UNIVERSITY OF LIVERPOOL

LEVERHULME TRUST

MUonE Tracker synchronisation

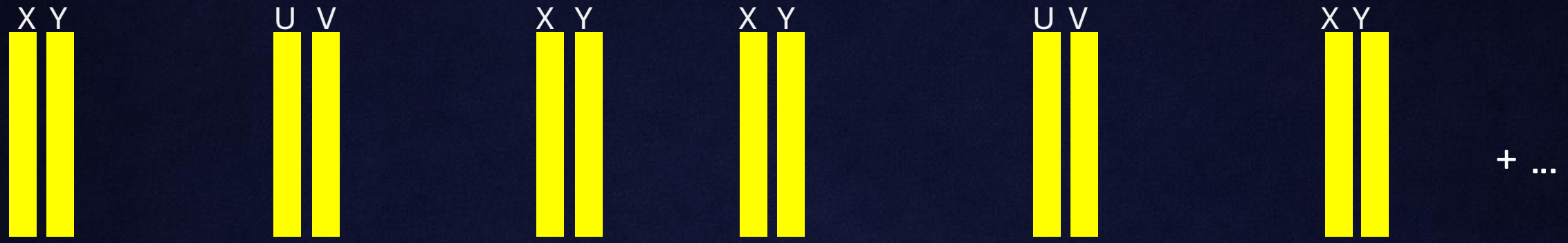
2025 TestRun

Clément Devanne

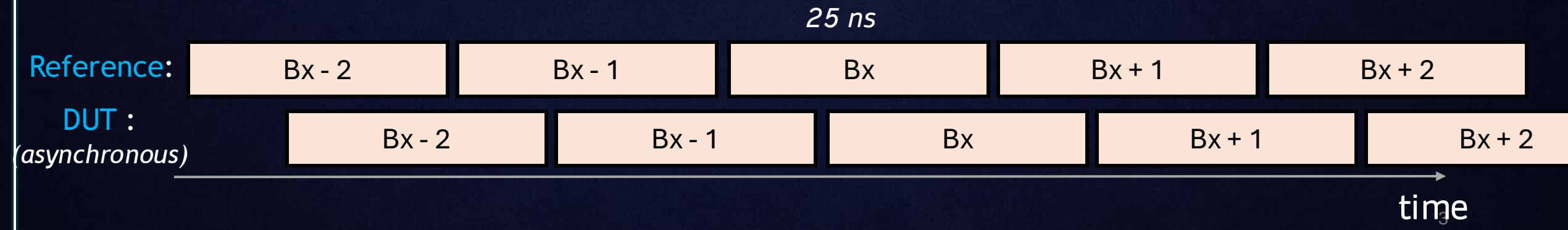
Synchronisation procedure

MUonE - setup

- **Simple view : bunch of 2S modules detector in series + ECAL**

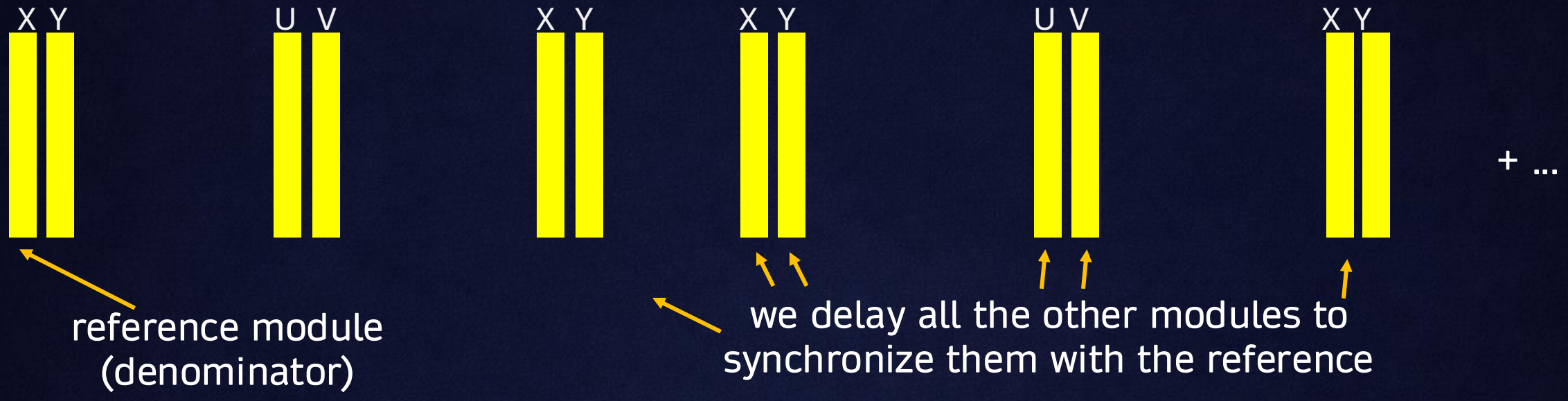


- **Each Module has its own internal clock cycle and send a `picture` (Bx) every 25 ns.**

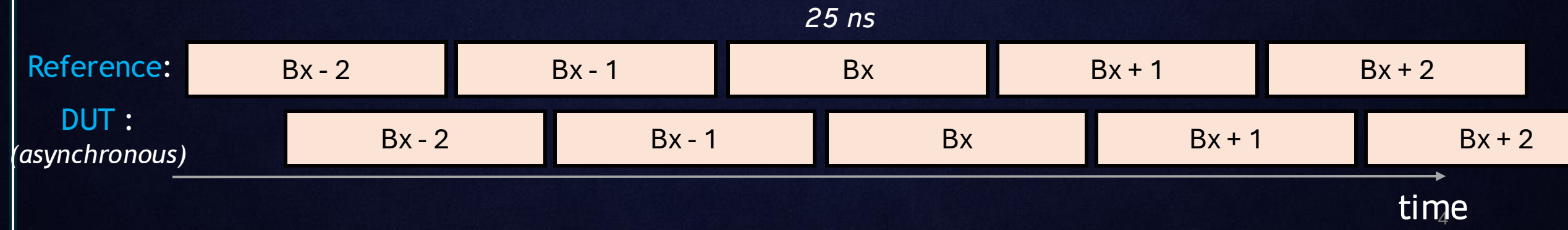


MUonE – setup and synchronisation

- **Simple view : bunch of 2S modules detector in series + ECAL**



- **Each Module has its own internal clock cycle and send a `picture` (Bx) every 25 ns.**

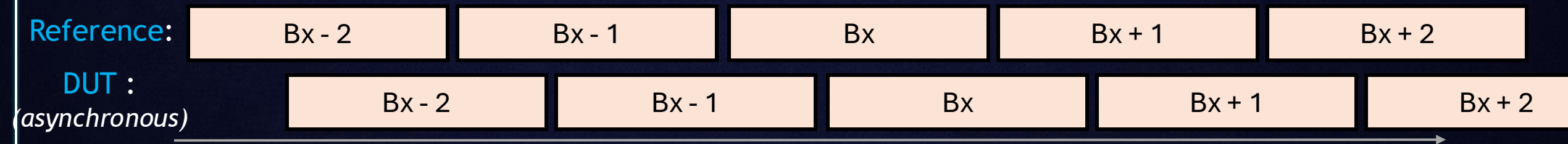


Synchronisation Procedure:

- Measured quantity: Ratio of events with at least one stubs in the tested module (DUT) when a stub is seen in the reference module *(sometime called efficiency on plots)*

$$\mathbf{R}(\text{module}_{\alpha}, \text{BX}_{\delta}) = \frac{N(\text{hits in module}_{0} \ \& \ \text{BX}_{0})}{N(\text{hits in module}_{\alpha} \ \& \ \text{BX}_{0+\delta})}$$

- Event Selection: Exactly 1 stubs in the reference module within 5 consecutive bunch crossing *(labelled as Bx-2, Bx-1, Bx, Bx+1, Bx+2)*



Note: For BMS modules we had to perform first a scan because default offset was larger than 2 Bx

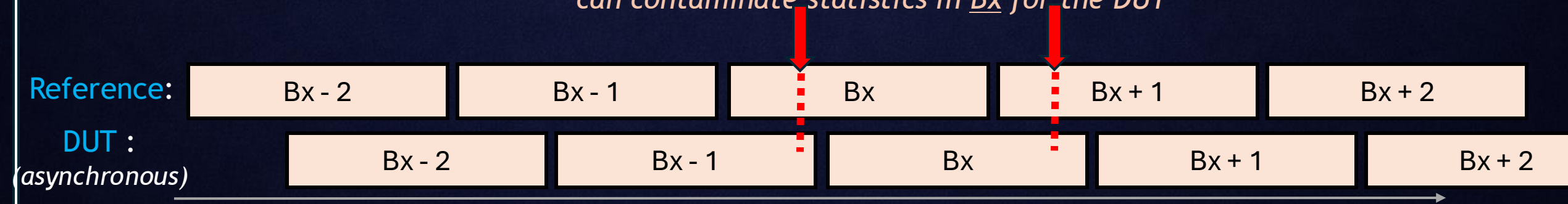
Synchronisation Procedure:

- Measured quantity: Ratio of events with at least one stubs in the tested module (DUT) when a stub is seen in the reference module *(sometime called efficiency on plots)*

$$\mathbf{R}(\text{module}_{\alpha}, \text{BX}_{\delta}) = \frac{N(\text{hits in module}_{0} \;\&\; \text{BX}_{0})}{N(\text{hits in module}_{\alpha} \;\&\; \text{BX}_{0+\delta})}$$

- Event Selection: Exactly 1 stubs in the reference module within 5 consecutive bunch crossing *(labelled as Bx-2, Bx-1, Bx, Bx+1, Bx+2)*

An event happening in Bx+1 in reference module can contaminate statistics in Bx for the DUT

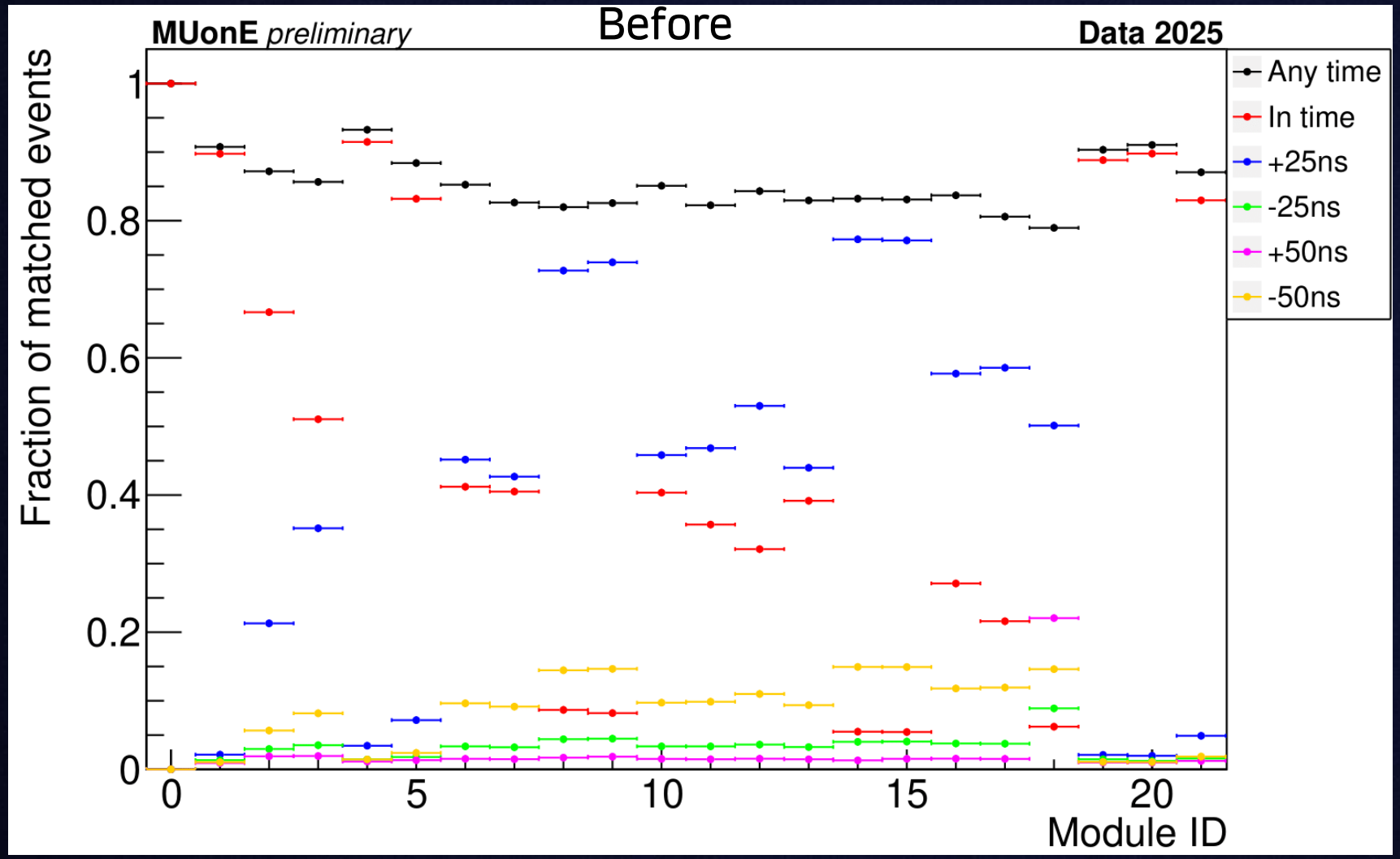


Note: For BMS modules we had to perform first a scan because default offset was larger than 2 Bx

Synchronisation Procedure – *fast command delay*:

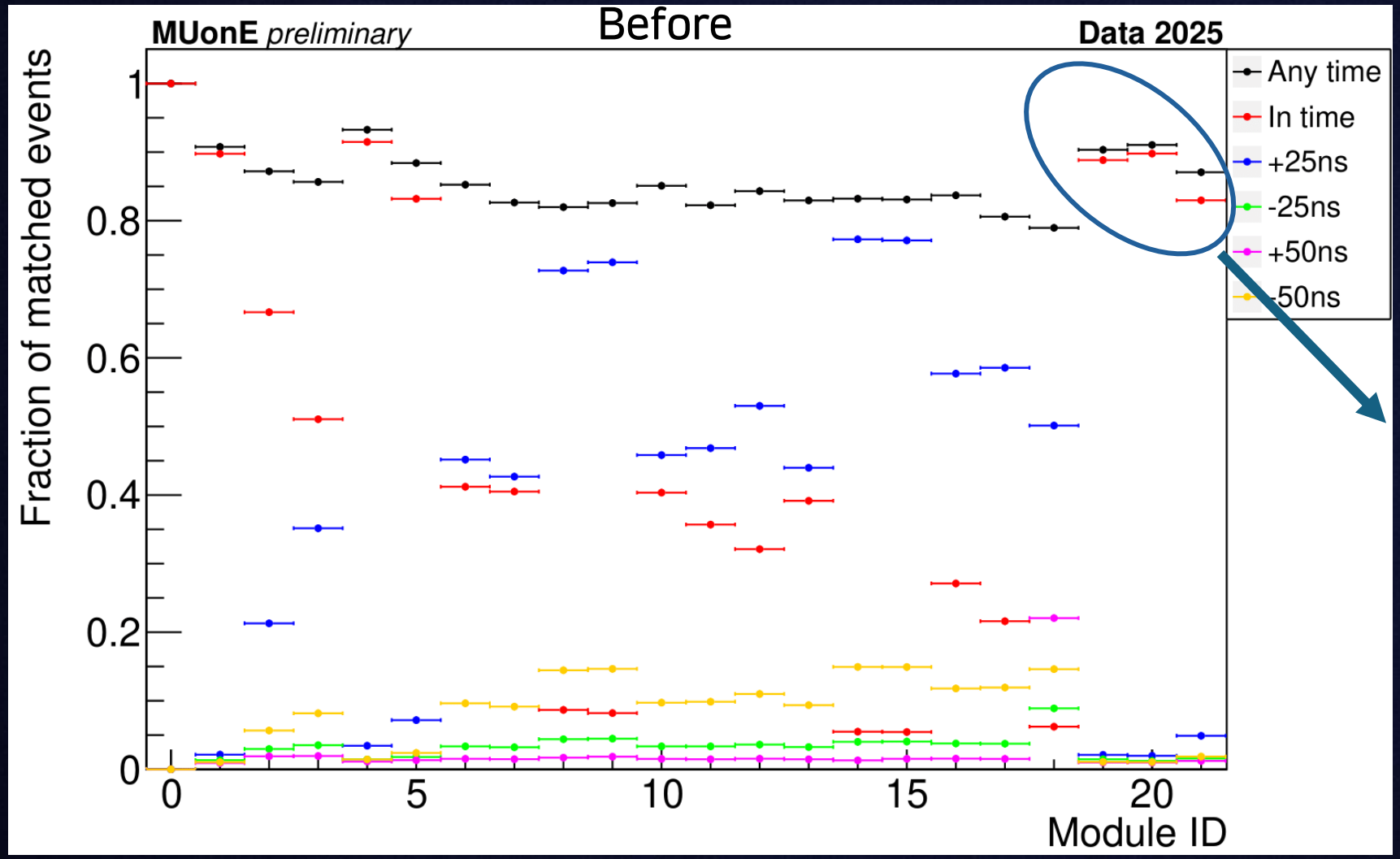
Synchronisation Procedure – *fast command delay*:

- Goal: set transmission delays in 25 ns units to maximise the ratio of matched events when the reference module and the DUT are in the same Bx.



Synchronisation Procedure – *fast command delay*:

- Goal: set transmission delays in 25 ns units to maximise the ratio of matched events when the reference module and the DUT are in the same Bx.

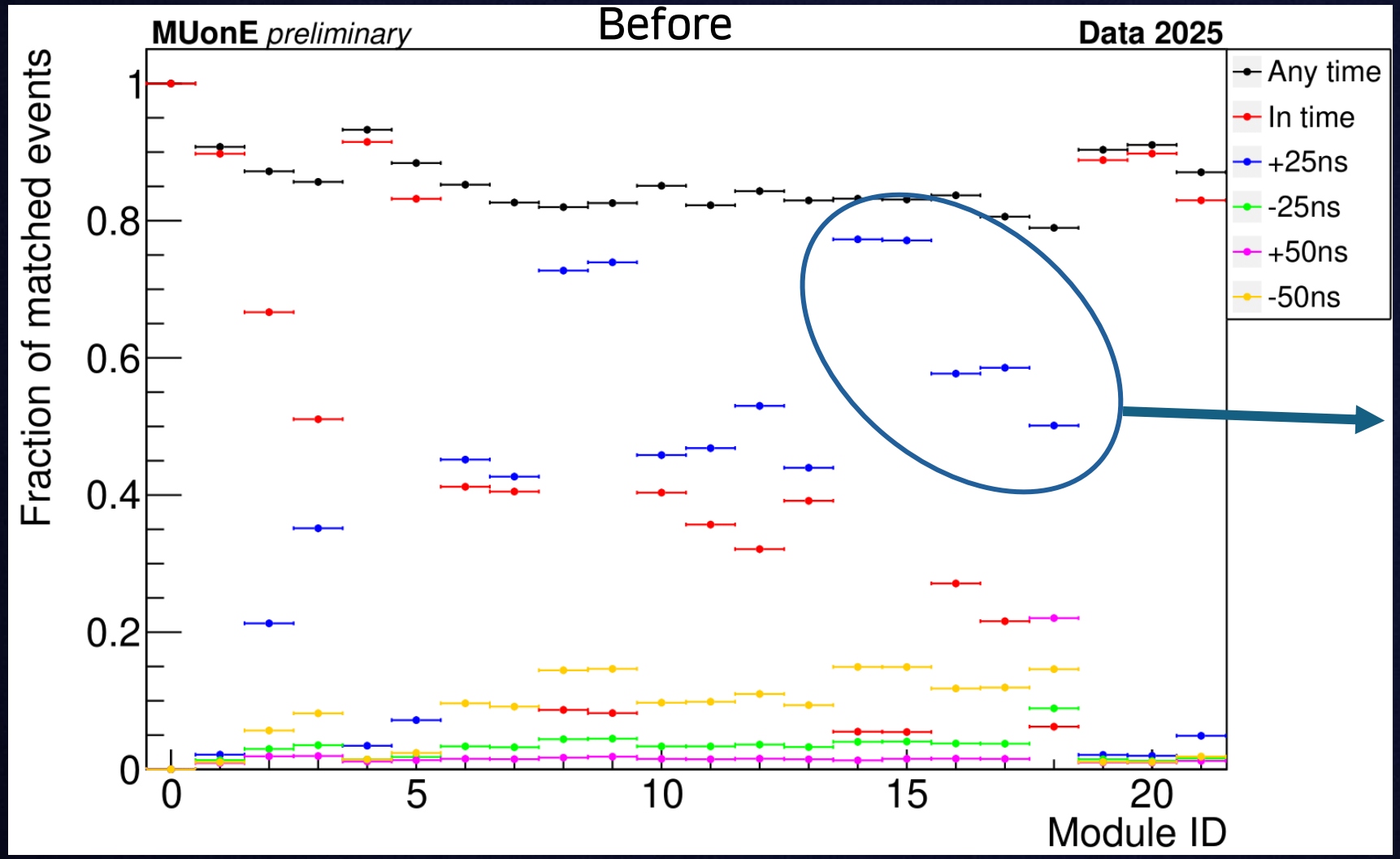


The fraction of matched events is already larger for coincidences in the same bx

No Delay needed

Synchronisation Procedure – *fast command delay*:

- Goal: set transmission delays in 25 ns units to maximise the ratio of matched events when the reference module and the DUT are in the same Bx.

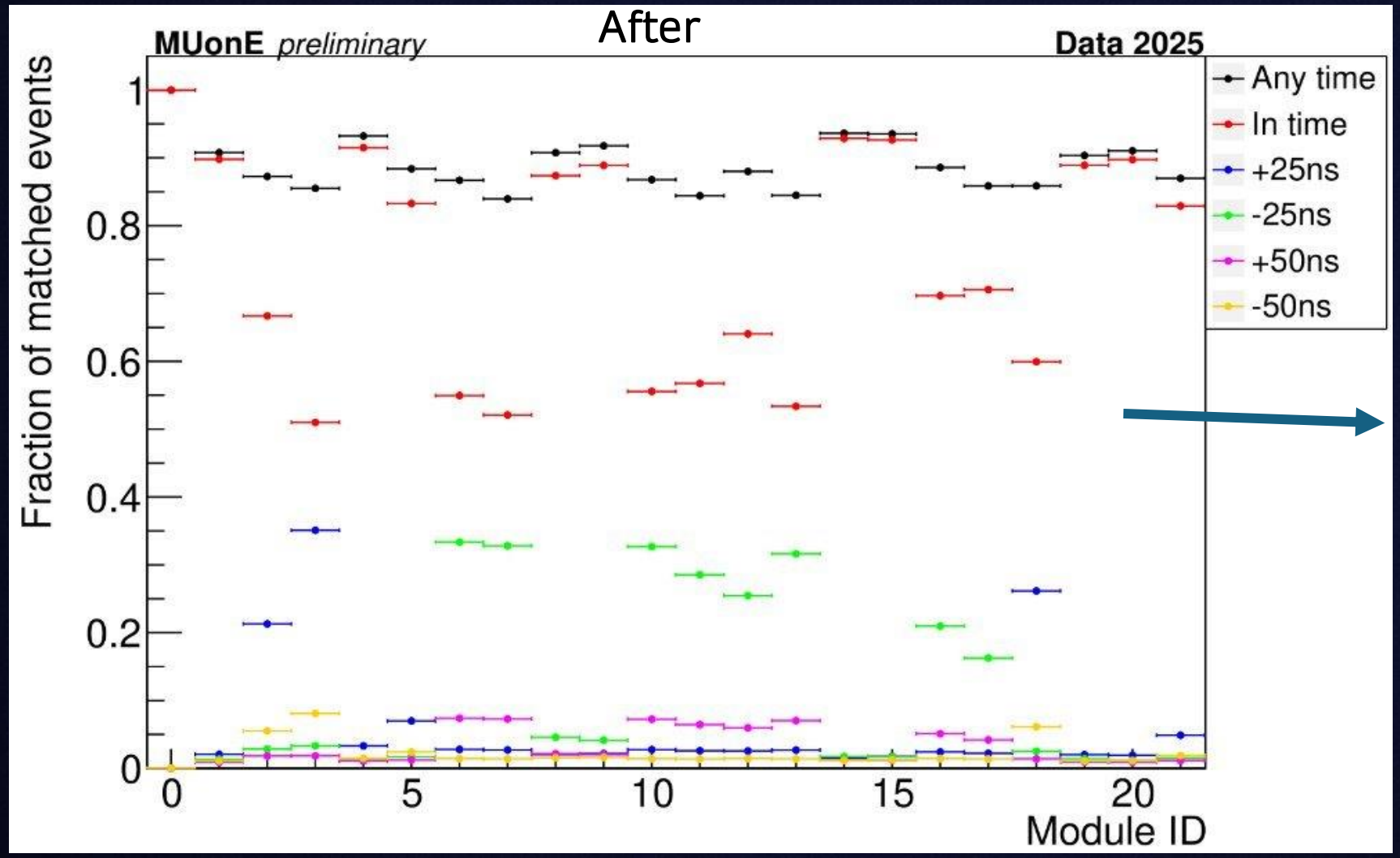


The fraction of matched events is larger between the reference module at a given Bx, and the DUT module in the following Bx+1

Delay the DUT by 1 Bx (25ns)

Synchronisation Procedure – *fast command delay*:

- After the fast command delay



The fraction of matched events is maximised for coincidences in the same bx for all modules after the fast delay command

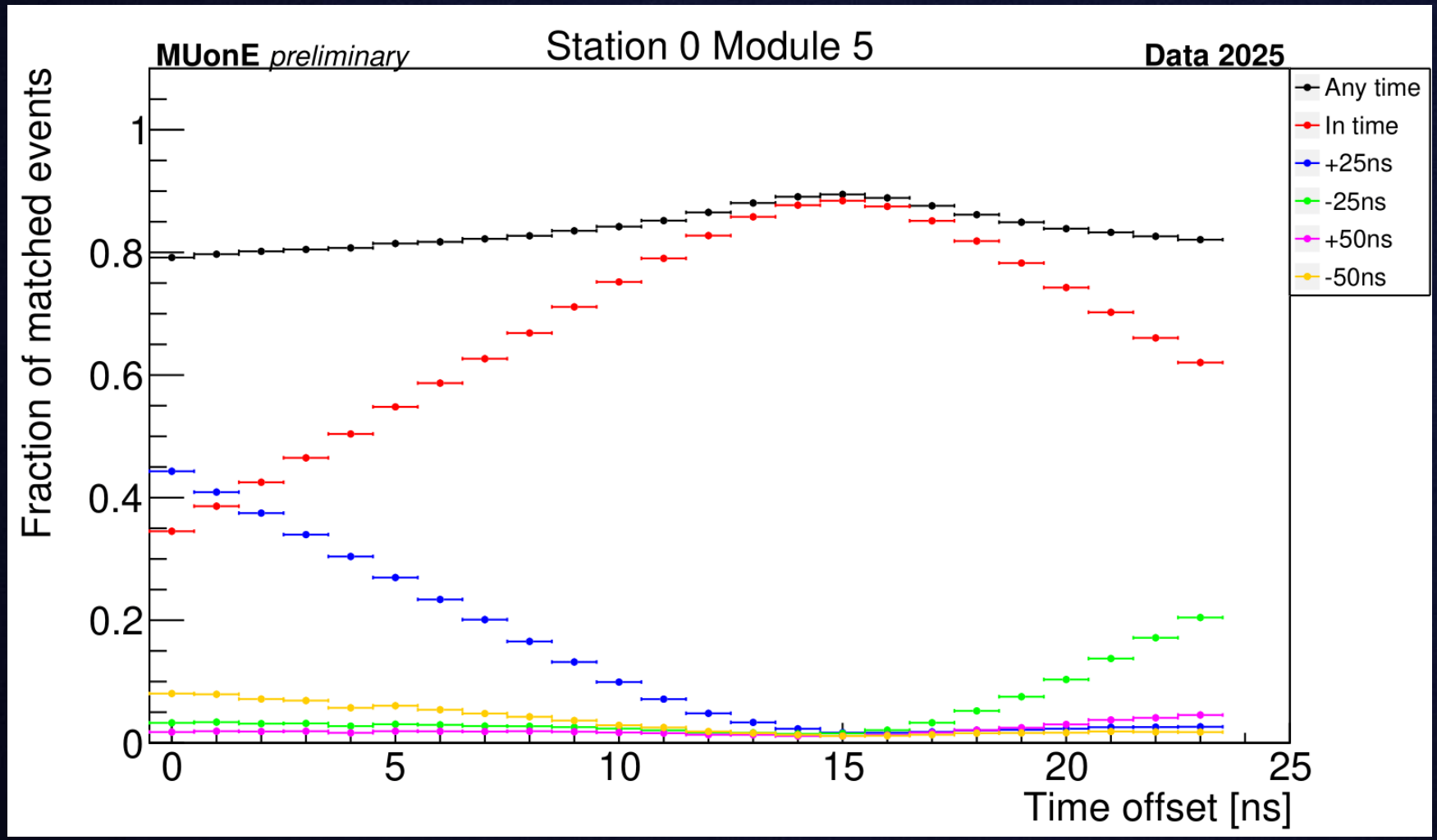
Ready for the DLL scan

Synchronisation Procedure – *DLL scan*:

Synchronisation Procedure – *DLL scan*:

Goal: maximise the ratio of matched events between the reference module and the DUT

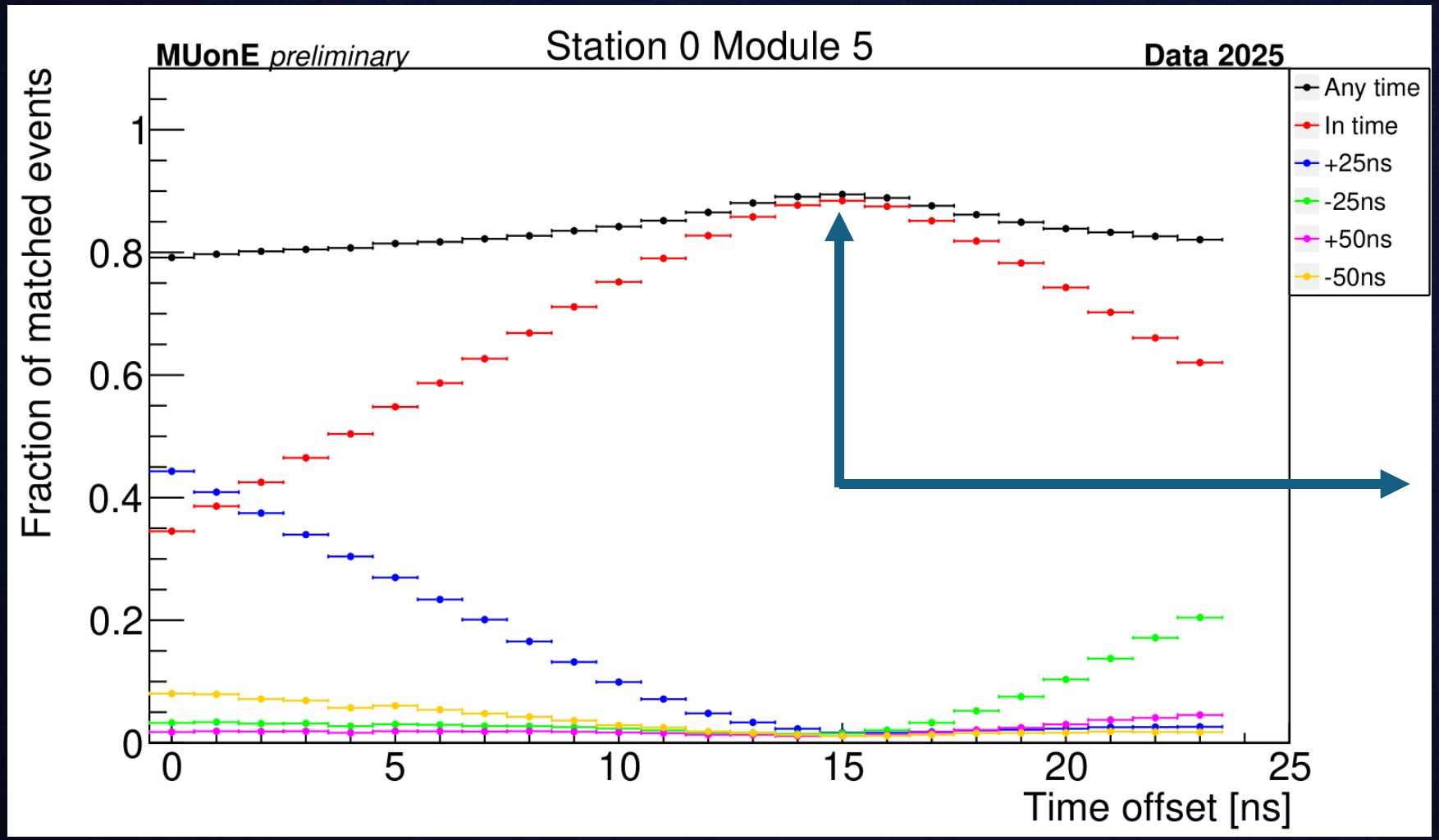
- Fix the internal delay (DLL) of the reference module to 12 ns (centre of the 25 ns window).
- Scan the DLL of all the other modules at the same time



Synchronisation Procedure – *DLL scan*:

Goal: maximise the ratio of matched events between the reference module and the DUT

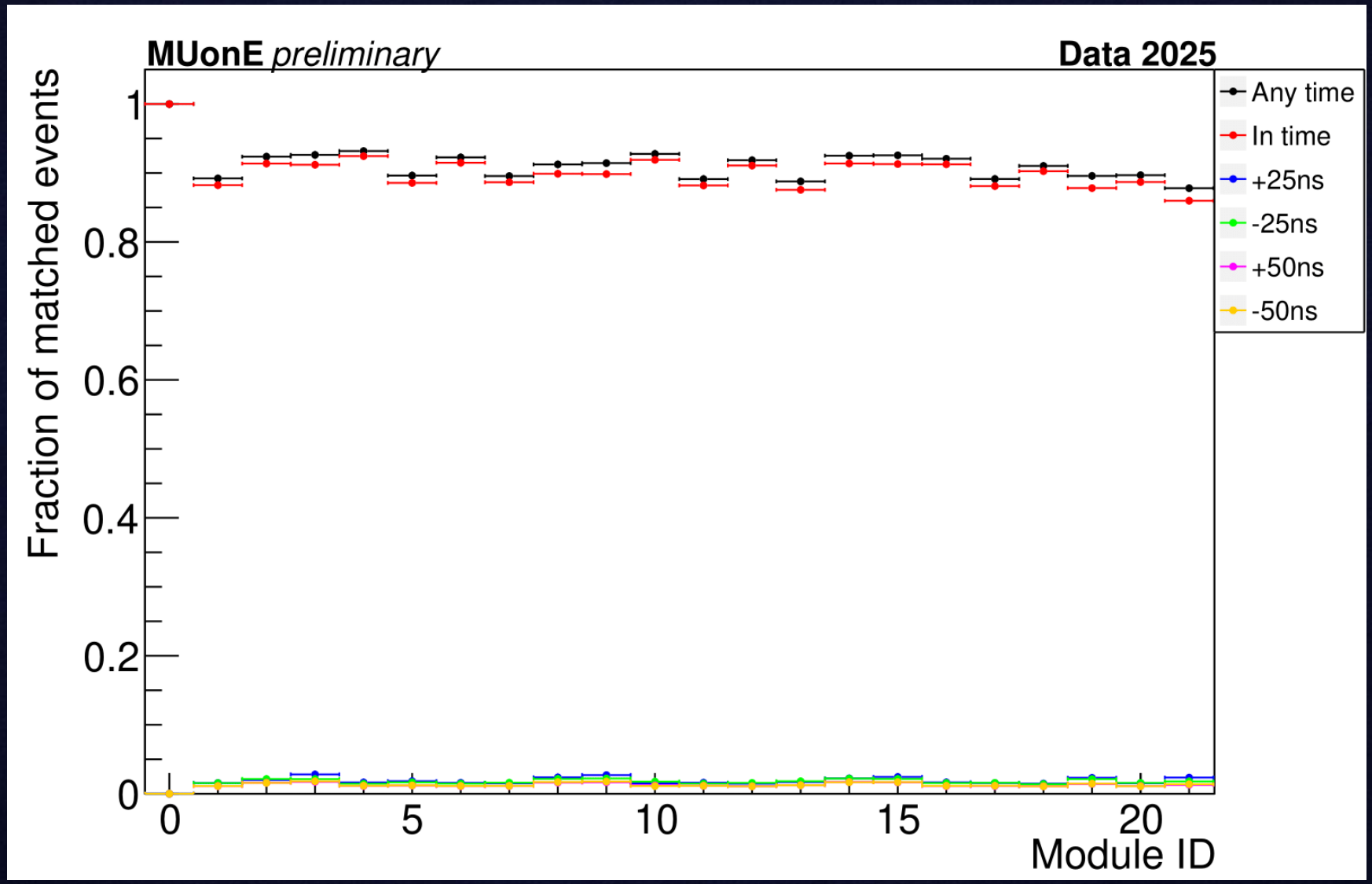
- Fix the internal delay (DLL) of the reference module to 12 ns (centre of the 25 ns window).
- Scan the DLL of all the other modules at the same time



DLL = 15ns: best delay to maximise the coincidences

Synchronisation Procedure – *Results*:

After all the steps the fraction of matched events is maximized for all modules



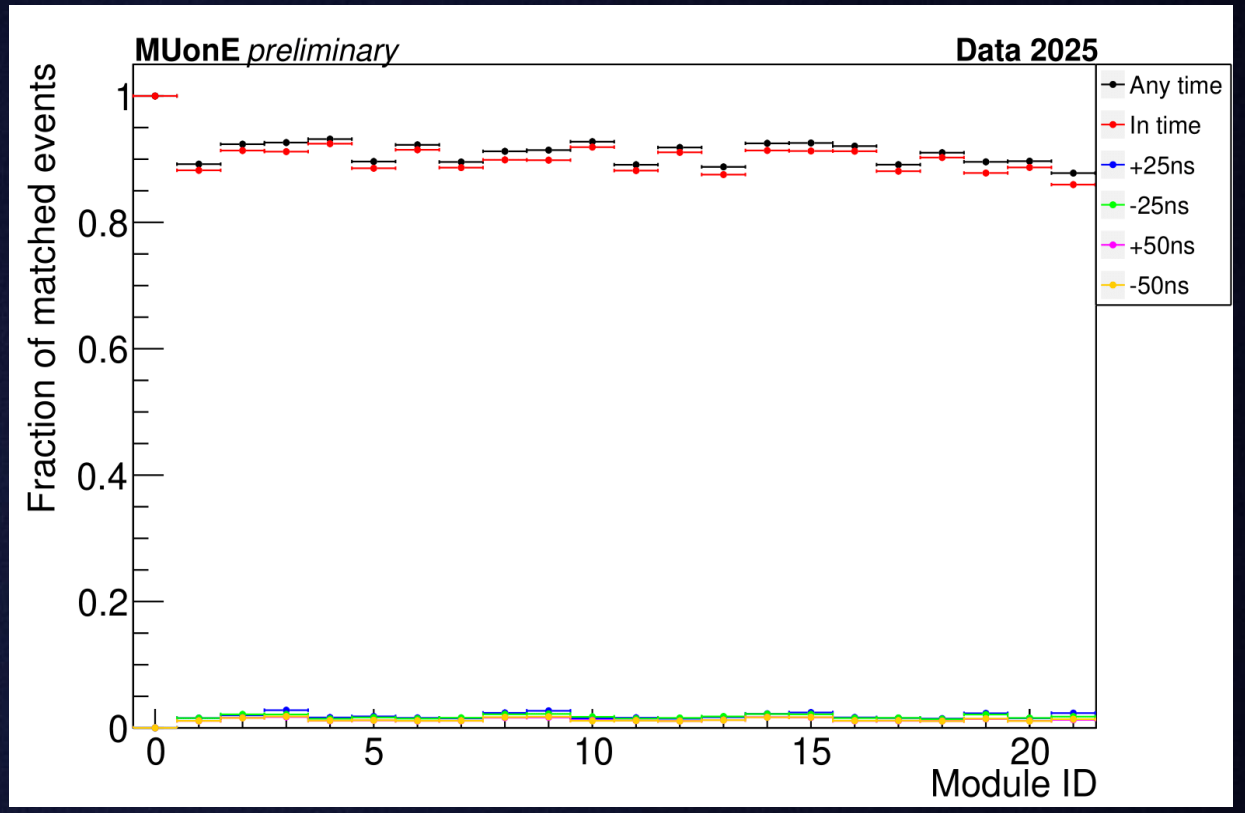
Synchronisation stability over runs

*NB: in the following section the **Fraction of matched events** is labelled **efficiency***

Synchronisation stability test:

We don't have a DLL scan between each run, nor at different time of the run

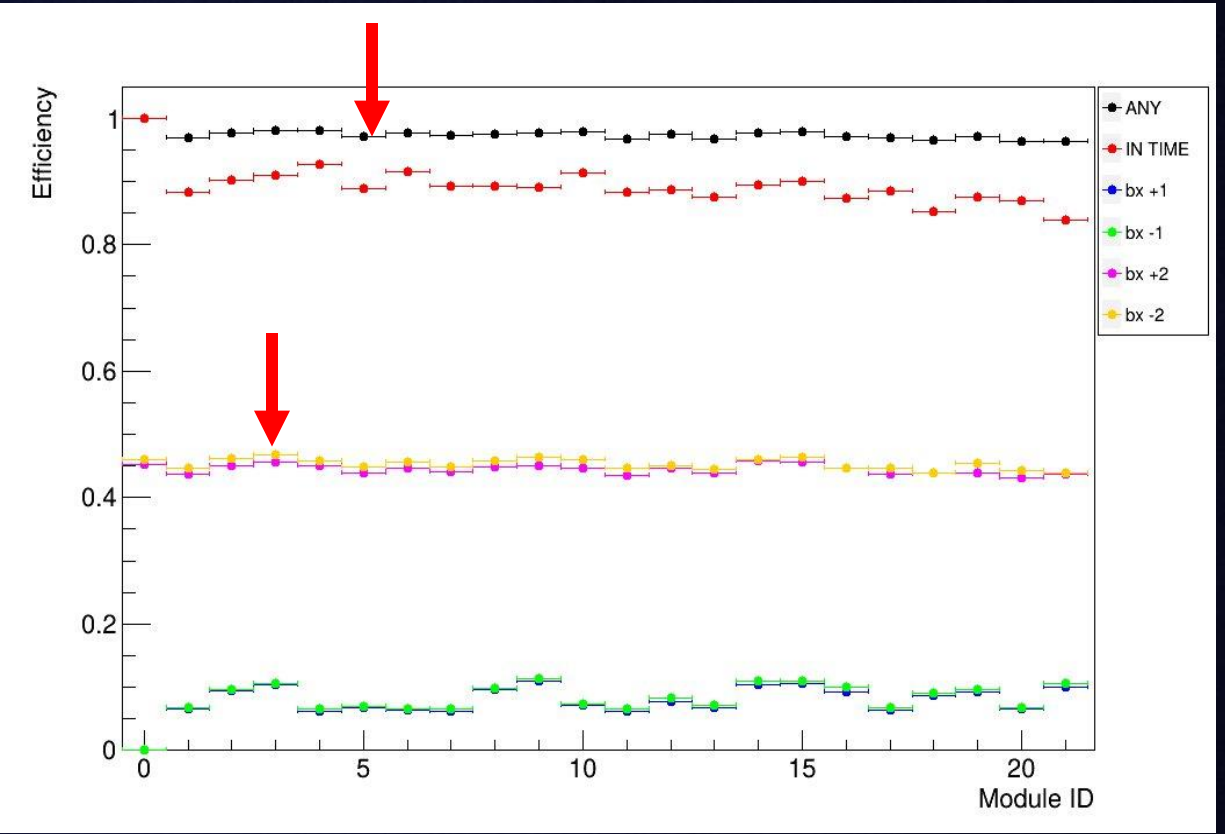
To test the stability of the synchronisation we will produce this same plot for different chunk of files within a run



Synchronisation stability test:

We don't have a DLL scan between each run, nor at different time of the run

To test the stability of the synchronisation we will produce this same plot for different chunk of files within a run



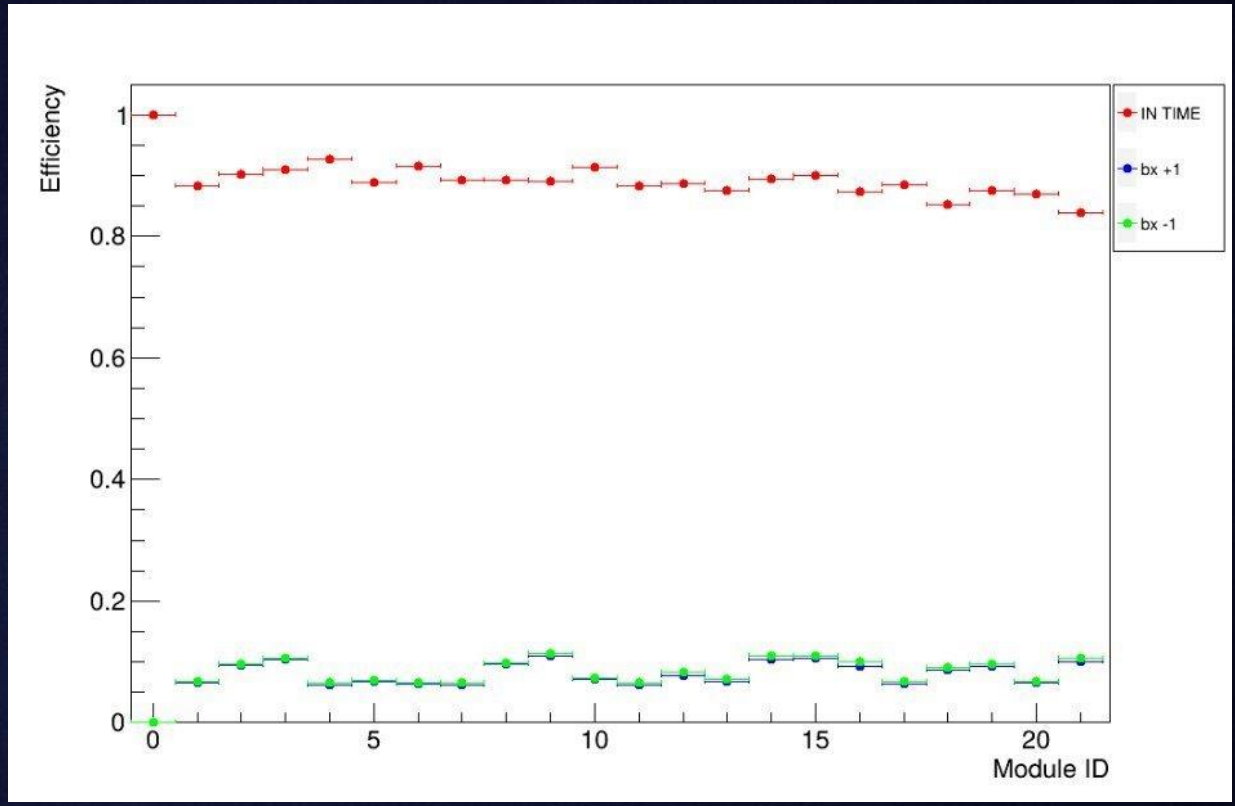
Selection changes:

- Exactly one stubs in the reference module within **3** consecutive Bx rather than 5
  (synch is not likely to change by more than 25ns)

Synchronisation stability test:

We don't have a DLL scan between each run, nor at different time of the run

To test the stability of the synchronisation we will produce this same plot for different chunk of files within a run

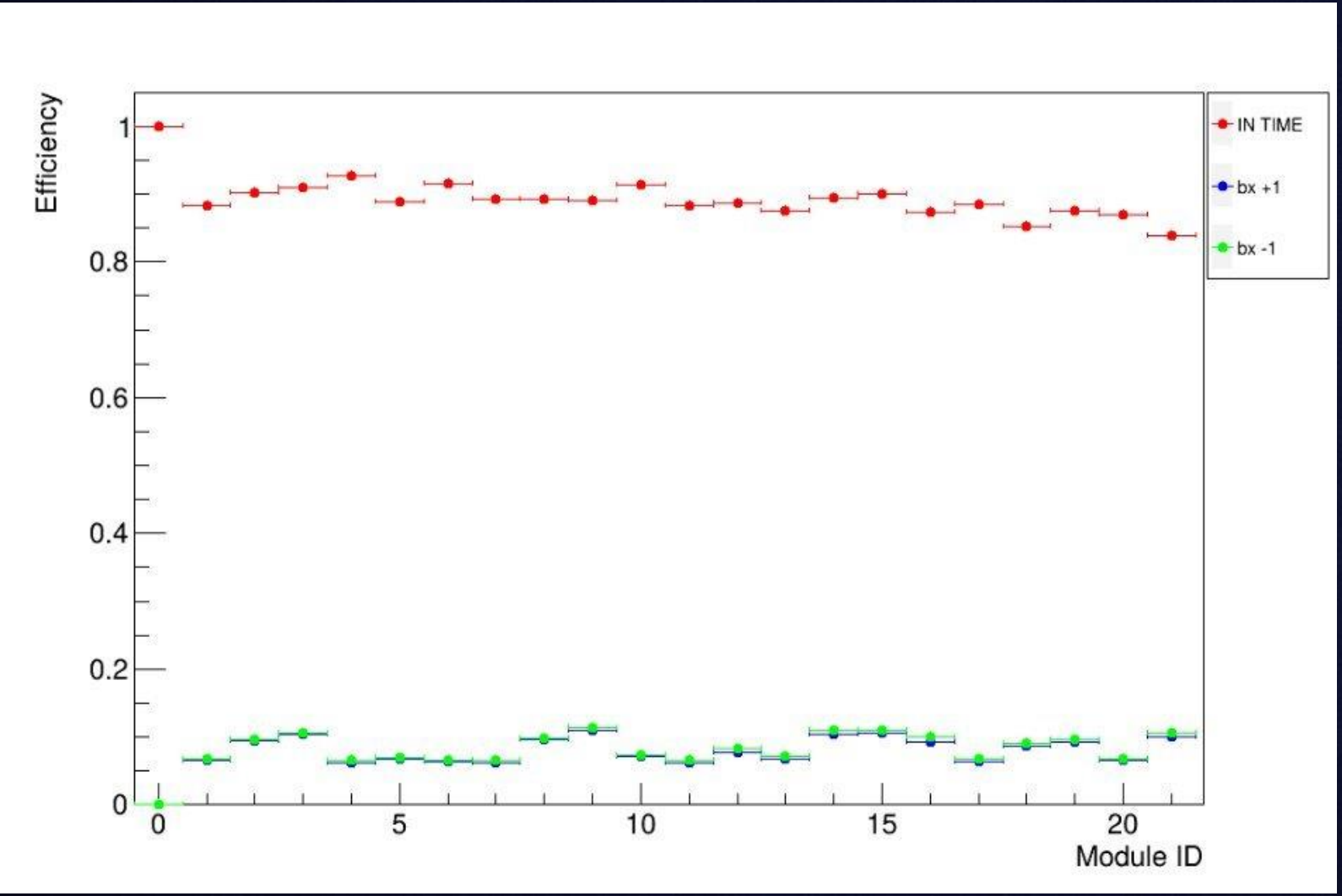


Selection changes:

- Exactly one stubs in the reference module within **3** consecutive Bx rather than 5
  (synch is not likely to change by more than 25ns)

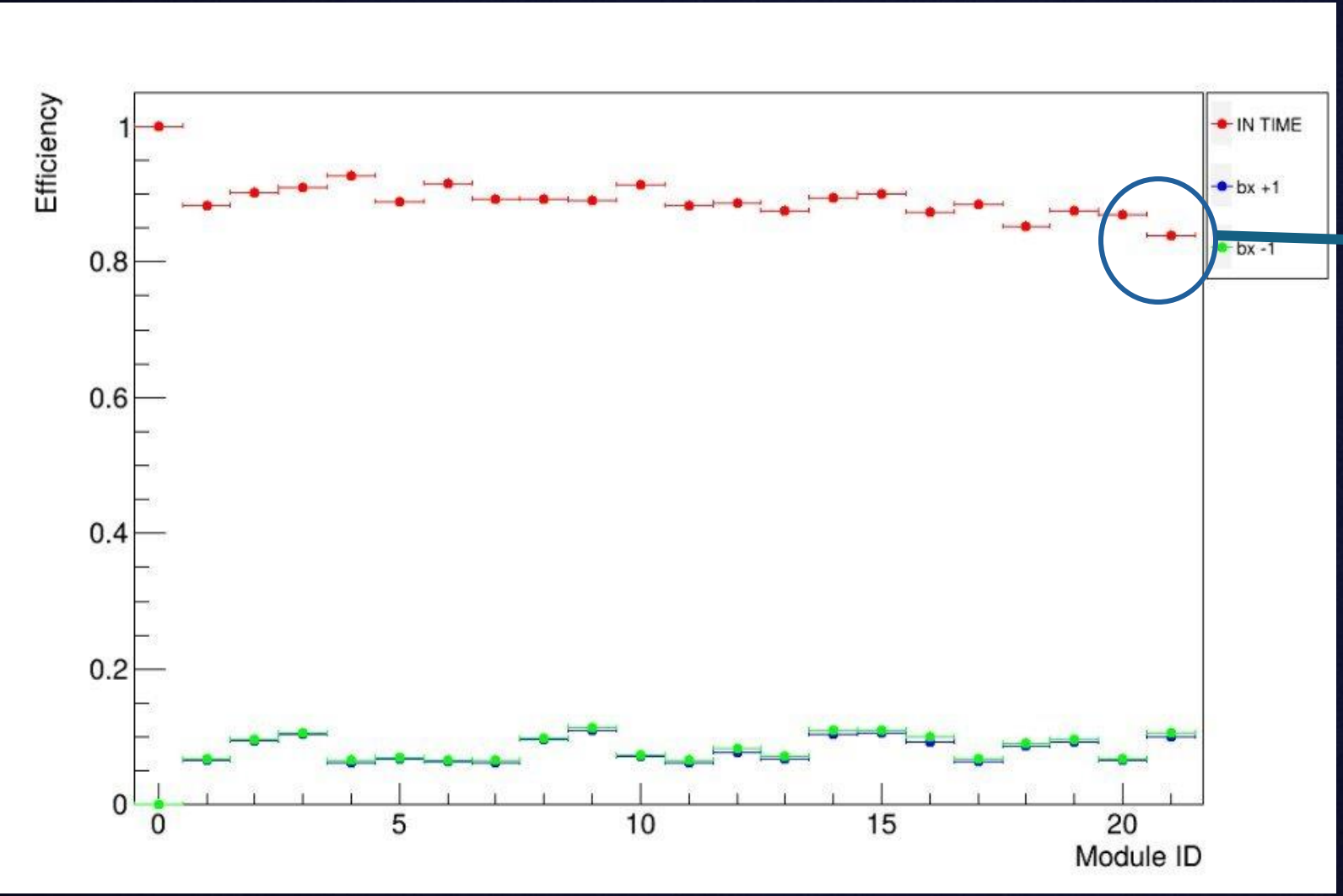
Synchronisation stability test:

How to estimate the synchronisation of modules ?



Synchronisation stability test:

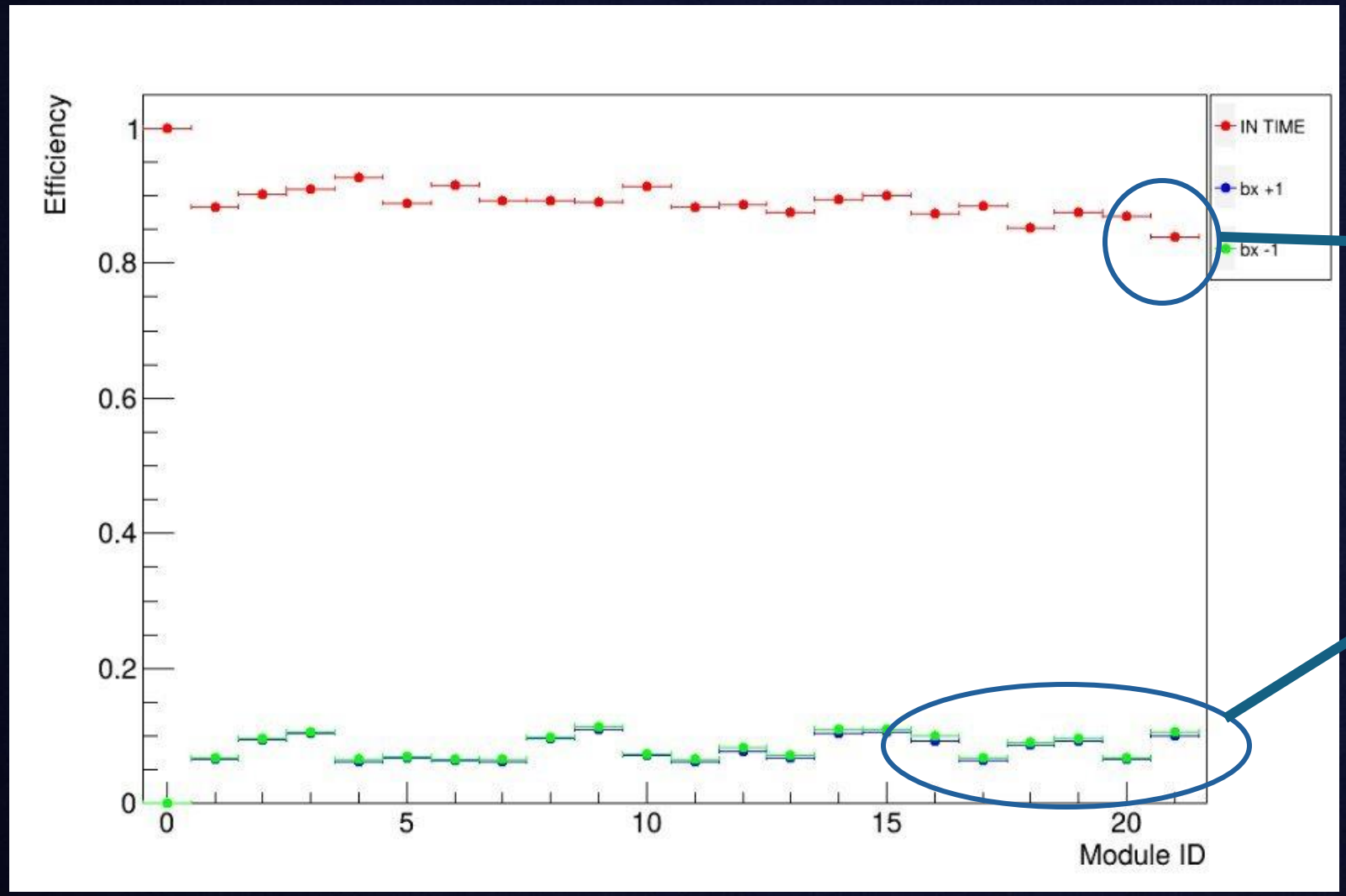
How to estimate the synchronisation of modules ?



If the value of the ration of matched event for a module is stable in time, this mean the modules synchronisation is stable

Synchronisation stability test:

How to estimate the synchronisation of modules ?



If the value of the ration of matched event for a module is stable in time, this mean the modules synchronisation is stable

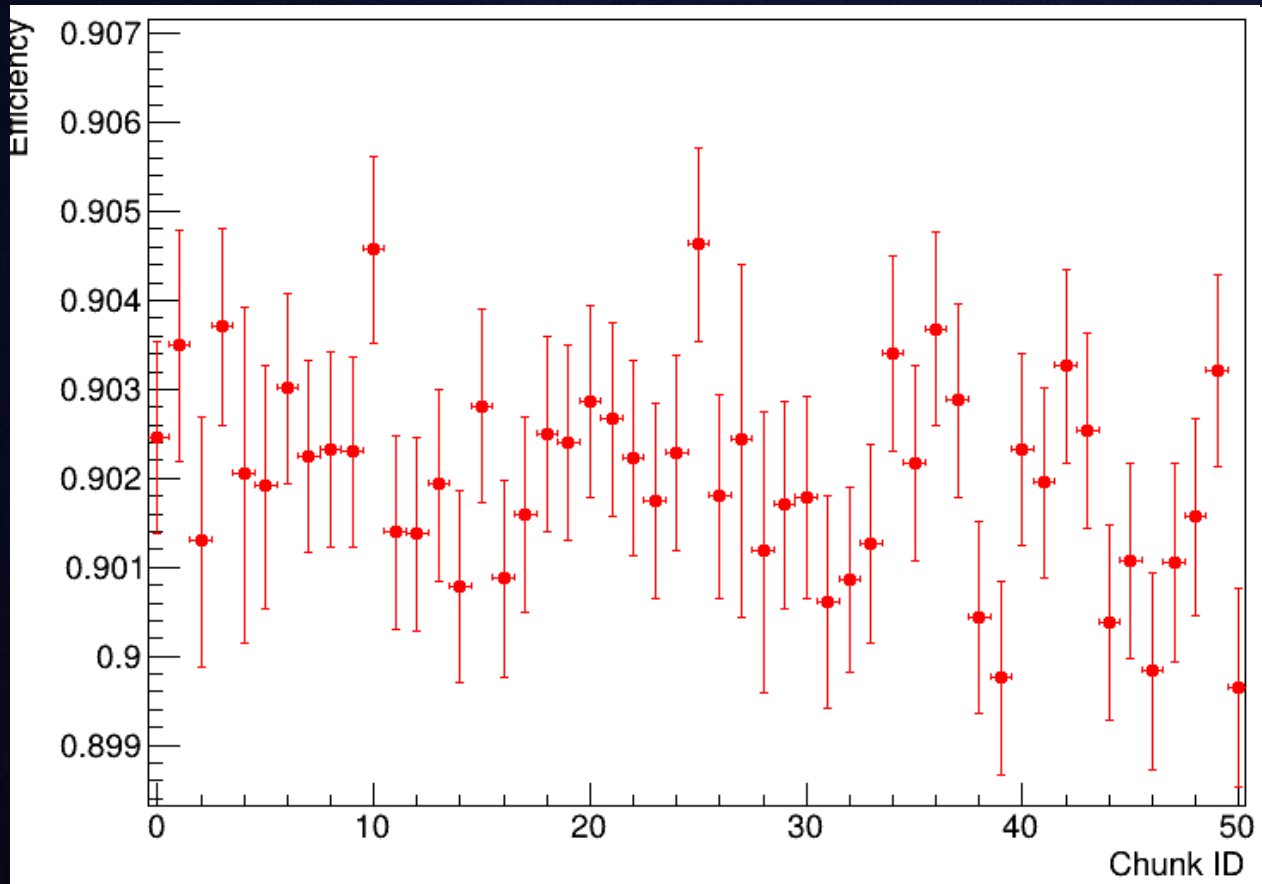
If the ratio in Bx+1 and Bx-1 are equal, this mean they seems to be synchronized

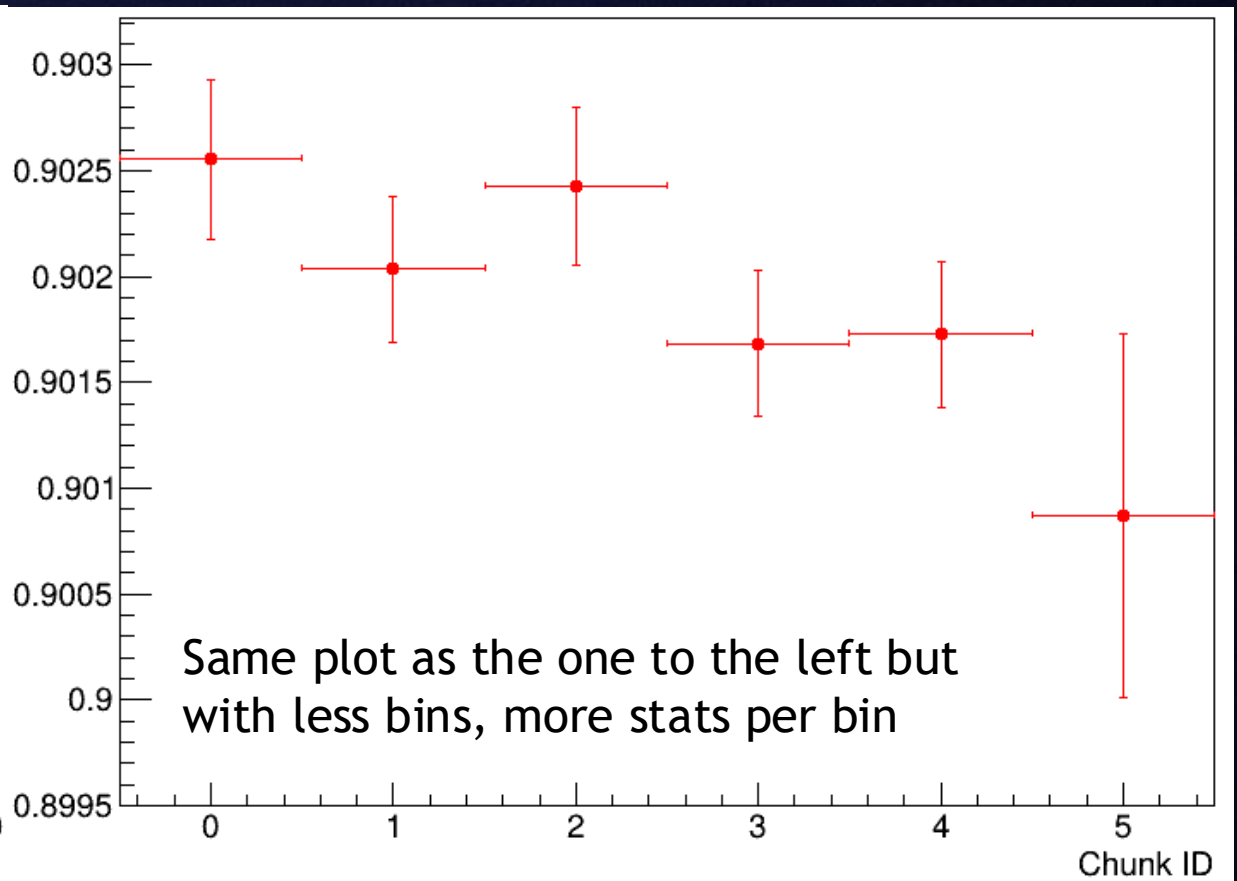
Results - Stability over run 32:

Chunk ID is a chunk of file and depend on time

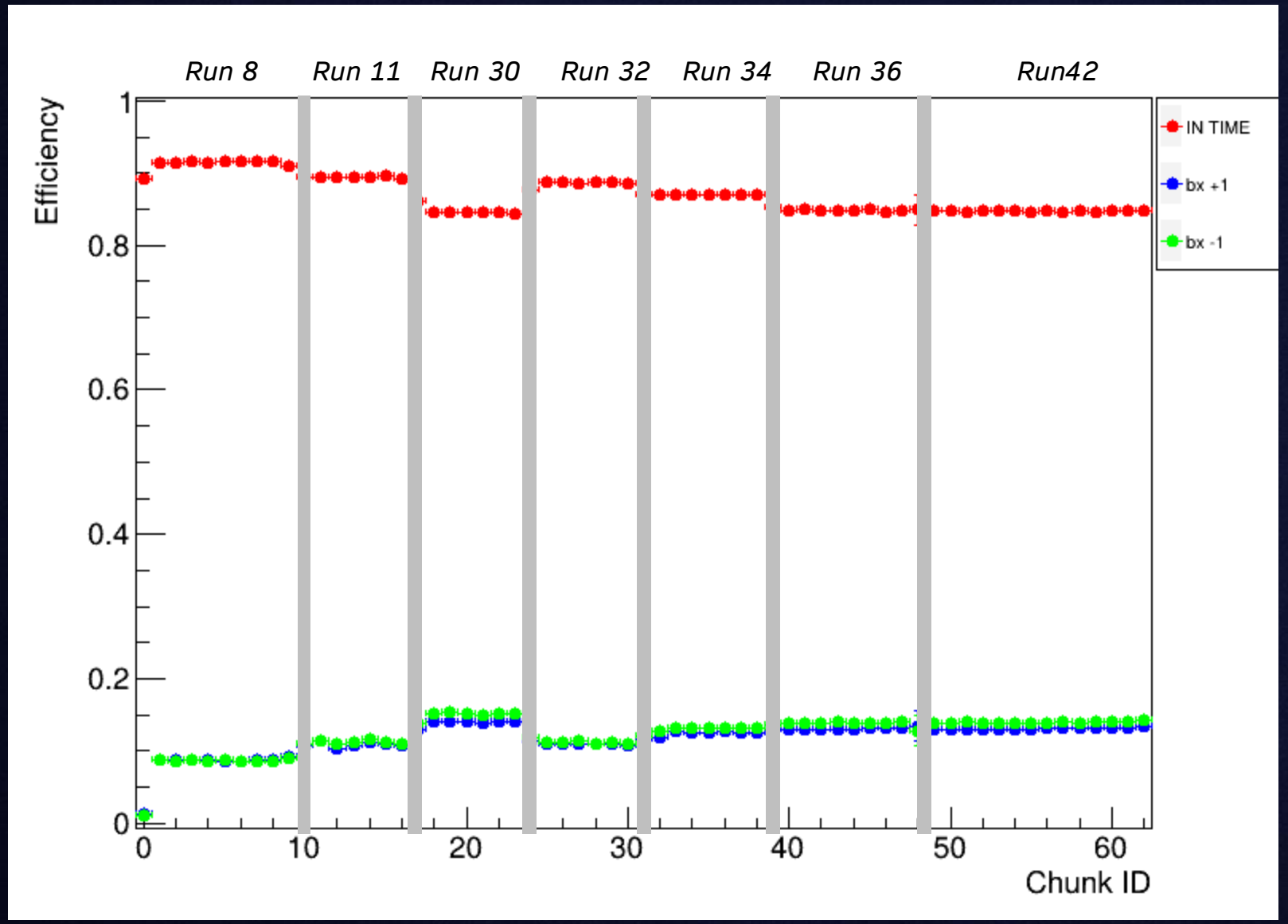
->We see stability at sub-percent level all over the run 32

Plots are for module 2, all modules behave the same





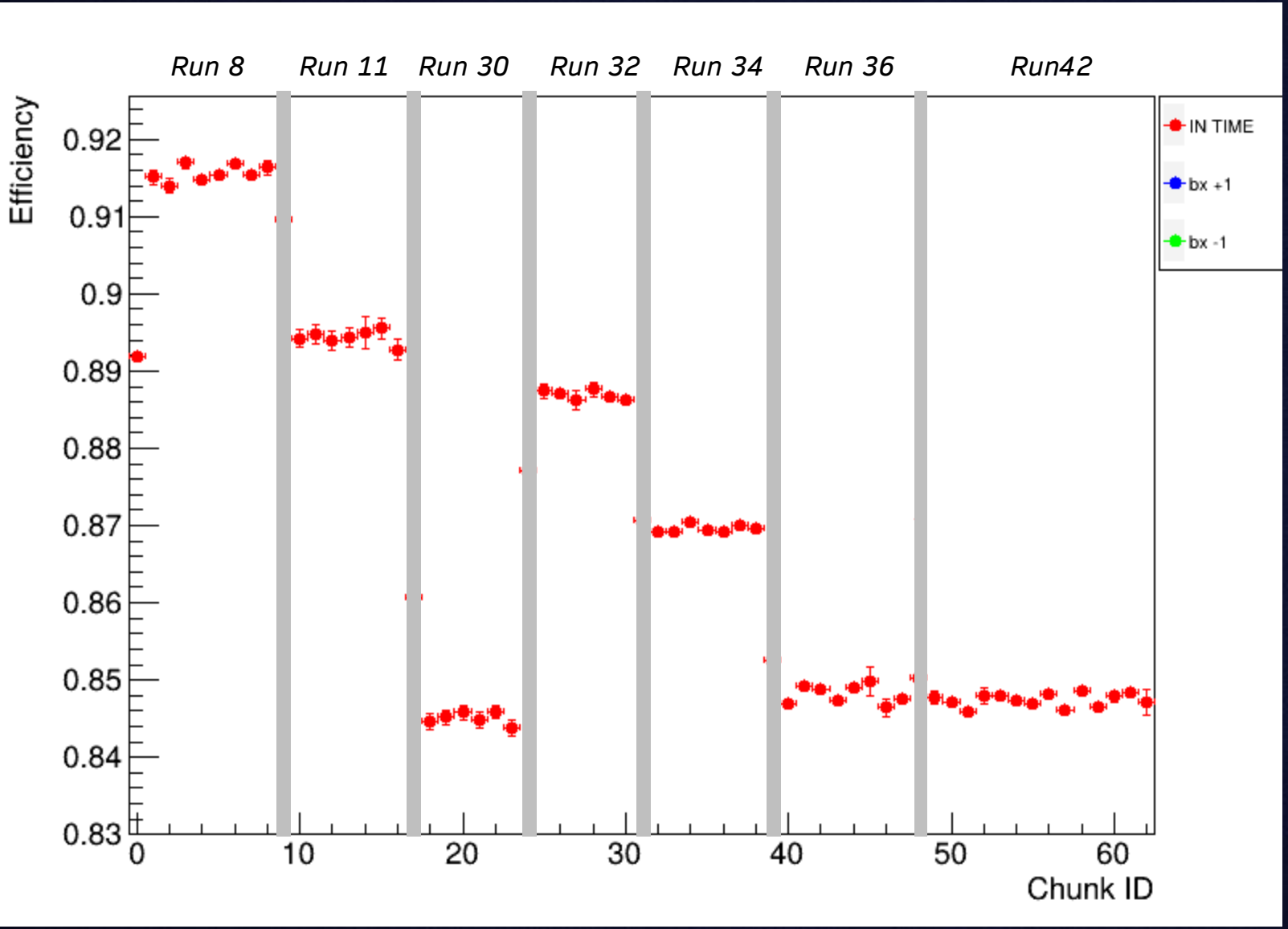
Results – Stability over different runs



Run 8 and 11 was before oxygen run so used a different synchronisation than the others, make sense there is some diff

I don't think there is any synchronisation changes from run 30 to 42 so the differences are strange

Results – Stability over different runs

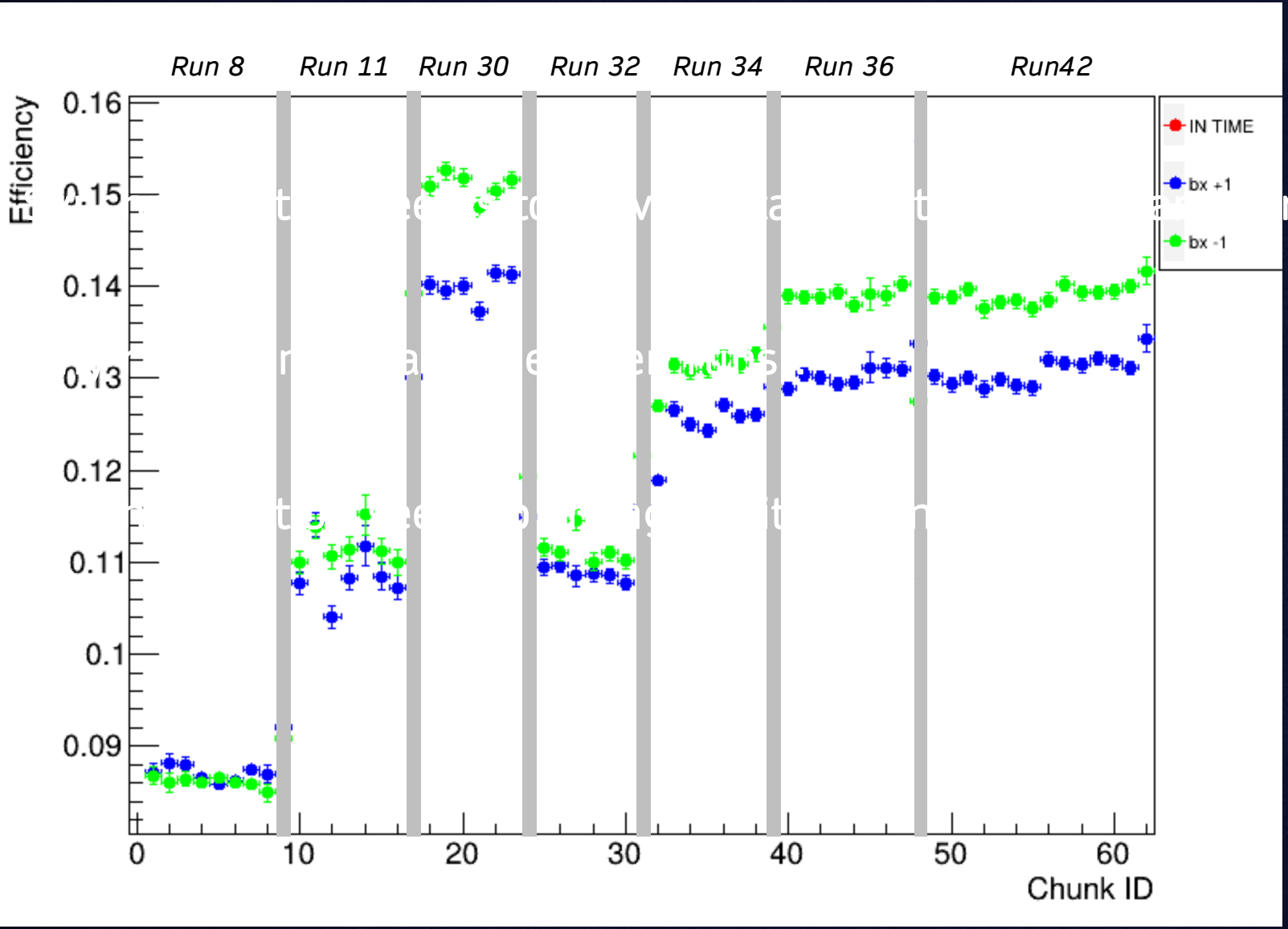


Fluctuation on the ration of match event IN TIME are very large (up to almost 10%)

But within runs they are all very stable

Some modules vary has more or less variation, it seems to be random

Results – Stability over different runs

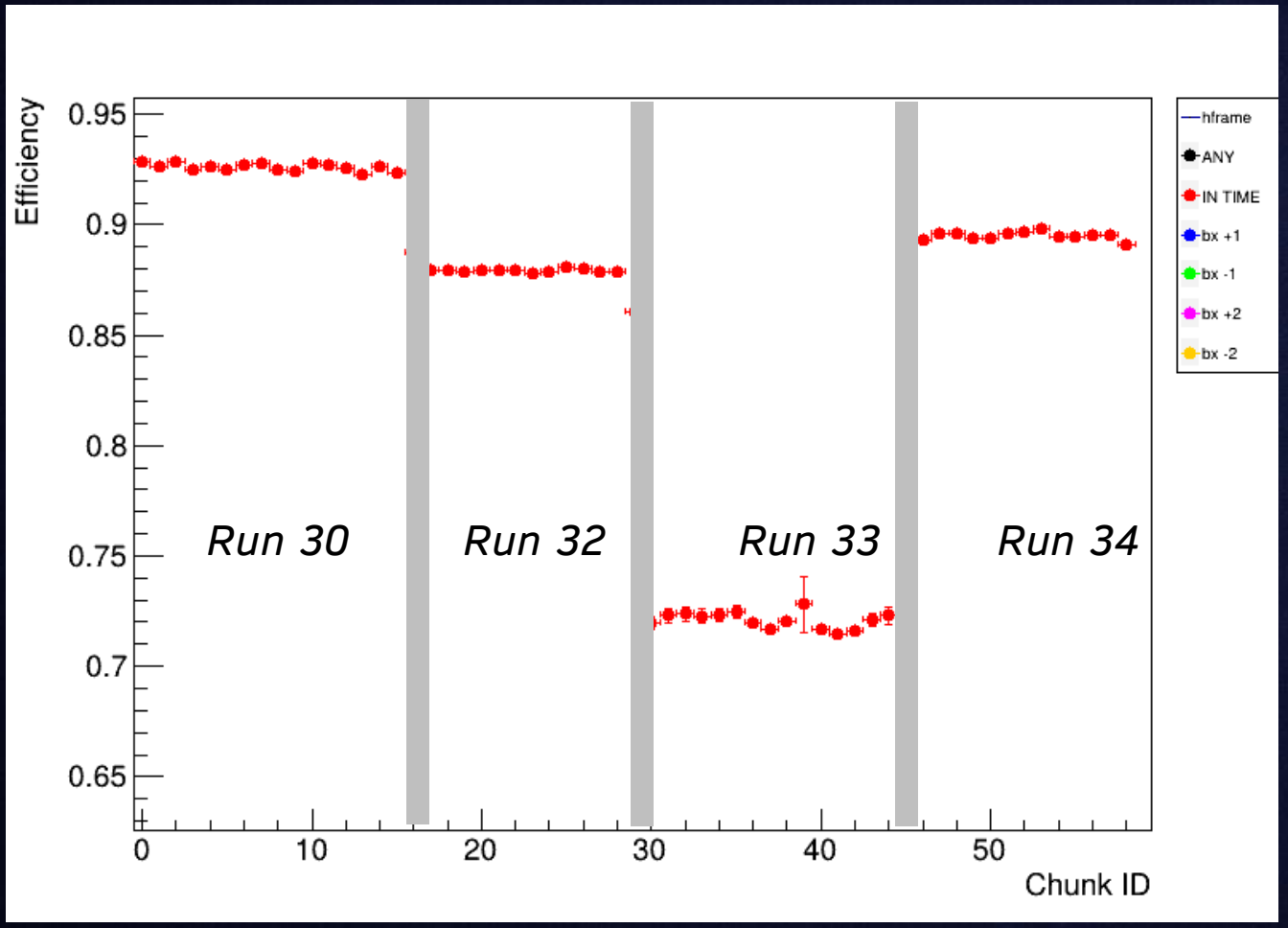


Looking at Bx+1 and Bx-1, it seems the synchronisation for this modules is not optimal (we would expect them to overlap

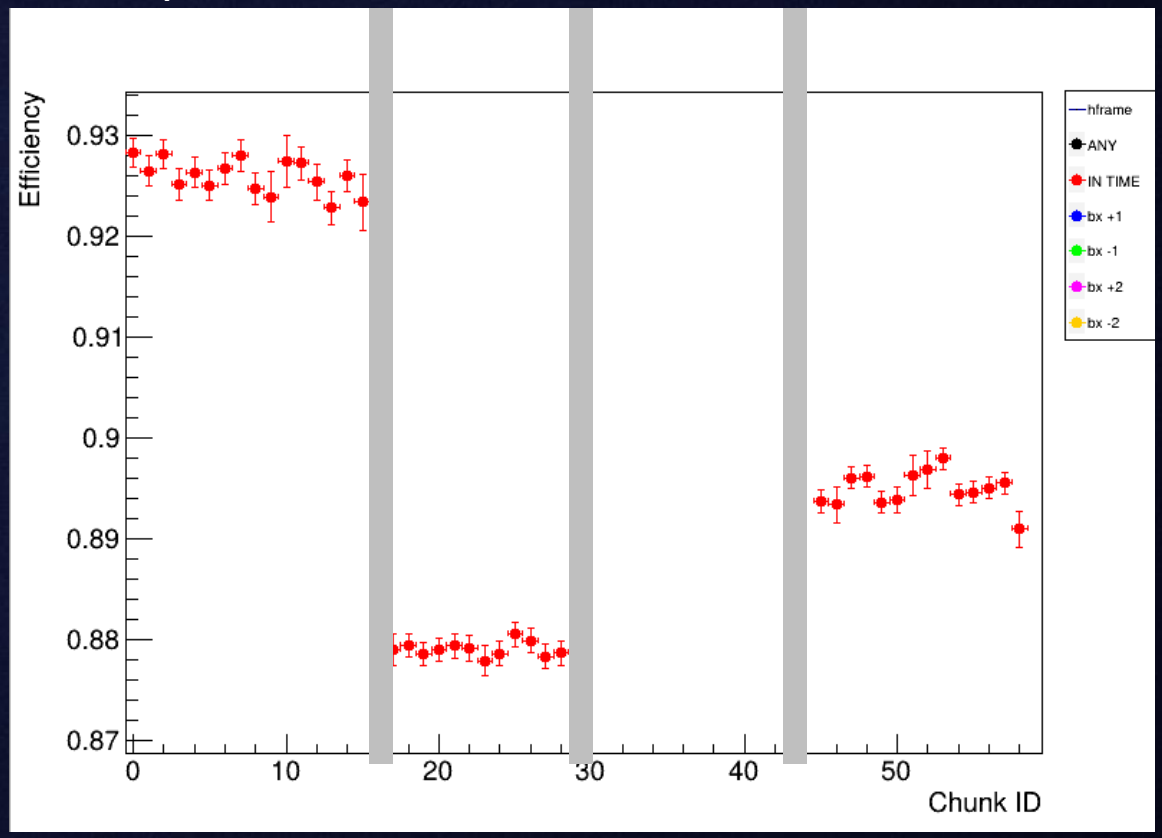
This module does not seems well synchronized for any run except 8/11 ?

Results – run 33:

Run 33 seems very very bad

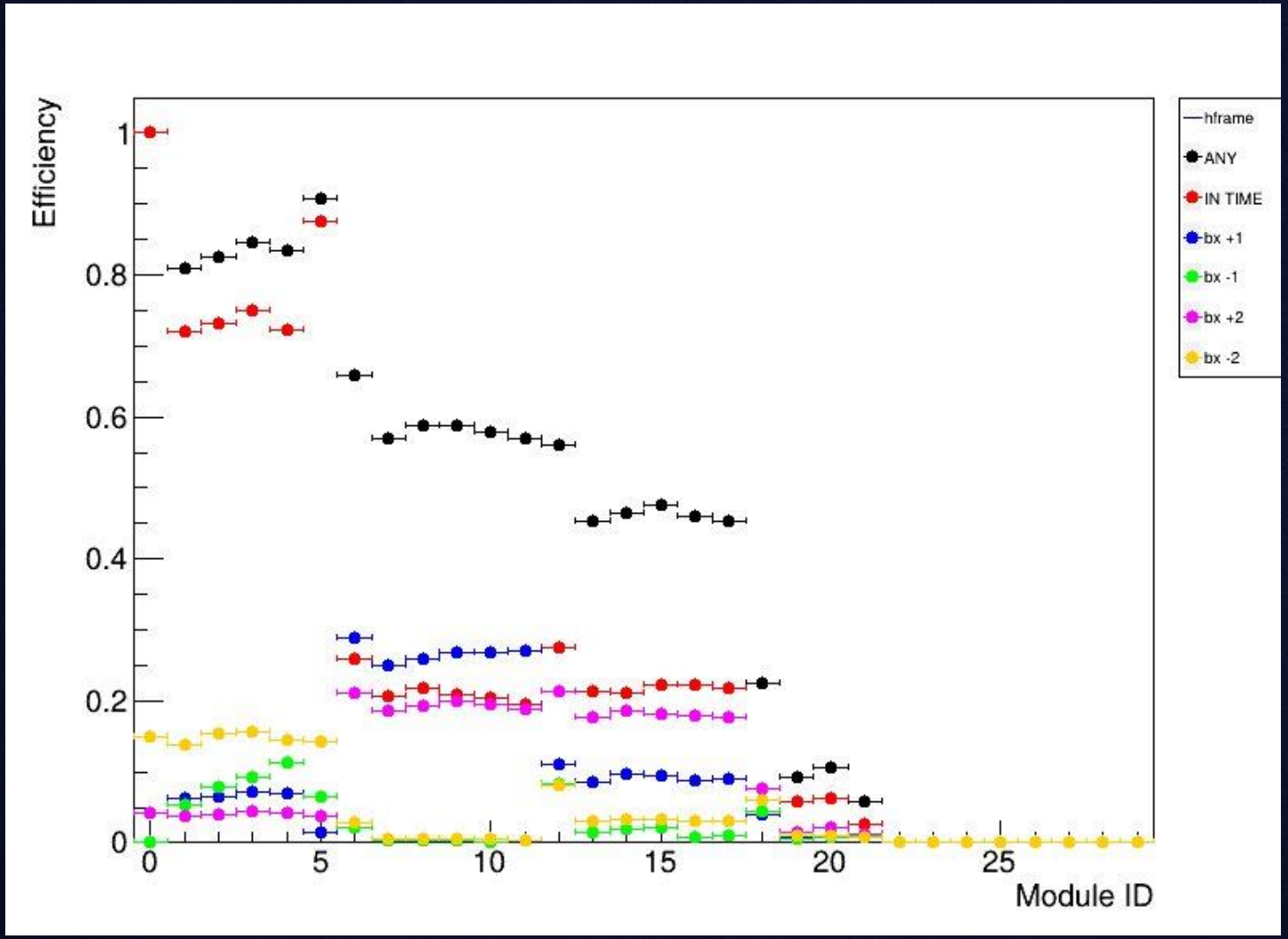


same plot zoomed



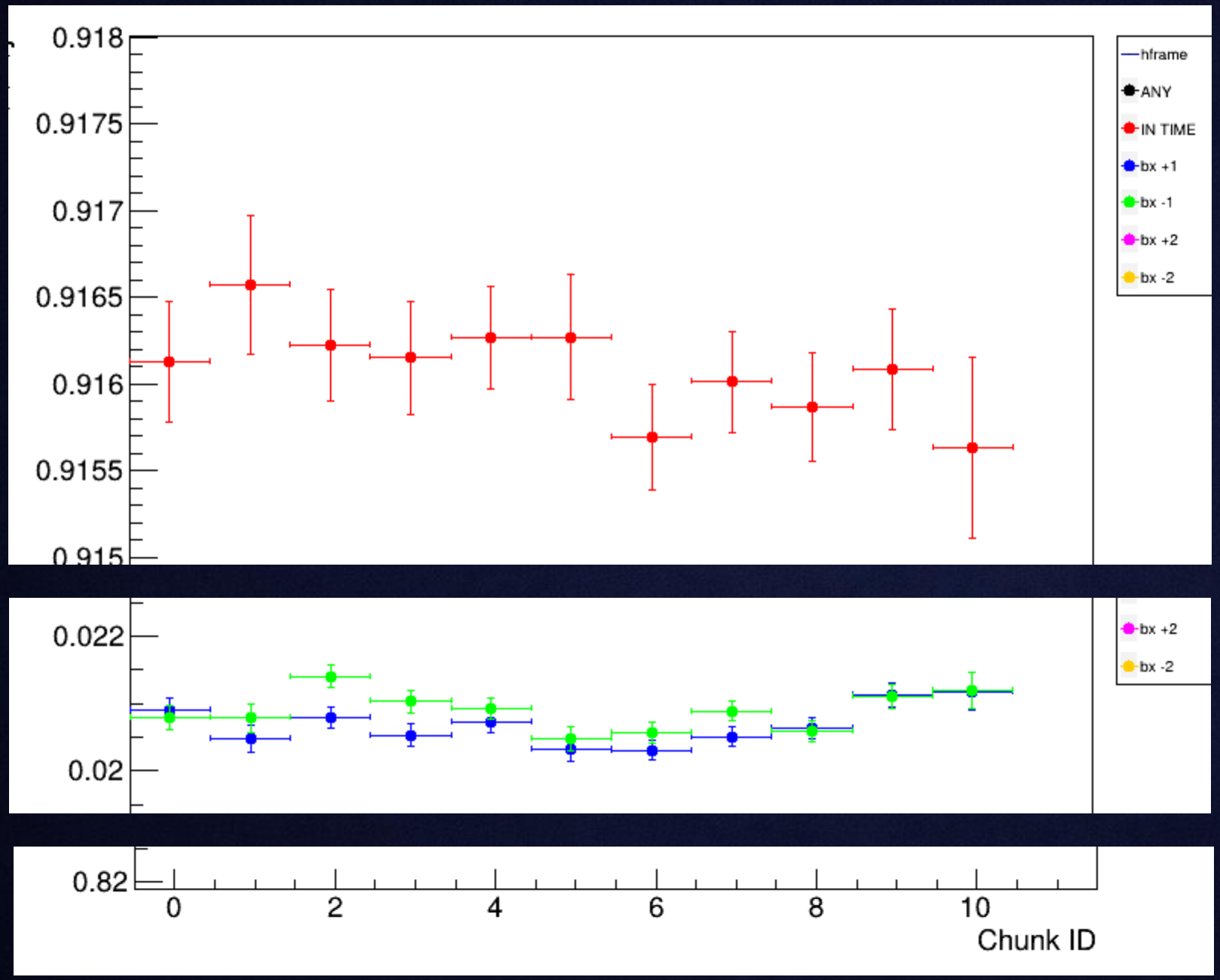
Results – run 33:

Run 33 seems very very bad



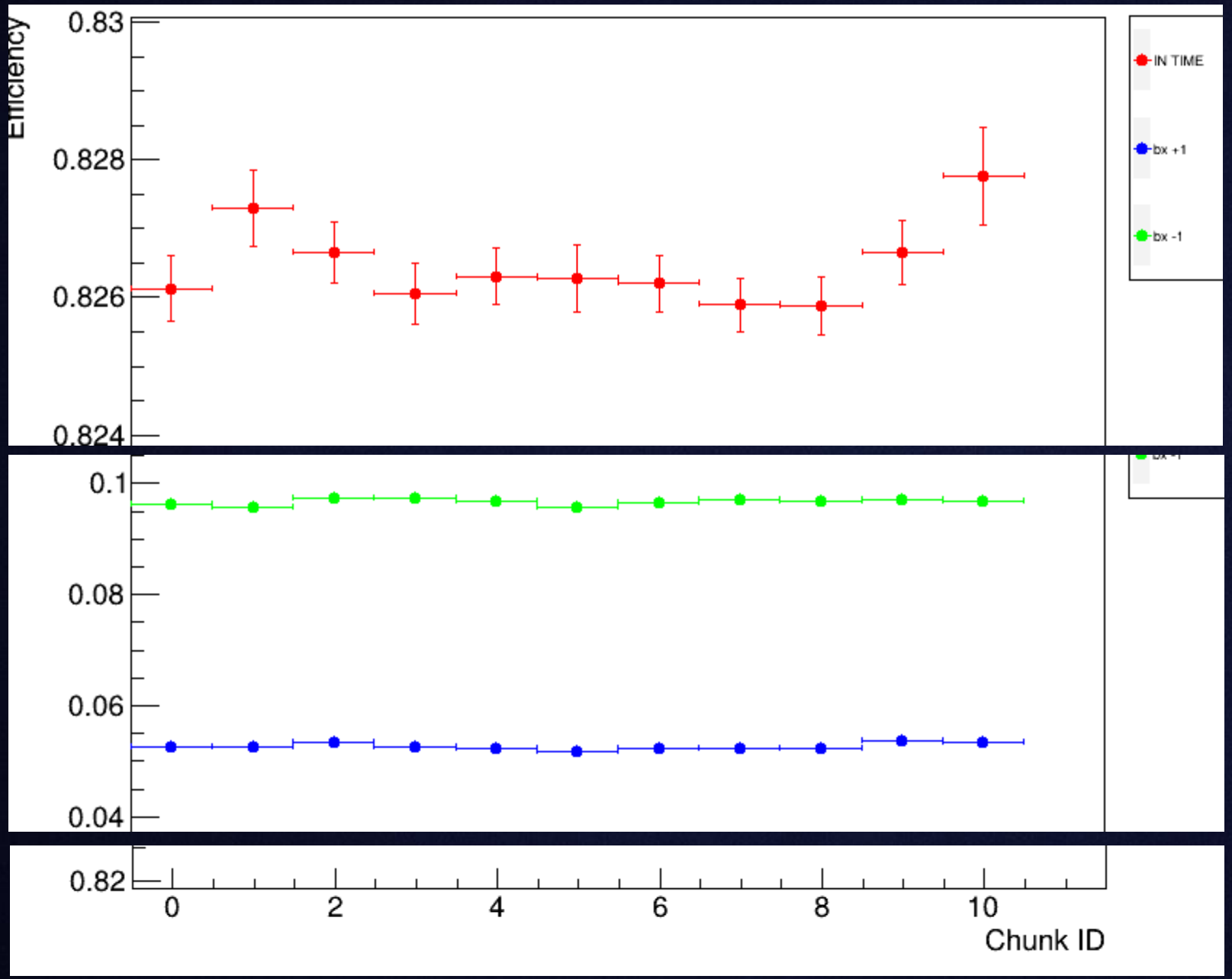
X axis = modules

Results – run 52 (For BMS data):



Run 52 is very stable and some modules looks good...

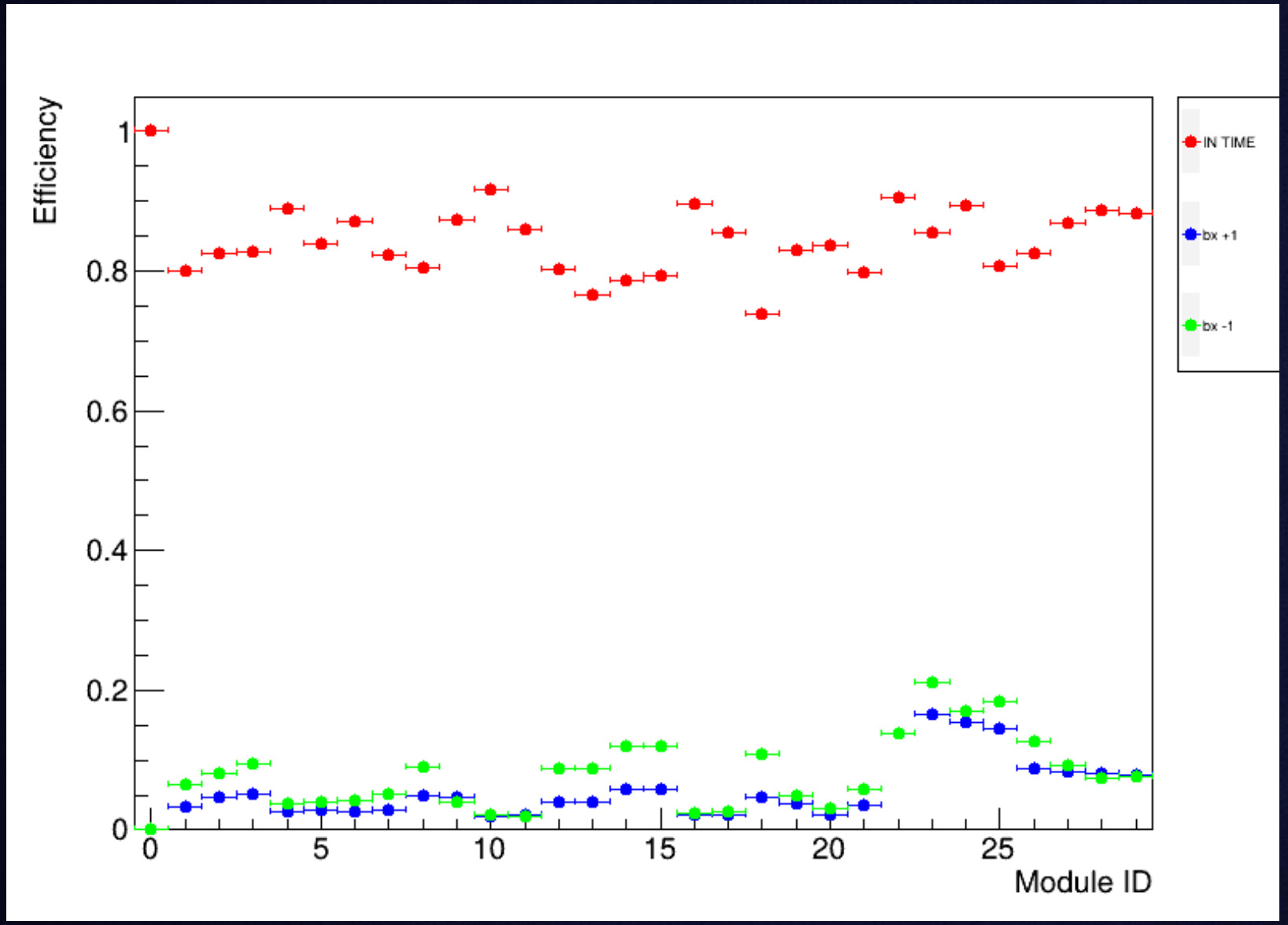
Results – run 52 (For BMS data):



Run 52 is very stable and some modules looks good...

... But some others does not seems very well synchronized

Results – run 52 (For BMS data):



Run 52 is very stable and some modules looks good...

... But some others does not seems very well synchronized

X axis = modules

Conclusion

Synchronisation seems to be very stable within a run (and for all runs)

Why ratio is not stable between runs ?

Synchronisation seem to change a little ~1ns between run ?

Most of runs looks good , run 33 is super bad, run 52 (BMS) quite bad

Thanks