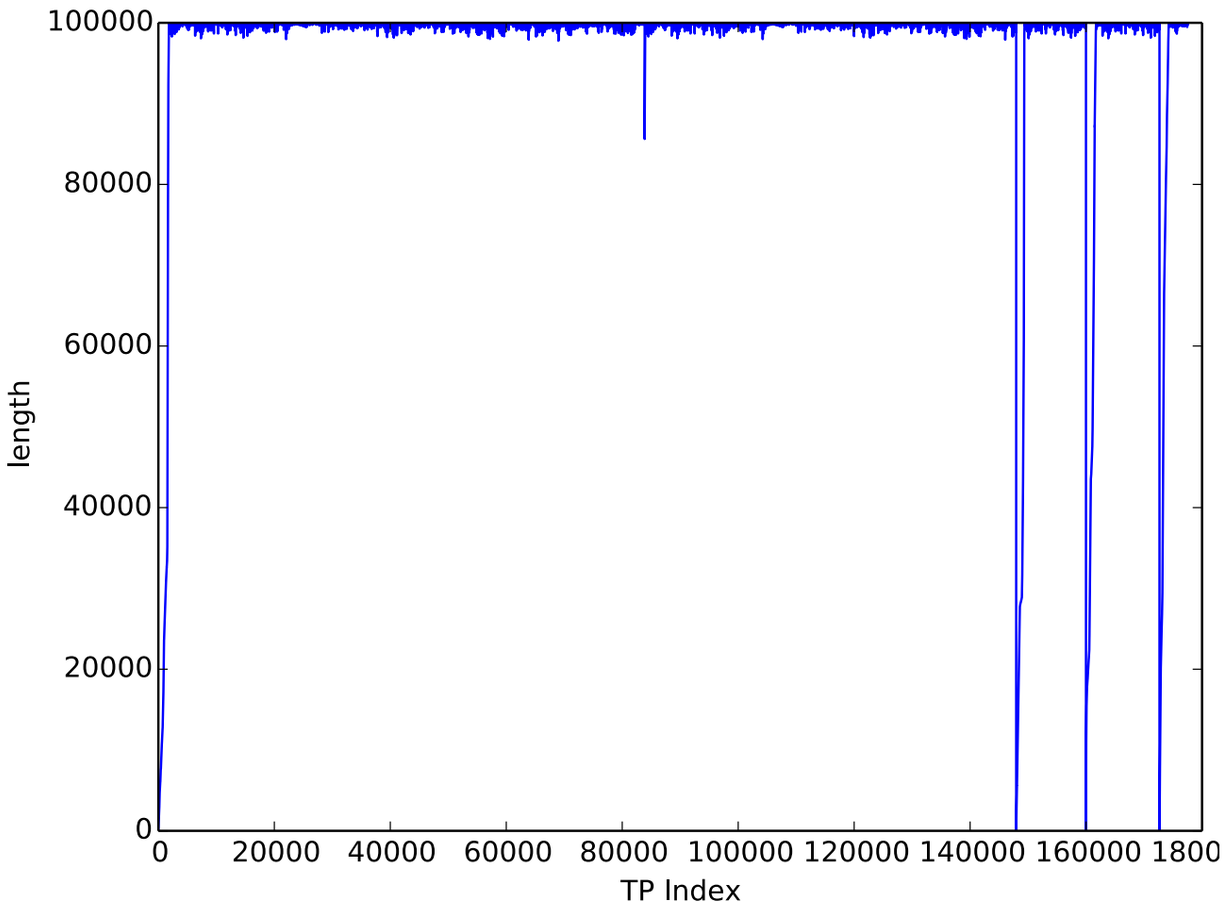
Window: TriggerActivityMaker Nested Class

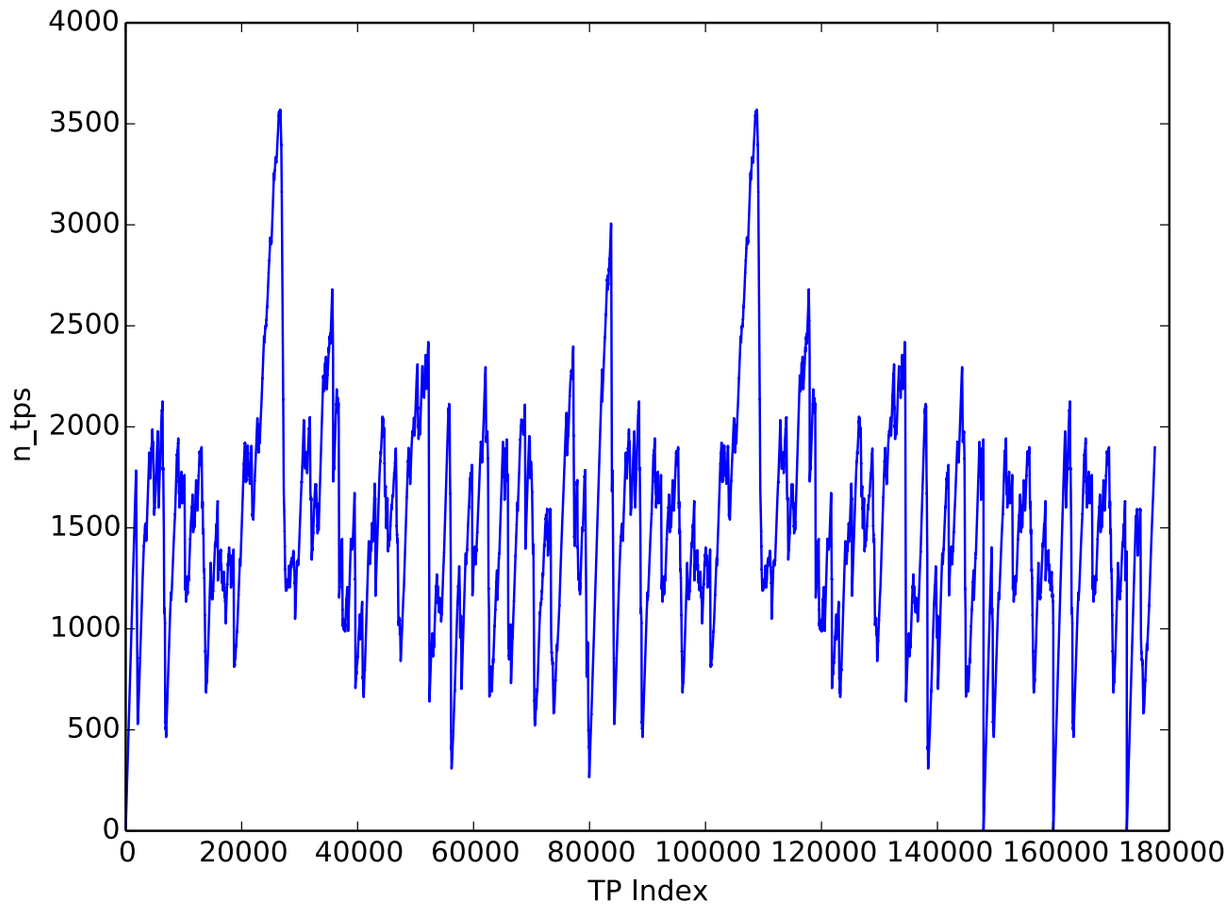
- Most of the logic is dealt with by the `Window` class, a class nested inside `TriggerActivityMakerHorizontalMuon`.
- It contains three member functions and five functions. `operator<<` is also defined. The following explanations can be used to interpret the flow chart on the next page:
  - `time_start`: Start time of the first TP chronologically in the window.
  - `adc_integral`: Total ADC of the TPs in the window.
  - `channel_states`: A record of channels with hits. ←
  - `tp_list`: A vector of TPs in the window.
  - `is_empty()`: If the vector of TPs is empty, returns true, else false.
  - `clear()`: Clears the vector of TPs.
  - `move()`: Finds all of the TPs in the current window which need to be removed if the input TP is to be added and the earliest and latest TPs in the window aren't separated by more than `window_length`. The earliest TPs in the window are removed (their ADC and channel contribution subtracted) until this condition is satisfied. `time_start` for the window is then set to what is now the first TP start time. Note that in the case that the input TP is further away than 1 `window_length` from the latest TP in the window, the window is reset with the input TP.
  - `reset()`: Clear the TP list, restart the ADC sum and hit channel list, set the start time of the window to the start time of the input TP. Add, the input TP to the window TP list.
  - `add()`: Add input TP to window TP list, its ADC to window ADC total and its channel to the hit channel list.

Logical Flow: TriggerActivityMakerHorizontalMuon



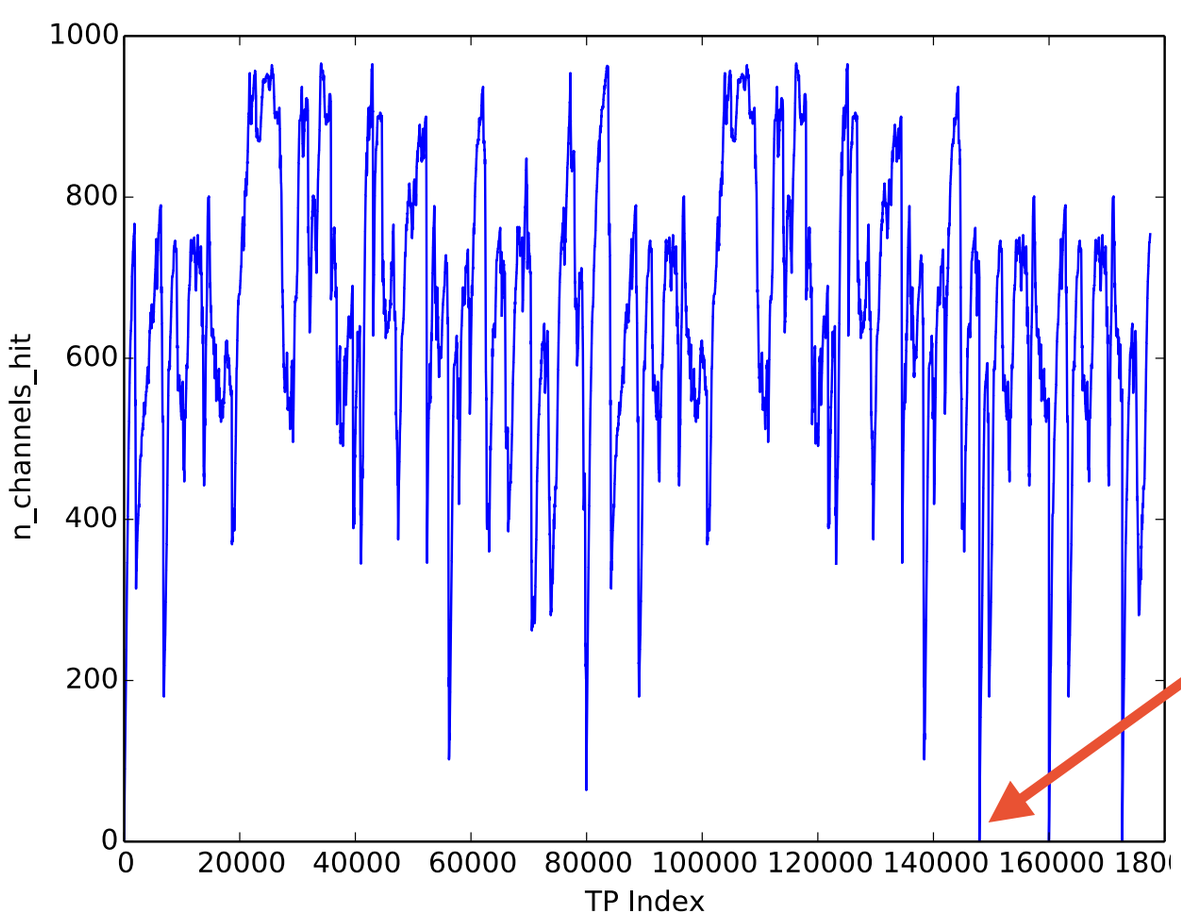
Sanity Check Plots: Channel threshold set above obtainable value

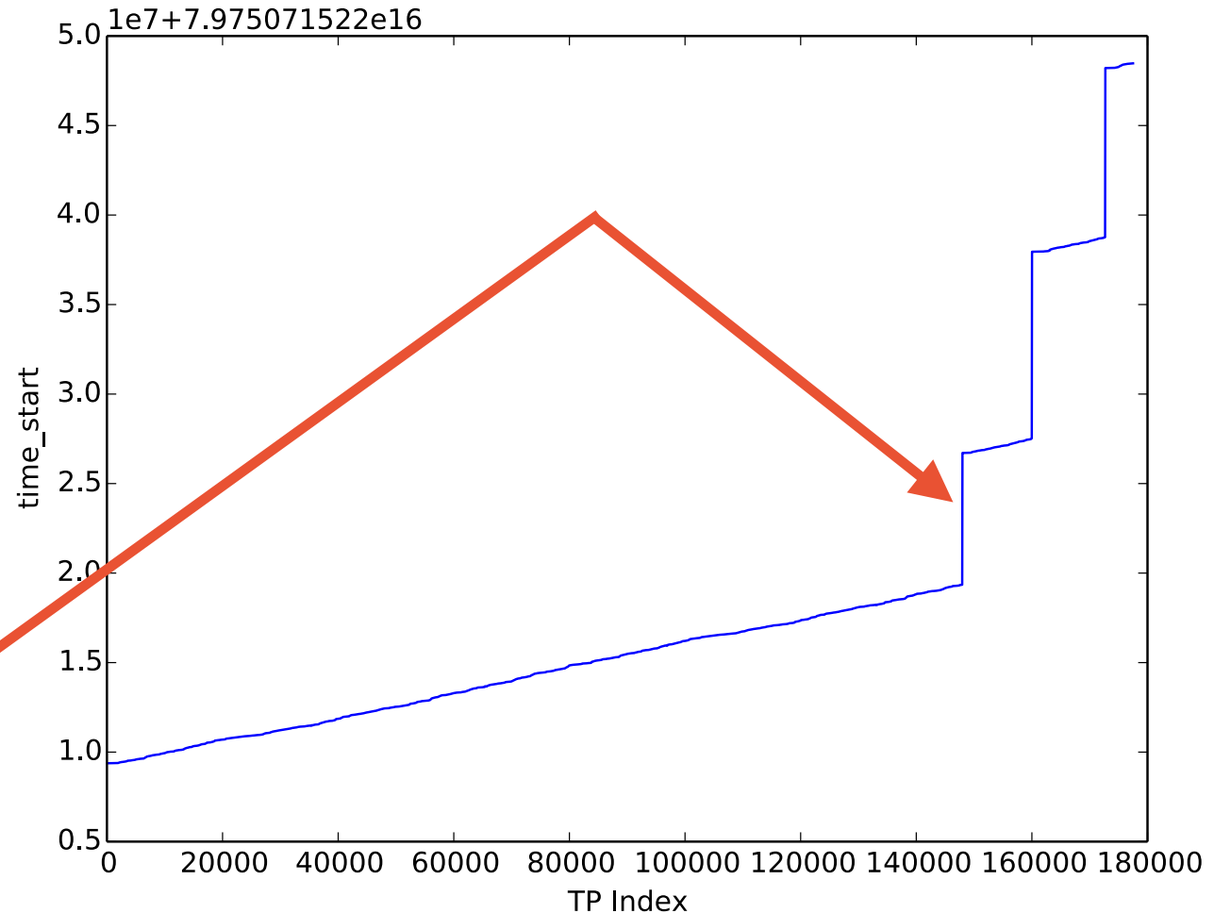




- The length of the current window in time window never exceeds the user specified length.
- Drops to zero: in order for an input tp to be added to the current window, the difference in time between it and the first tp already in the window cannot exceed the maximum window length.
- If the difference in time between subsequent tps is longer than the maximum length of the window, the window is emptied.

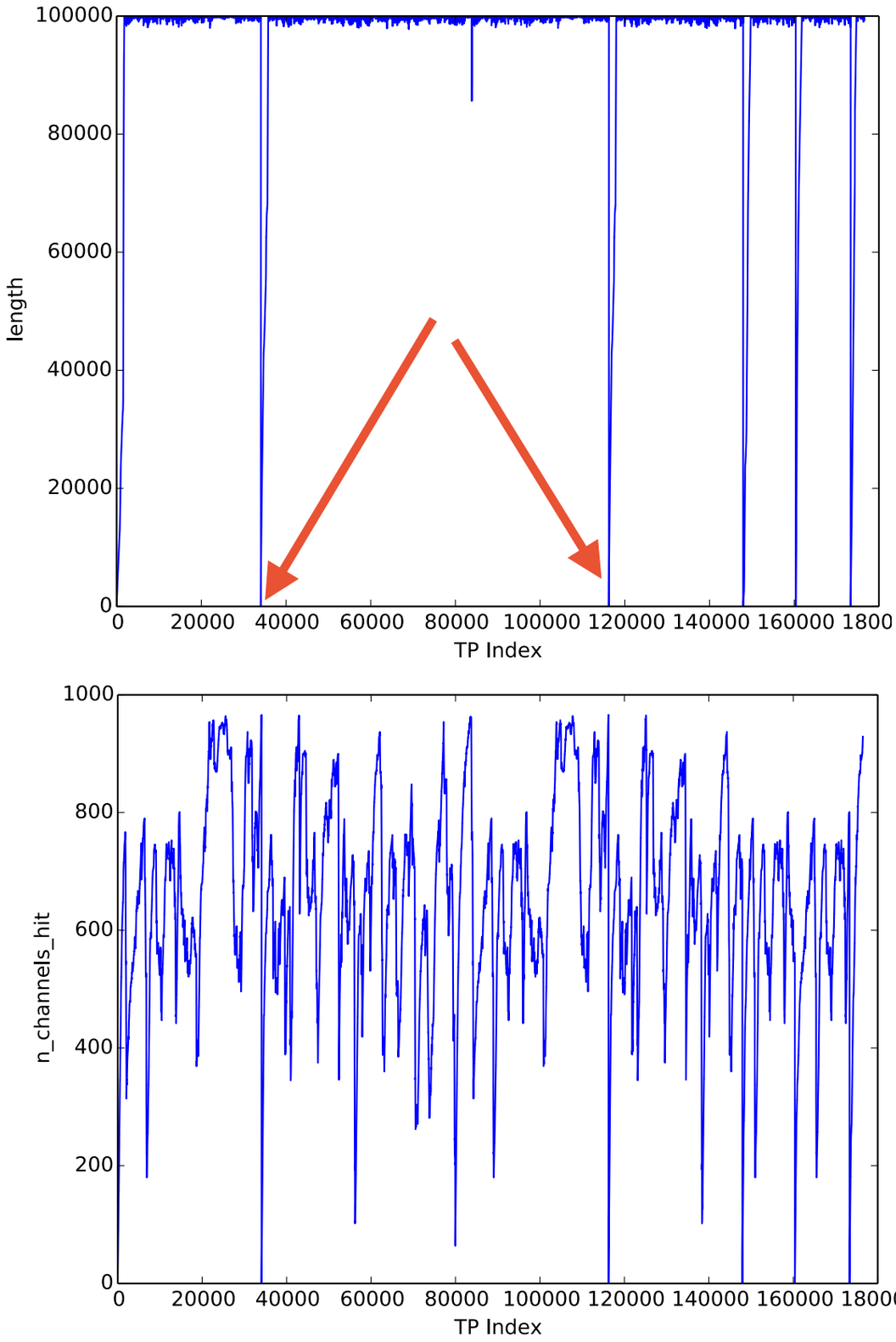
Sanity Check Plots: Channel threshold set above obtainable value

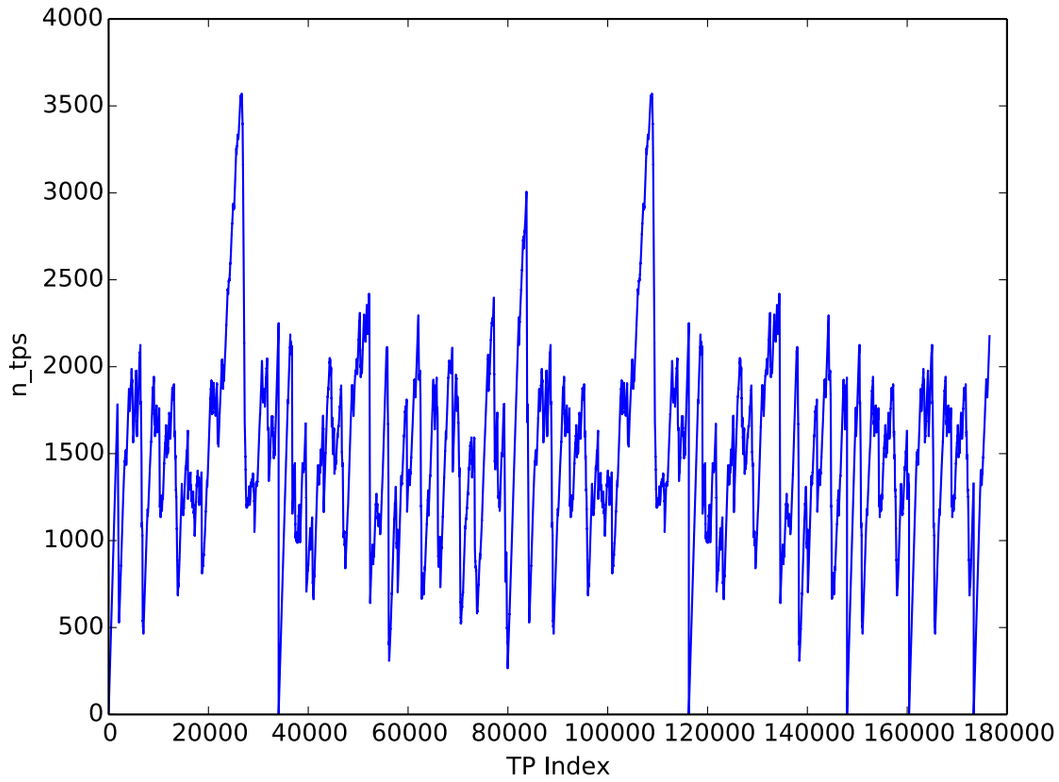




- Can see that the spikes down to 0 correspond to big differences in the start time of the windows.
- From this plot, chose a triggering threshold of 965 channels. Leading to 2 trigger activities being created and passed on to candidate maker.

Sanity Check Plots: Channel threshold set at 965





- Number of channels never goes above the threshold.
- Drops to zero in number of TPs in window and window length correspond to where the window gets to threshold and, as before, where there is a large gap between subsequent TPs.

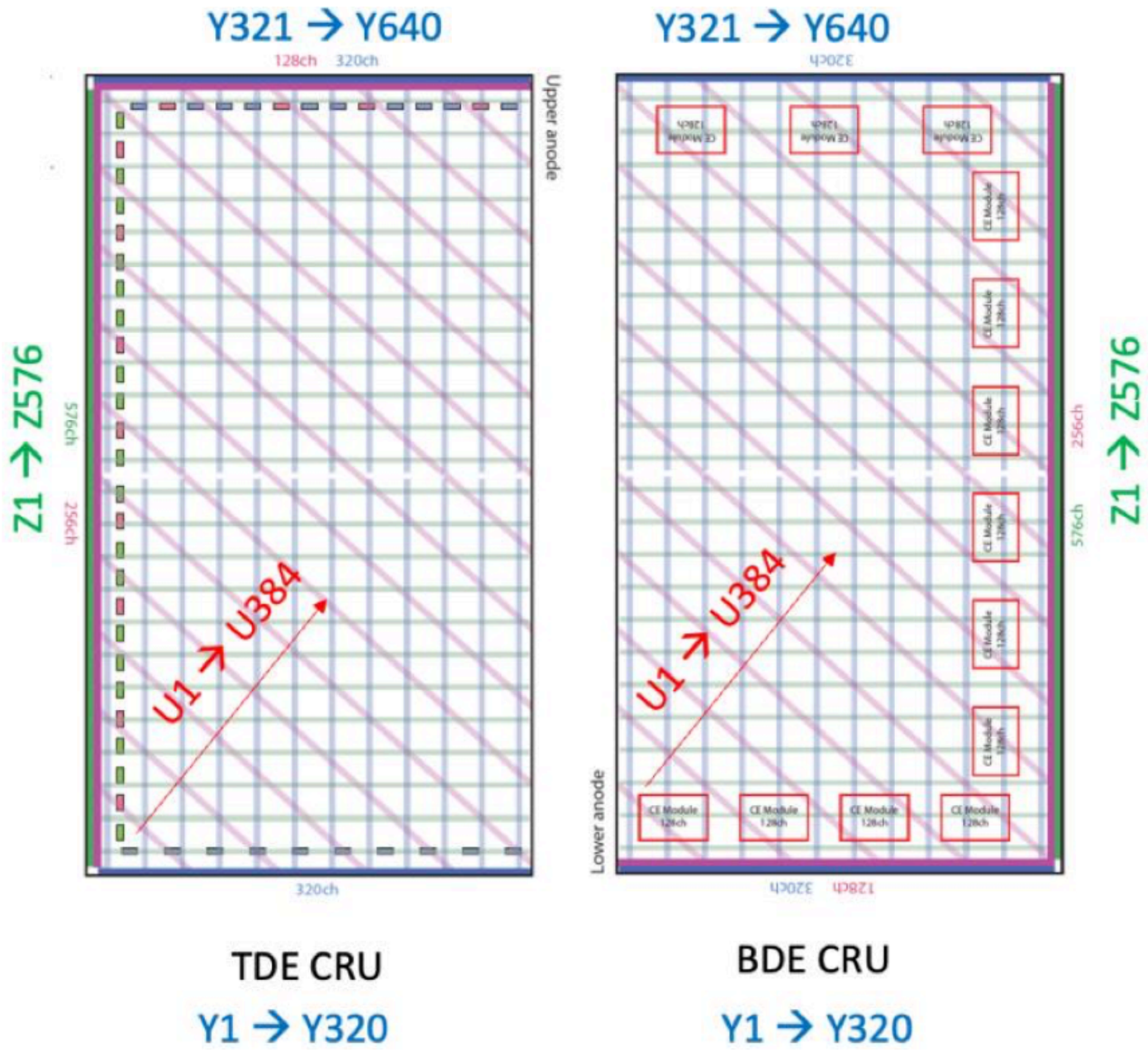
Making the Algorithm More Physics Ready

We have an algorithm that has been tested on fake TP files that can issue a trigger if the number of channels hit in a given window of time goes above some threshold in a given “detector element”.

- **It does not yet consider channel adjacency**.
- **It does not yet compare trigger activities across “detector elements” - TriggerCandidateMaker**.

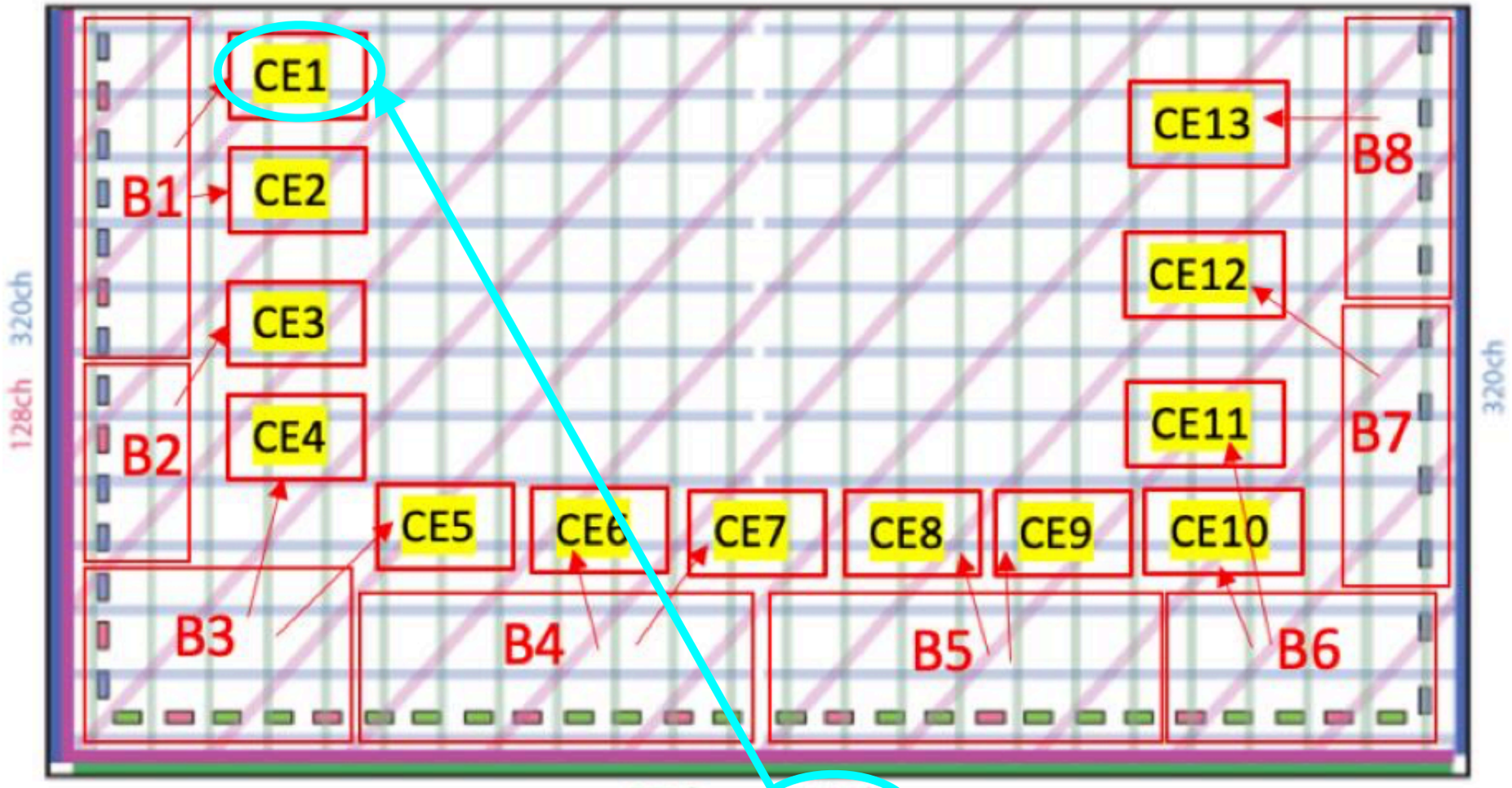
VD Coldbox Geometry

Info gathered from here.



- VD cold box test stand is installed with **two** charge readout units (**CRUs**).
  - Top drift electronics (**TDEs**).
  - Bottom drift electronics (**DBEs**).
- 1600 channels / CRU, 576 "z" **collection channels**.

Channel Map



1600 channels -> 8 adapter boards -> 13 CE modules -> 4 WIBs

FEMB1 -> WIB1 connector 1 → 1 TAMaker?
FEMB2 -> WIB1 connector 2
FEMB3 -> WIB1 connector 3
FEMB4 -> WIB1 connector 4
FEMB5 -> WIB2 connector 1
FEMB6 -> WIB2 connector 2
FEMB7 -> WIB2 connector 3
FEMB8 -> WIB2 connector 4 → 1 TAMaker?
FEMB9 -> WIB3 connector 1
FEMB10 -> WIB3 connector 2
FEMB11 -> WIB3 connector 3
FEMB12 -> WIB3 connector 4
FEMB13-> WIB4 connector 1

Channel Map

AB4 has 2 CE Boards, #6 and #7
Total Channels = 256
U179 - U256
Z111 - Z288

Strip Number	Connector #	Connector Pin #	CE Board #	CEB Channel #	ASIC	ASIC Channel
U198	2	1	6	79	5	15
Z158	2	4	6	78	5	14
Z159	2	7	6	77	5	13
U200	2	10	6	76	5	12
Z162	2	13	6	75	5	11
Z163	2	16	6	74	5	10
U202	2	19	6	73	5	9
Z166	2	22	6	72	5	8
U203	2	25	6	71	5	7
Z169	2	28	6	70	5	6
Z170	2	31	6	69	5	5
U205	2	34	6	68	5	4
Z173	2	37	6	67	5	3
Z174	2	40	6	66	5	2
Z176	2	43	6	65	5	1
Z177	2	46	6	64	5	0
U208	2	49	6	95	6	15
Z180	2	52	6	94	6	14
Z181	2	55	6	93	6	13
U210	2	58	6	92	6	12
Z184	2	61	6	91	6	11
U211	2	64	6	90	6	10
Z187	2	67	6	89	6	9
Z188	2	70	6	88	6	8
U213	2	73	6	87	6	7
Z191	2	76	6	86	6	6
Z192	2	79	6	85	6	5
Z194	2	82	6	84	6	4
Z195	2	85	6	83	6	3
U216	2	88	6	82	6	2
Z198	2	91	6	81	6	1
Z199	2	94	6	80	6	0

Strip Number	Connector #	Connector Pin #	CE Board #	CEB Channel #	ASIC	ASIC Channel
Z156	2	3	6	96	7	0
Z157	2	6	6	97	7	1
U199	2	9	6	98	7	2
Z160	2	12	6	99	7	3
Z161	2	15	6	100	7	4
U201	2	18	6	101	7	5
Z164	2	21	6	102	7	6
Z165	2	24	6	103	7	7
Z167	2	27	6	104	7	8
Z168	2	30	6	105	7	9
U204	2	33	6	106	7	10
Z171	2	36	6	107	7	11
Z172	2	39	6	108	7	12
U206	2	42	6	109	7	13
Z175	2	45	6	110	7	14
U207	2	48	6	111	7	15
Z178	2	51	6	112	8	0
Z179	2	54	6	113	8	1
U209	2	57	6	114	8	2
Z182	2	60	6	115	8	3
Z183	2	63	6	116	8	4
Z185	2	66	6	117	8	5
Z186	2	69	6	118	8	6
U212	2	72	6	119	8	7
Z189	2	75	6	120	8	8
Z190	2	78	6	121	8	9
U214	2	81	6	122	8	10
Z193	2	84	6	123	8	11
U215	2	87	6	124	8	12
Z196	2	90	6	125	8	13
Z197	2	93	6	126	8	14
U217	2	96	6	127	8	15

Reference AB1 AB2 AB3 **AB4** AB5 AB6 AB7 AB8

- TriggerPrimitive struct has members det_id and channel.
  - Channel map for the algorithm needs to map {det_id,channel} -> {z channel number}.
- Not sure exactly what det_id and channel get set to (TriggerPrimitiveMaker?).
- If I get this, think I have everything I need to make the channel map for the algorithm to read.

Summary

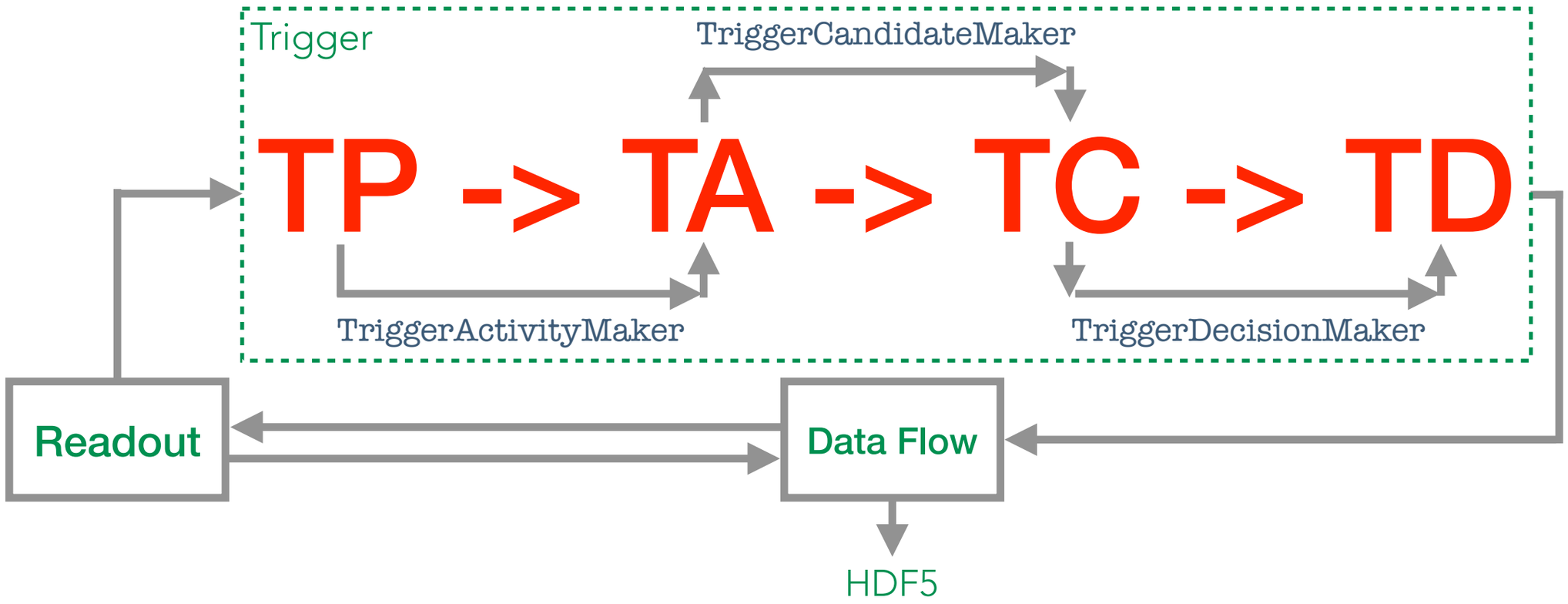
- A simple algorithm based on counting the number of channels with hits in a given window of time has been designed and tested in the TP -> TD chain.
- Python scripts exist to generate the configurations for it to be run with nanorc.
- Next steps are to focus on getting the channel map into the algorithm so can look at channel adjacency (within and across detector elements).

Back-up

Queen Mary
University of London

DEEP UNDERGROUND
NEUTRINO EXPERIMENT

A Reminder




```
namespace triggeralgs {

class TriggerActivityMaker
{
public:
  virtual void operator()(const TriggerPrimitive& input_tp, std::vector<TriggerActivity>& output_ta) = 0;
```

Existing example:

```
namespace triggeralgs {
class TriggerActivityMakerSupernova : public TriggerActivityMaker
{

void
TriggerActivityMakerSupernova::operator()(const TriggerPrimitive& input_tp, std::vector<TriggerActivity>& output_ta)
{
```

Running the Algorithm

Algorithm still in development. Use at own risk.

- Ben L developed an application which runs a full data selection chain:
  - `TriggerPrimitiveMaker` reads and constructs TPs from a file.
  - Uses `FakeDataFlow` to emulate the response of data flow.
  - Triggering algorithm of choice run in the middle.

- Can use `faketp_chain` confgen to generate the configuration to run this application:

```
python -m trigger.faketp_chain -a TriggerActivityMakerHorizontalMuonPlugin -A “dict(n_channel_threshold=1000000)” -c TriggerCandidateMakerHorizontalMuonPlugin -f tps_link_11.txt faketp_chain_HorizontalMuon
```

- Then to run:

```
nanorc faketp_chain_HorizontalMuon boot init conf start 1 resume wait 60 stop
```