Weekly INTT meeting 2025/11/14

Updates on hit carryover

Ryotaro Koike
Kyoto University

Topic

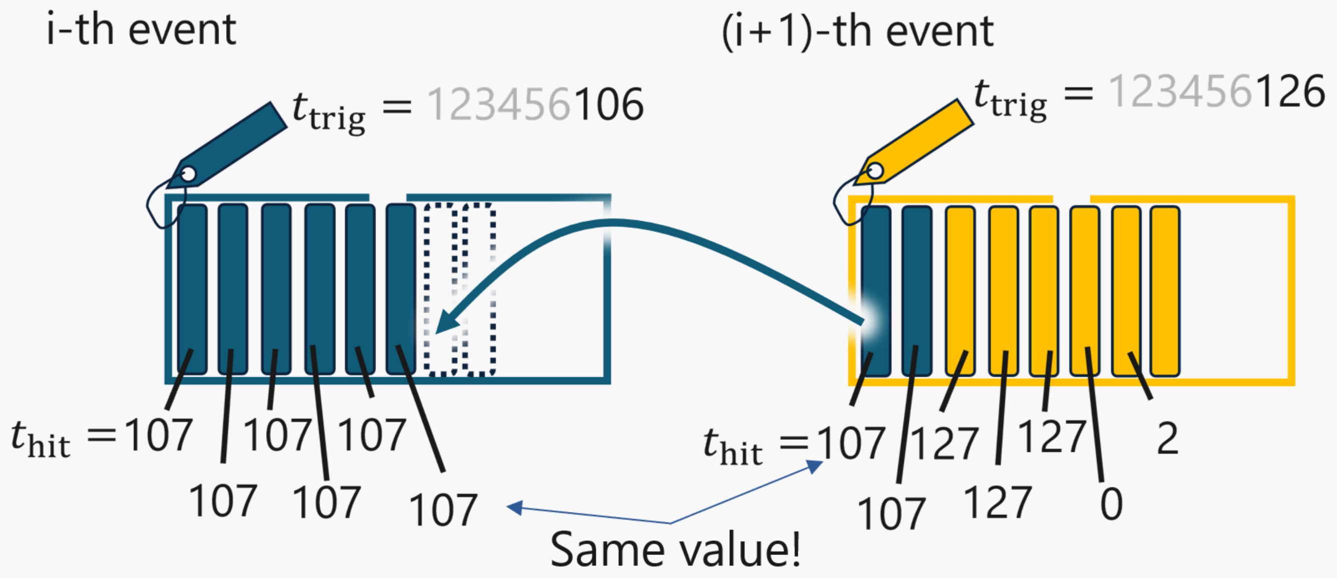
- Discussion with Raul
- Updates on the offline analysis
- Recovery plan at the decode process

Recap

Recovery of hit carryover confirmed to work basically. There is still a space for optimization. The procedure can be summarized as below.

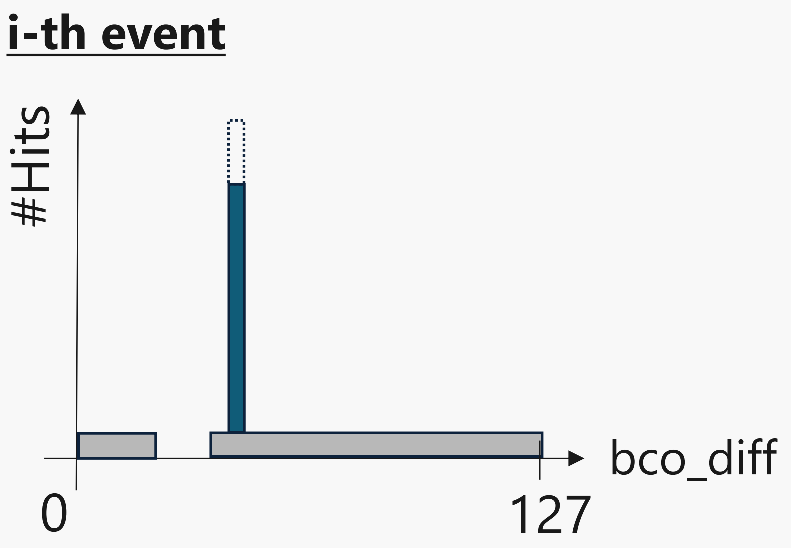
Run dependent value

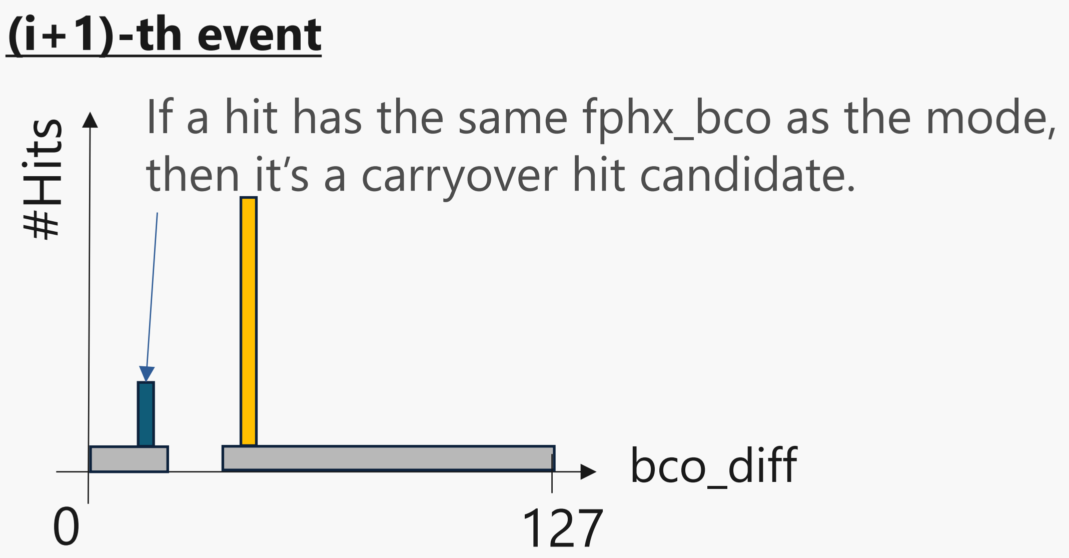
1. **Identify a fphx_bco value carried-over hits can have.**
   - Method 1: find the mode of fhpx_bco distribution in the previous event.
   - Method 2: calculate the value from event interval and a trigger timing bco_diff value.
2. **Push carryover hit candidates back following some criteria.**
   - Criteria ver. 1: requires those hits appear in the very beginning of a hit list.
   - Criteria ver. 2: no further requirement other than the fphx_bco value.


i-th event
(i+1)-th event
t_{trig} = 123456106
t_{trig} = 123456126
t_{hit} =107 107 107 107 107 107
107 107 107
t_{hit} =107 107 127 127 127 127 0 2
Same value!

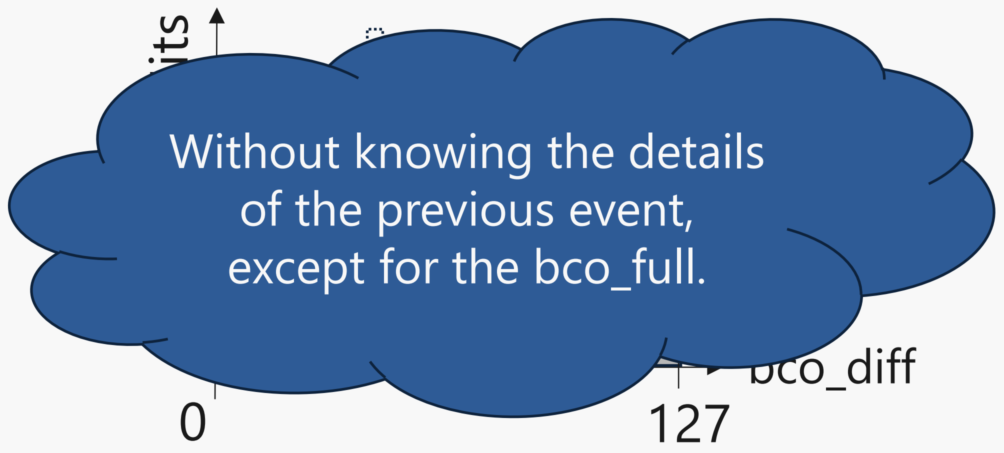
Step 1: Identification of candidate fphx_bco value

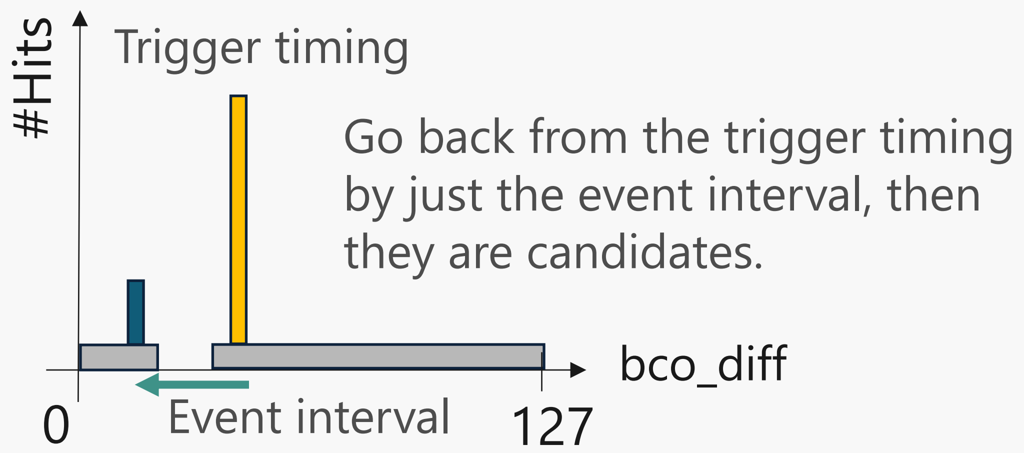
- **Method 1: find the mode of fhpx_bco distribution in the previous event.**




(i+1)-th event
#Hits
If a hit has the same fphx_bco as the mode, then it's a carryover hit candidate.
bco_diff
0
127

- **Method 2: calculate from event interval and a trigger timing bco_diff value.**

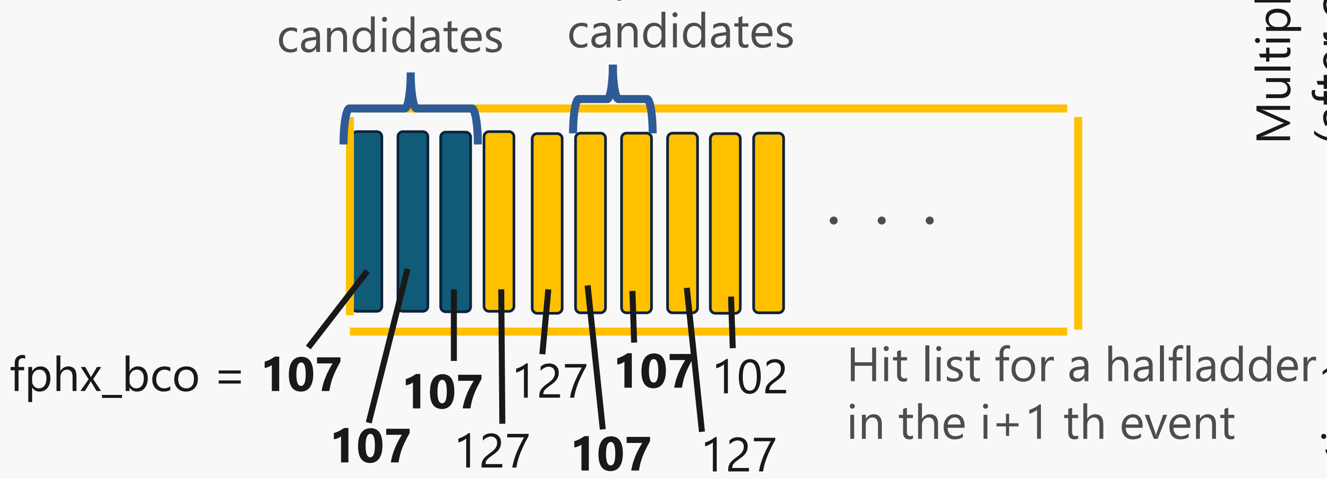

Without knowing the details of the previous event, except for the bco_full.
bco_diff
0
127

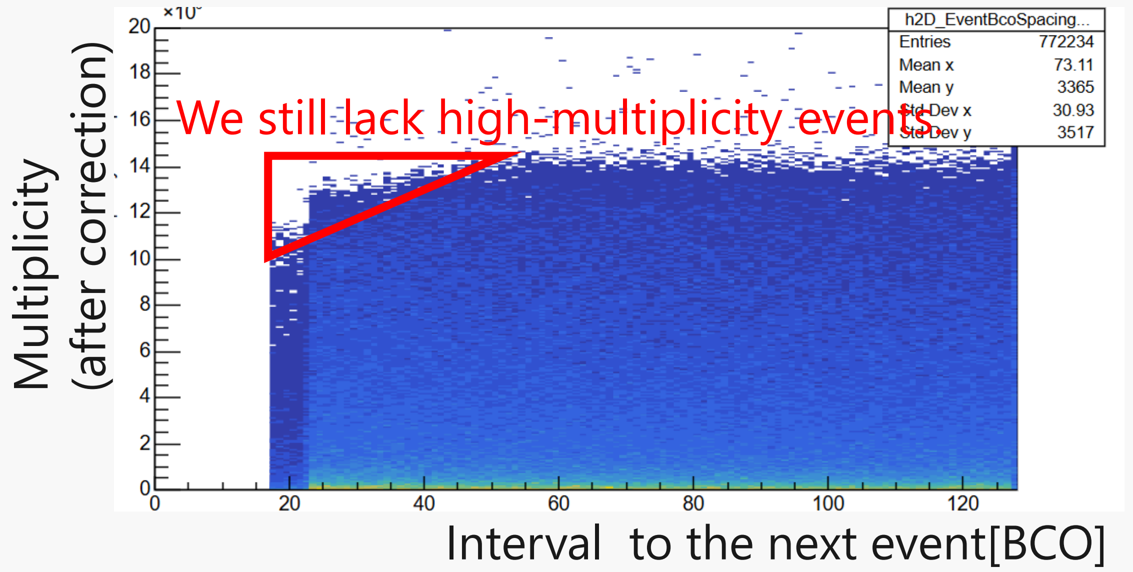

#Hits
Trigger timing
Go back from the trigger timing by just the event interval, then they are candidates.
bco_diff
0
Event interval
127

Step 2: Push candidates back

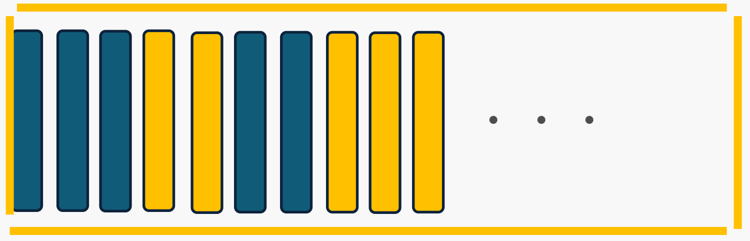
- **Criteria ver. 1:**
  - Push back only a chank of consecutive candidates that lies in the very first part of the hit list.

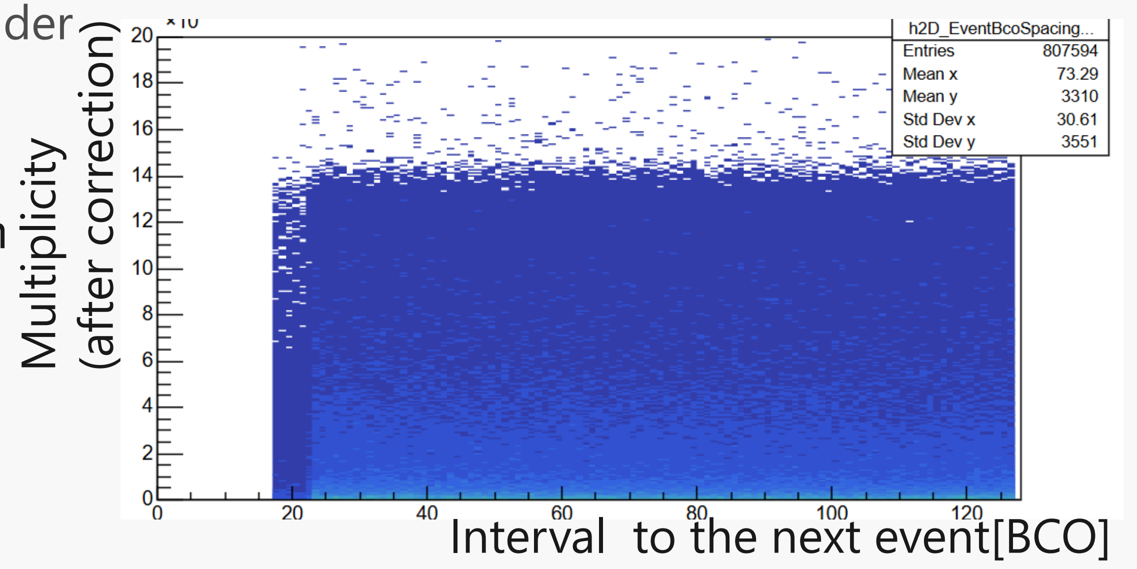
Suppose that we found hits with fphx_bco=107 are candidates.





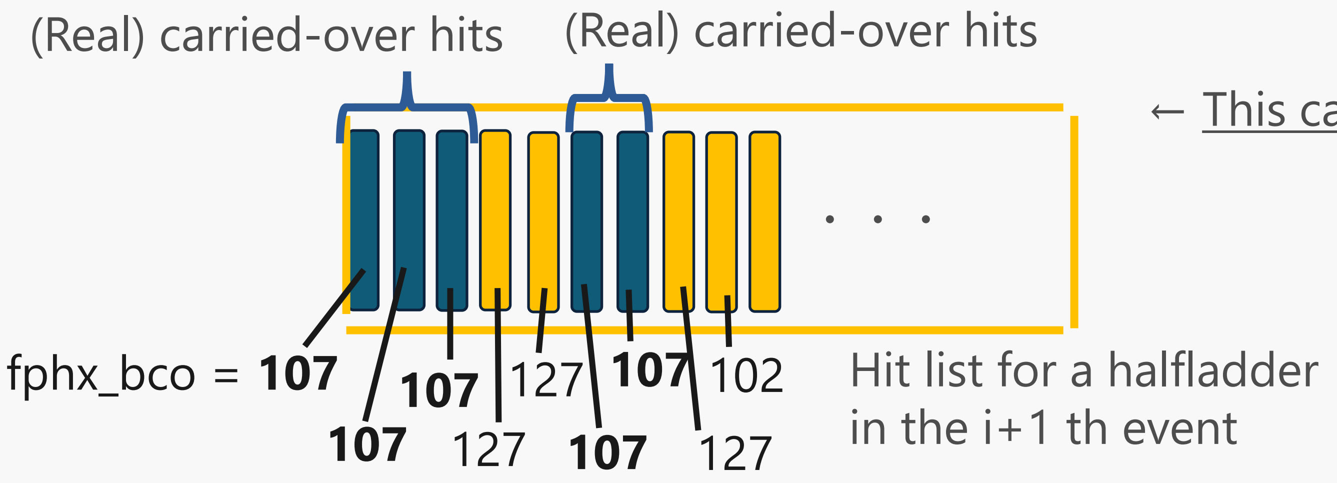
- **Criteria ver. 2:**
  - Push back all candidates regardless of their position in the hit list.



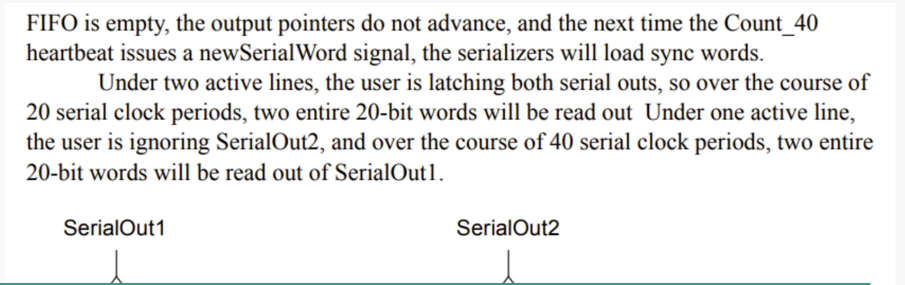


Discussion with Raul and new conclusions

- **We should not assume any order in a hit list.**
  - Carried-over hits don't necessarily come at the beginning of a hit list.
  - (For me,) Raul seemed assuming an order at least on a per-halfladder basis at some low-level structure, but his point was it's safe not to assume any order until we have a proof of it.



← This can happen.


FIFO is empty, the output pointers do not advance, and the next time the Count_40 heartbeat issues a newSerialWord signal, the serializers will load sync words.

Under two active lines, the user is latching both serial outs, so over the course of 20 serial clock periods, two entire 20-bit words will be read out Under one active line, the user is ignoring SerialOut2, and over the course of 40 serial clock periods, two entire 20-bit words will be read out of SerialOut1.

- ~~**A FPHX chip sends 2 hits/BCO when it's operated with the 2-serial-lines mode.**~~
  - ~~In a 1-serial-line mode, its throughput will be ½ of that of 2-serial-lines mode~~

蜂谷さんのミーティング内でのコメント：
ROC が 2 本のアウトプットの OR を取る挙動になっているので、実質的に 1 hit/BCO なはず。
そうでなければデータが壊れてまともなプロットが見えているはずがない。

Raul pointed out this explanation as an evidence. (FPHX document p.57)

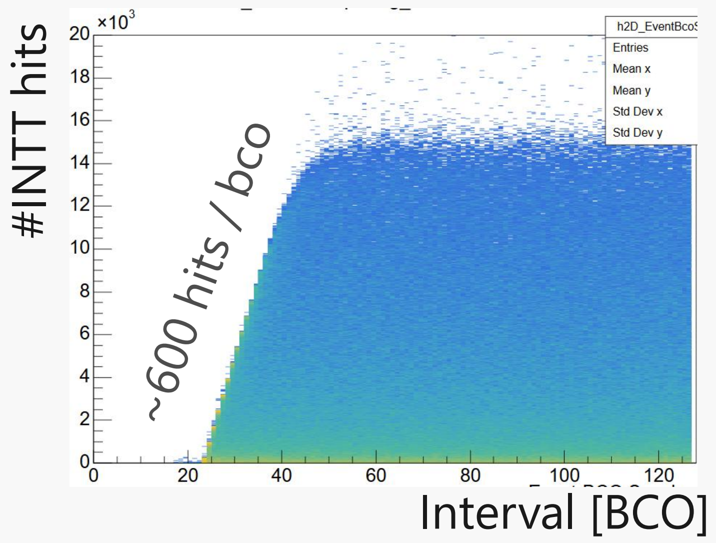
Updates on the offline analysis

Plots that I have shown as a per-event, per-halfladder, or per-chip basis result was not produced in the same analysis condition.
For an apple-to-apple comparison, I reproduced the plots on a common analysis condition.

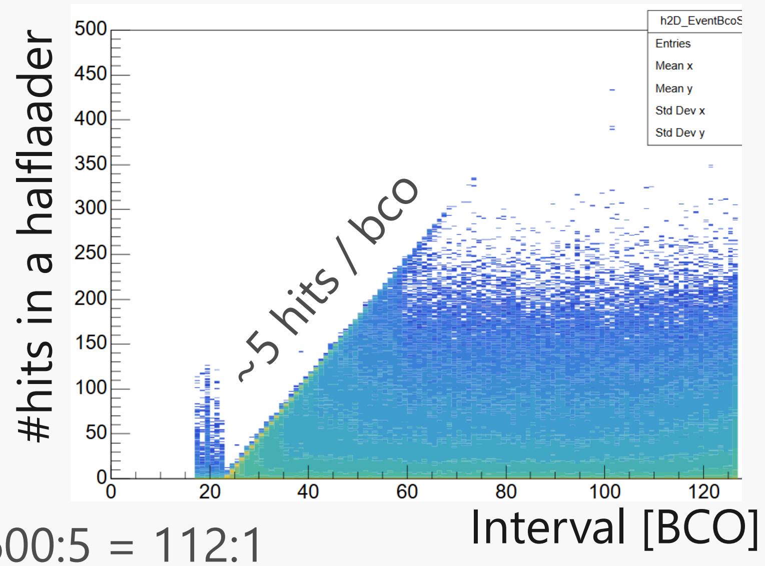
- **Condition:**
  - In step 1: method 2 for all interval region (0 < interval < 127).
  - In step 2: criteria ver. 2 (no special selection)
  - #hit = the number of hits in trigger timing ±1

- Only the events with carryover drawn.
- Clone hit not removed.
- Hot channel not removed.
- Skip an event if previous/next event is empty (container->get_nhits() == 0).

On a per-event basis

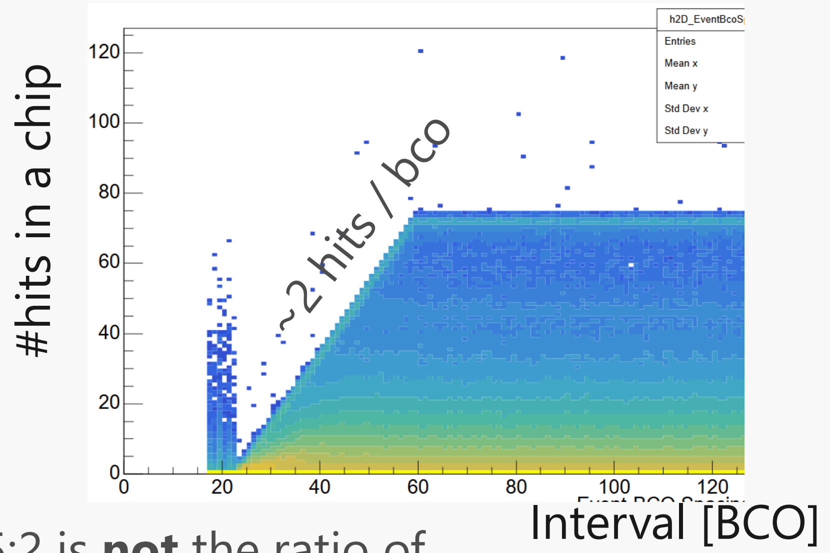


On a per-halfladder basis



600:5 = 112:1
(112 = #halfladder)

On a per-chip basis

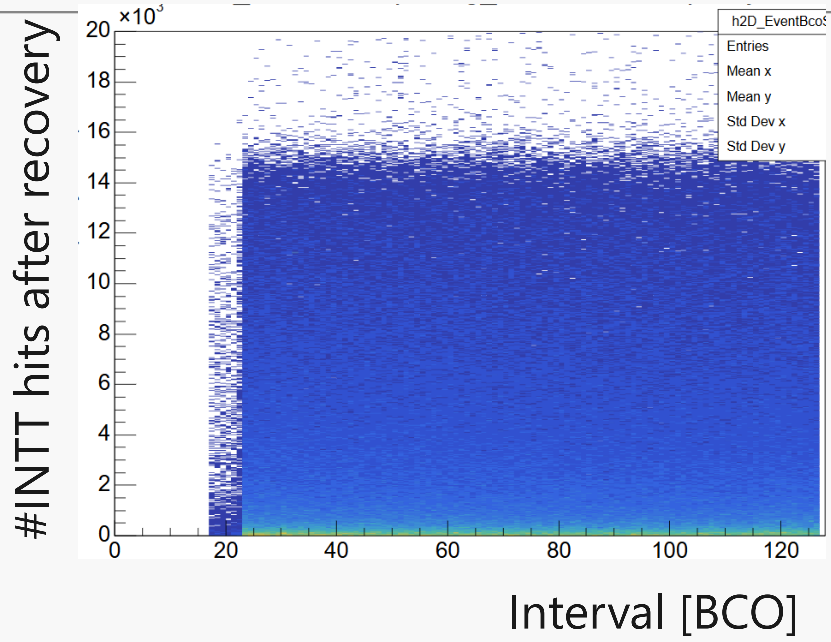


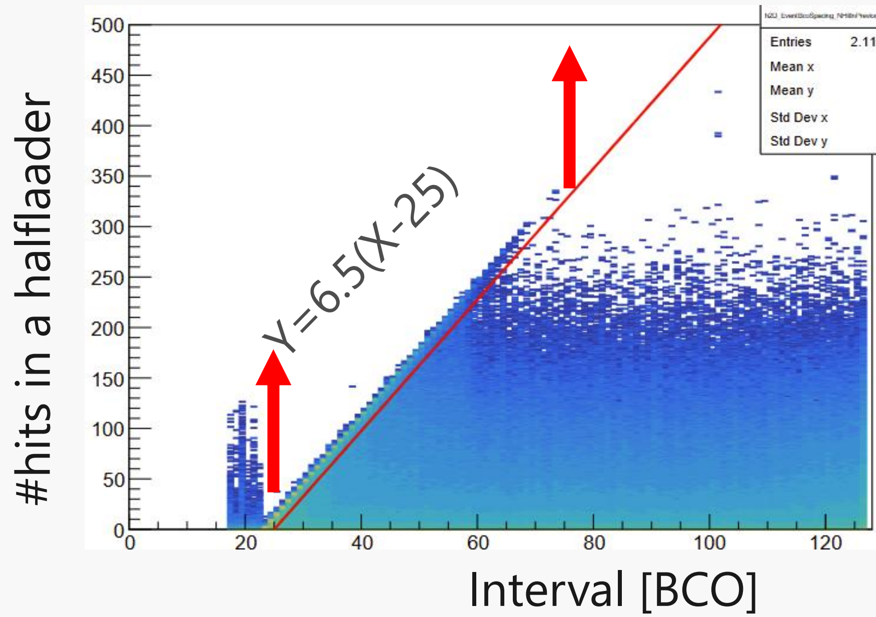
5:2 is **not** the ratio of #halfladder vs #chip

Updates on the offline analysis

- **Multiplicity after the recovery as a function of event interval.**
- **Less entries in short interval region $(17 \leq \text{interval} \leq 22)$.**
  - Step 1 method 2 is applied for all interval region, so it's not an artifact by a method difference in step 1.

  - Triggered events were little from the begging? → I will check the distribution of interval.
  - Missing carryover which leaves the previous event completely empty? → I will check the possibility.
  - Enhancement of entries in large interval region? → I will try to select the case in which previous event hit the upper limit.





Recovery plan at the decode process

The offline recovery "recovers" the issue only within the module's scope itself. We need to implement the recovery function in the decoder eventually to attain profit for various analysis.

- **Important modules in decode process**
  - InttProduction/Fun4All_Intt_Combiner.C
    Fun4All macro to run decoder.
  - offline/framework/fun4allraw/SingleStreamingInput.cc
    StreamingInputManager. Prepares an output DST file and adds InttRawHits to the file.
  - offline/framework/fun4allraw/SingleInttPoolInput.cc
    A class that plays a core role. Loops over every RCDAQ packet in input files, creates a INTTRawHit from decoded values, and pass it to StreamingInputManager.
  - offline/framework/fun4allraw/intt_pool.c
    Helper class which pools a RCDAQ packet and decode hits in the packet.

- **How INTT decoder works**
  - See this →
    https://indico.bnl.gov/event/30566/contributions/116709/attachments/66243/113714/InttDecoderExplained.pdf

Recovery plan at the decode process

- **Recovery of hit carryover at the decode process is possible**

- **What should be done is**
  - interrupt when SingleInttPoolInput tries to pass an InttRawHit to StreamingInputManager
  - check if it's carried-over hit
  - If so, correct the bco_full of the hit and pass the hit with the previous gtm_bco.

- **We would be able implement it simply, if following method is employed;**
  - In step 1: method 2 for all interval
  - In step 2: criteria ver. 2

No need to store an information of one "event".
Method 2 is a purely hit-by-hit determination method.

We don't have to care the position of the hit in the hit list.

Maybe we can improve the recovery algorithm by employing a new step (step 0) which determine whether a carryover is ocuuring considering the number of halfladder hits recorded in previous event.
i.e. check of fphx_bco will be done only when the number of halfladder hits reached the upper limit.

- **More careful inspection is needed in implementation.**
  - There are some code block that I don't understand what it's doing.
  - I don't fully foresee the effect of the new code that I am trying to put into, to the other part of the code.

Backup

FIFO is empty, the output pointers do not advance, and the next time the Count_40 heartbeat issues a newSerialWord signal, the serializers will load sync words.

Under two active lines, the user is latching both serial outs, so over the course of 20 serial clock periods, two entire 20-bit words will be read out Under one active line, the user is ignoring SerialOut2, and over the course of 40 serial clock periods, two entire 20-bit words will be read out of SerialOut1.

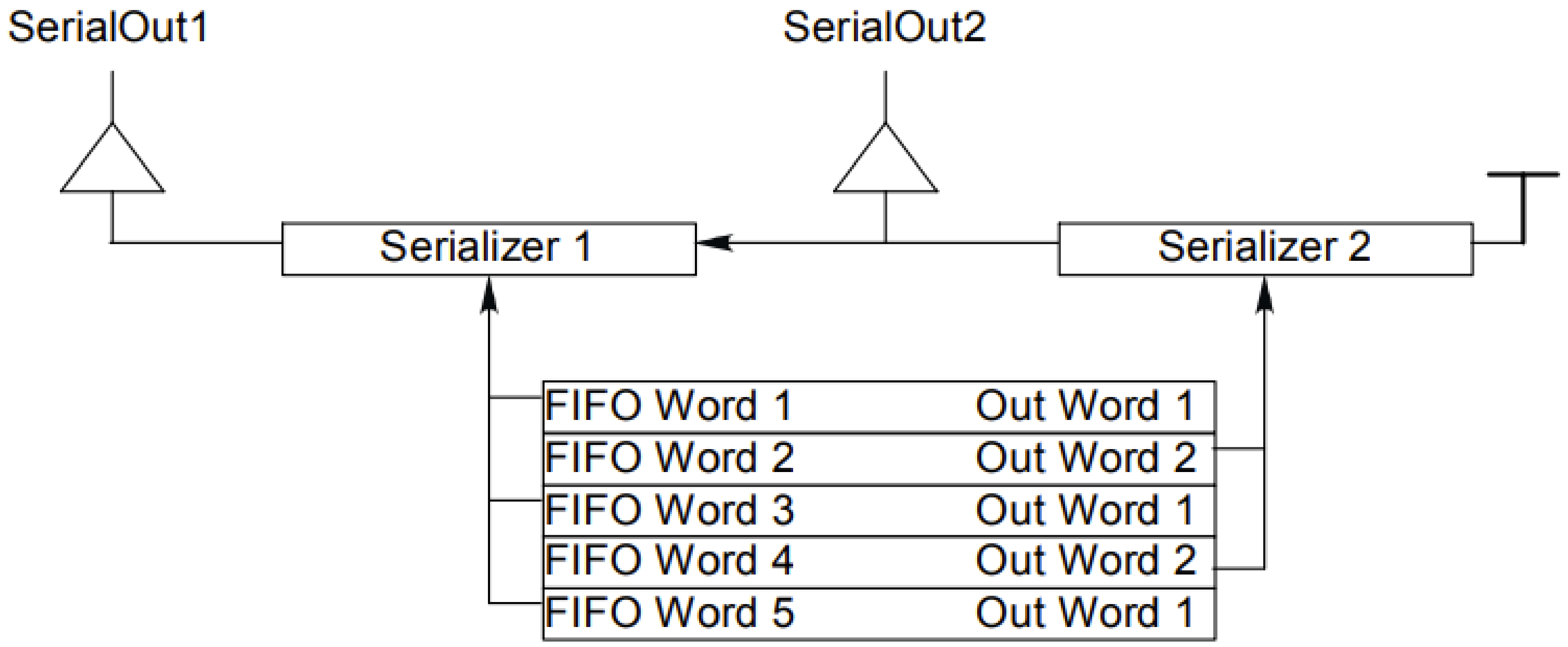
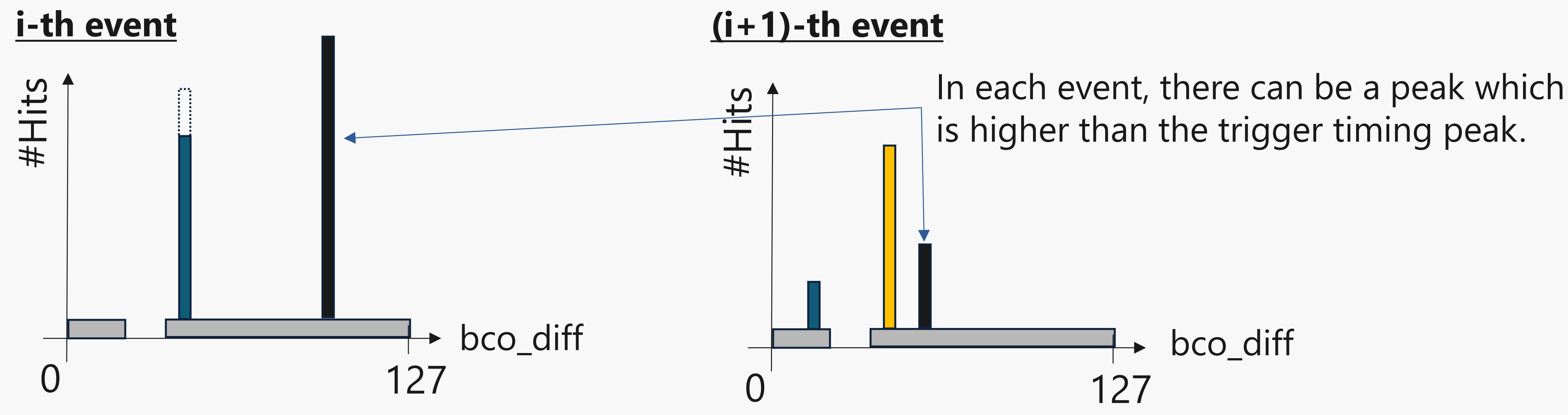


Figure 16 - The Serializers

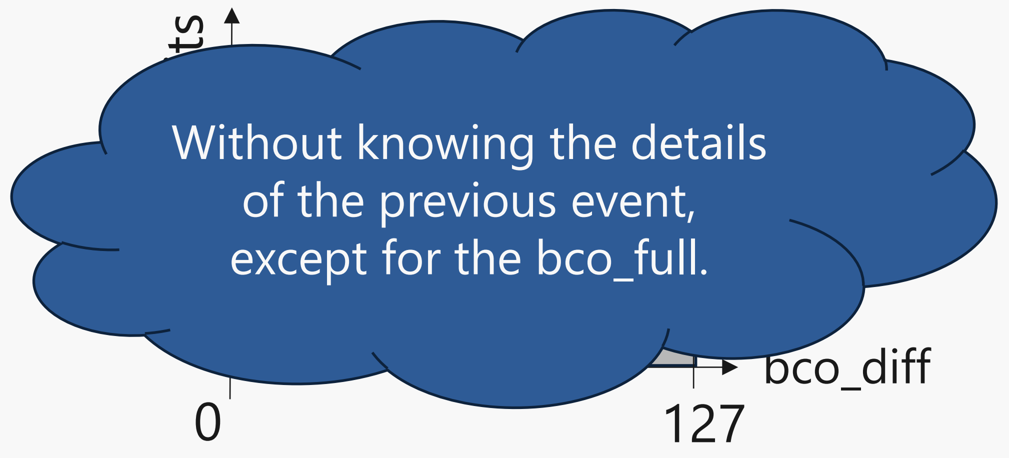
Step 1: Identification of fphx_bco value

- Method 1: find the mode of fhpx_bco distribution in the previous event.

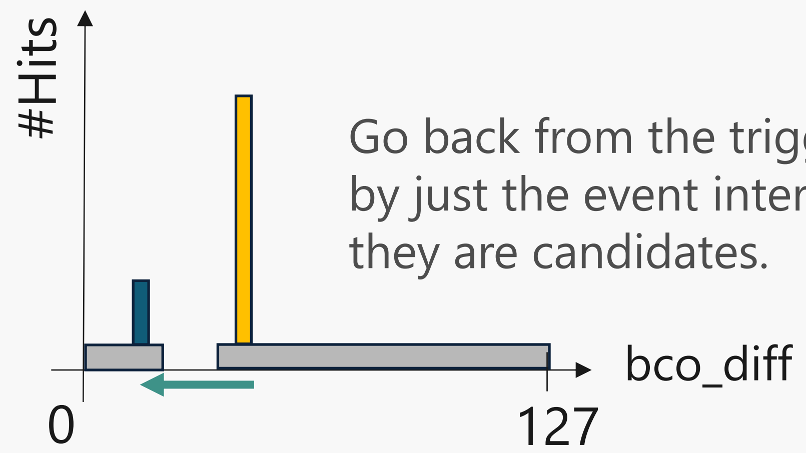


In each event, there can be a peak which is higher than the trigger timing peak.

- Method 2: calculate from event interval and a trigger timing bco_diff value.



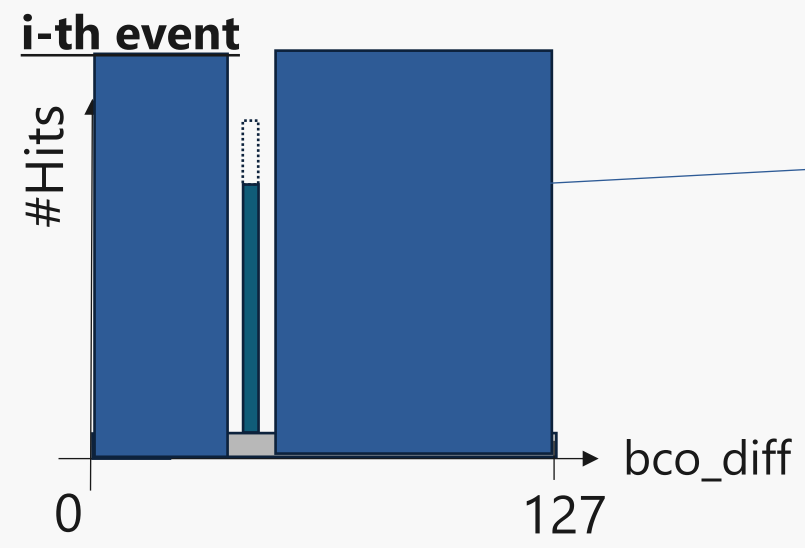
Without knowing the details of the previous event, except for the bco_full.

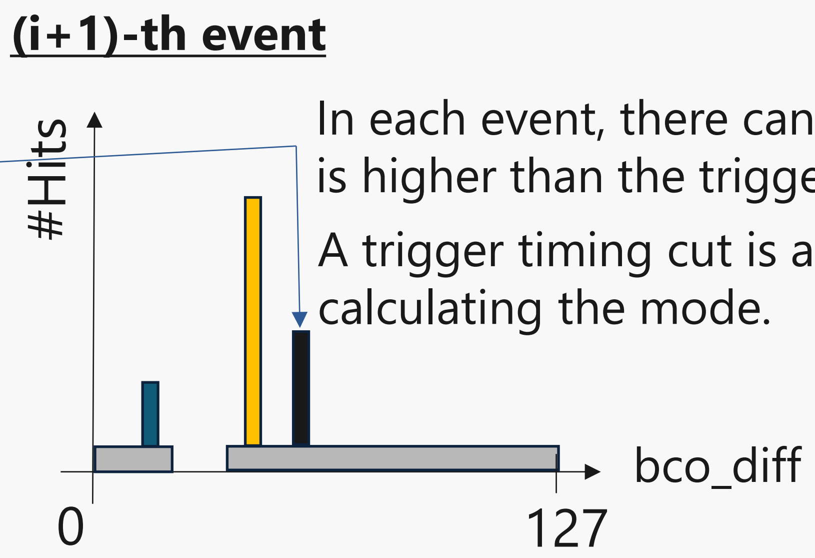


Go back from the trigger timing by just the event interval, then they are candidates.

Step 1: Identification of fphx_bco value

- Method 1: find the mode of fhpx_bco distribution in the previous event.

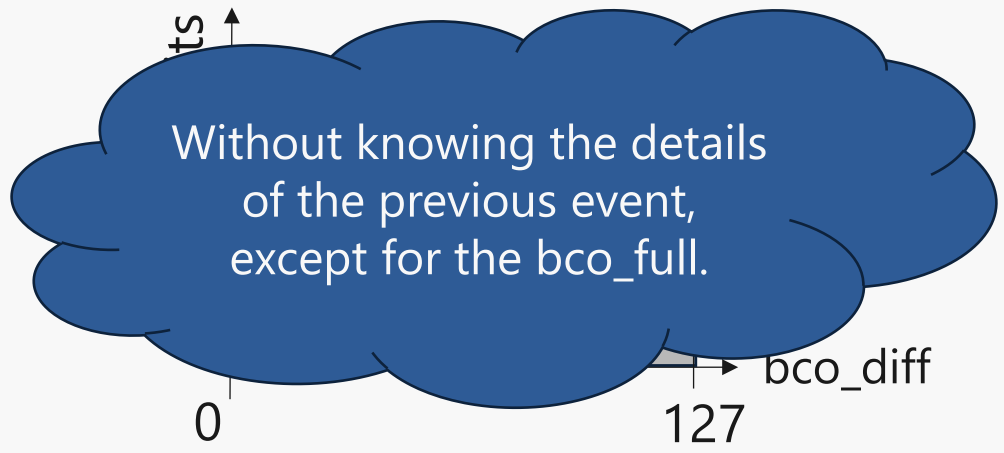




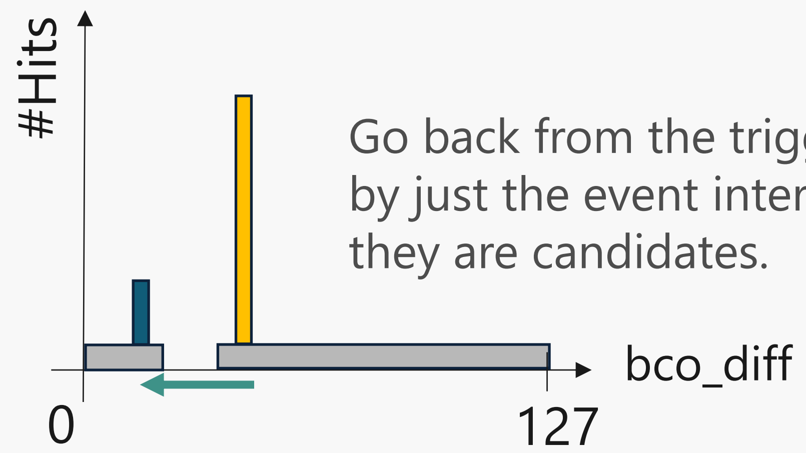
In each event, there can be a peak which is higher than the trigger timing peak.

A trigger timing cut is applied before calculating the mode.

- Method 2: calculate from event interval and a trigger timing bco_diff value.



Without knowing the details of the previous event, except for the bco_full.



Go back from the trigger timing by just the event interval, then they are candidates.