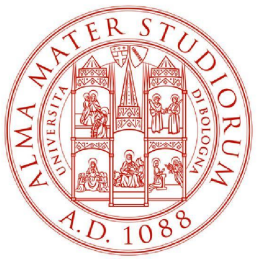
Desynchronization Treatment

Nico Giangiacomi

Universitá di Bologna, Dipartimento di Fisica e Astronomia
Istituto Nazionale di Fisica Nucleare - sezione Bologna

nico.giangiacomi@bo.infn.it

March 3, 2017



Resynchronization

What we would like to do:

1. **detect desynchronization** using L1ID from module and from TIM
2. if L1ID from module $>$ L1ID from tim $\longrightarrow$ add empy fragments $\longrightarrow$ **resynchronize**
3. if L1ID from module $<$ L1ID from tim $\longrightarrow$ discard fragments $\longrightarrow$ **resynchronize**
4. using **BCID** to check that everything is ok

Situation

FEI4

- module L1 = **13 bits** $\rightarrow$ **enough** to check if the module is ahead or behind the others

MCC

- module L1 = **4 bits** $\rightarrow$ not enough!!!!!
- how to **decide** if L1_module $\gtrless$ L1_Tim ??????
- using module **BCID** info to resynchronize
  1. module BC = **8 bits** $\rightarrow$ counter reset after **255 BCs**!
  2. assuming **100 kHz** trigger rate $\rightarrow$ average distance between 2 triggers: **400 BCs**
  3. **impossible** to use BCID to resynchronize

$$\text{Assuming } \textbf{Desynch} \overset{\Downarrow}{=} \textbf{Skipped Triggers}$$

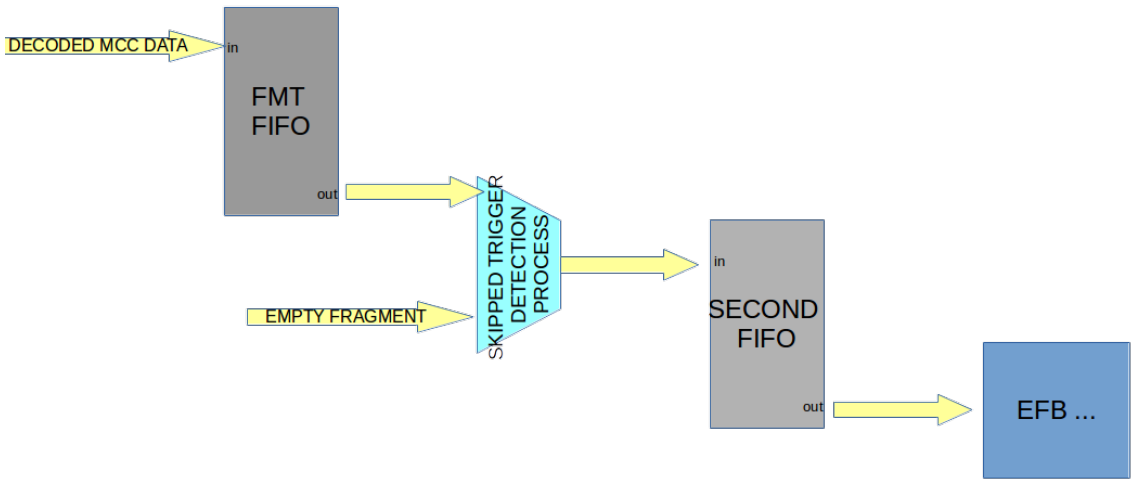
Possible scenarios

After investigation, those are the possible **sources of desychronization** (identified so far):

1. Skipped triggers correctly reported by modules (< 16);
2. Skipped triggers not reported by modules (< 16);
3. Skipped triggers not reported by modules (≥ 16);
4. Bitflip;
5. Extra **random** event.

Where to act

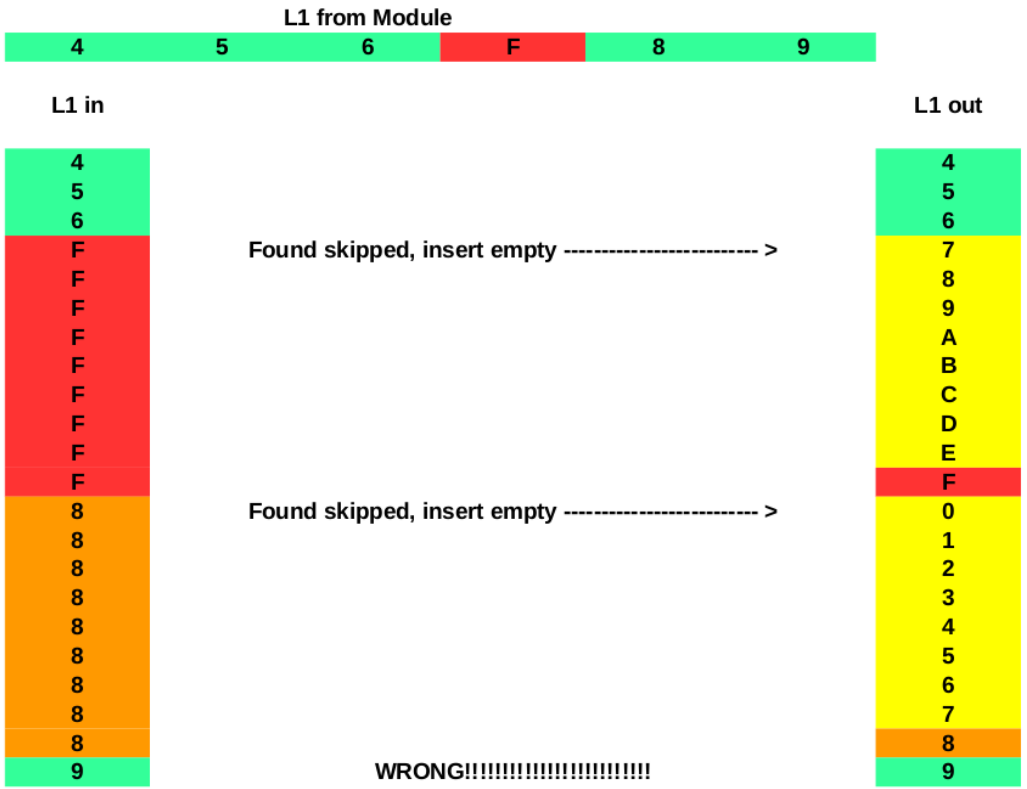
The **resynchronization** algorithm has been added after the formatter fifo (before EFB).
A **second FIFO** has been added which contains the *resynchronized events*.



Resynchronization

What we don't want to happen

- Suppose that there is a **misidentification** of desynchronization.
- Firmware tries to **resynchronize** $\longrightarrow$ **artificially introduced desynchronization**
- Synchronization **lost** until next ECR
- Formatter FIFOs won't empty $\longrightarrow$ **BUSY** signal generated



Proposed Algorithm

How to avoid **artificial desynchronization**:

- try to identify possible bitflips
  1. compare number of different bits
  2. check if both L1 and BC are **different**
  3. don't do anything if bitflip identified
- double check with module flags (not 100% reliable but still useful)
- try to identify **extra events** (see next slide)
- check if desynchronization is *real*
  1. if desynchronization identified $\longrightarrow$ don't do anything but mark event as **suspicious**
  2. if following event is desynchronized $\longrightarrow$ desynch is **real**!
  3. **problem:** desynch corrected 1 event later (unavoidable)

Extra Events

SR1 tests in slot C1_S7 showed a *weird module behaviour* when running with detector at a 50kHz trigger frequency:

- sometimes the module add an **extra** (random????) event between two consecutive events;
- this behavior has been observed only at **high trigger rate** and **high occupancy**.

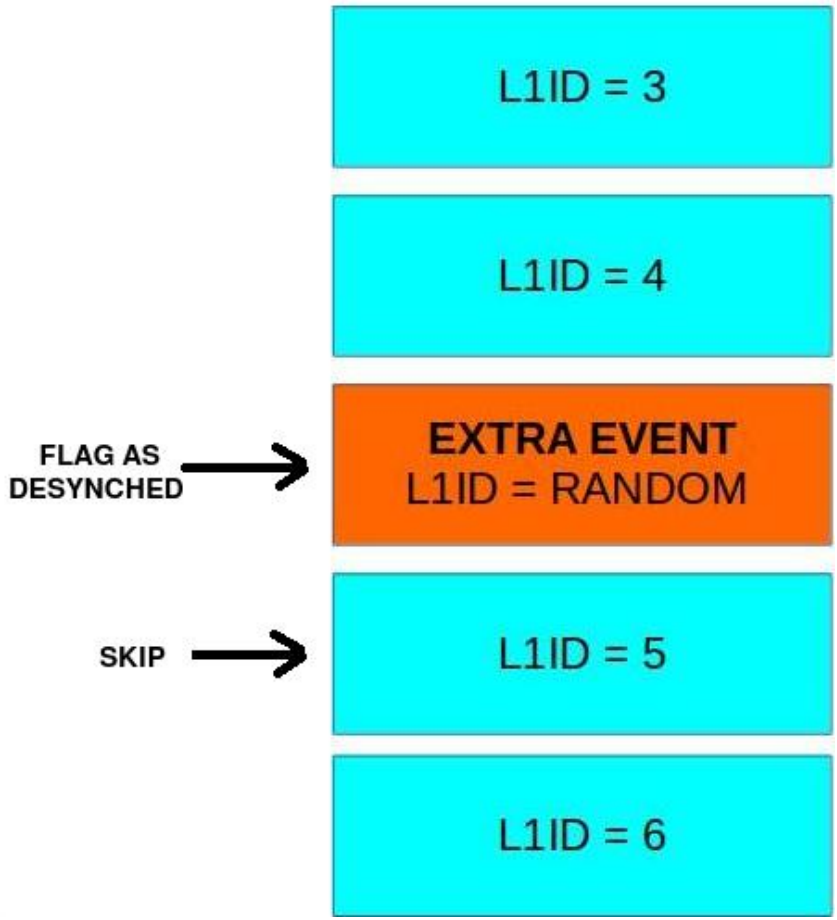
Extra Events

SR1 tests in slot C1_S7 showed a *weird module behaviour* when running with detector at a 50kHz trigger frequency:

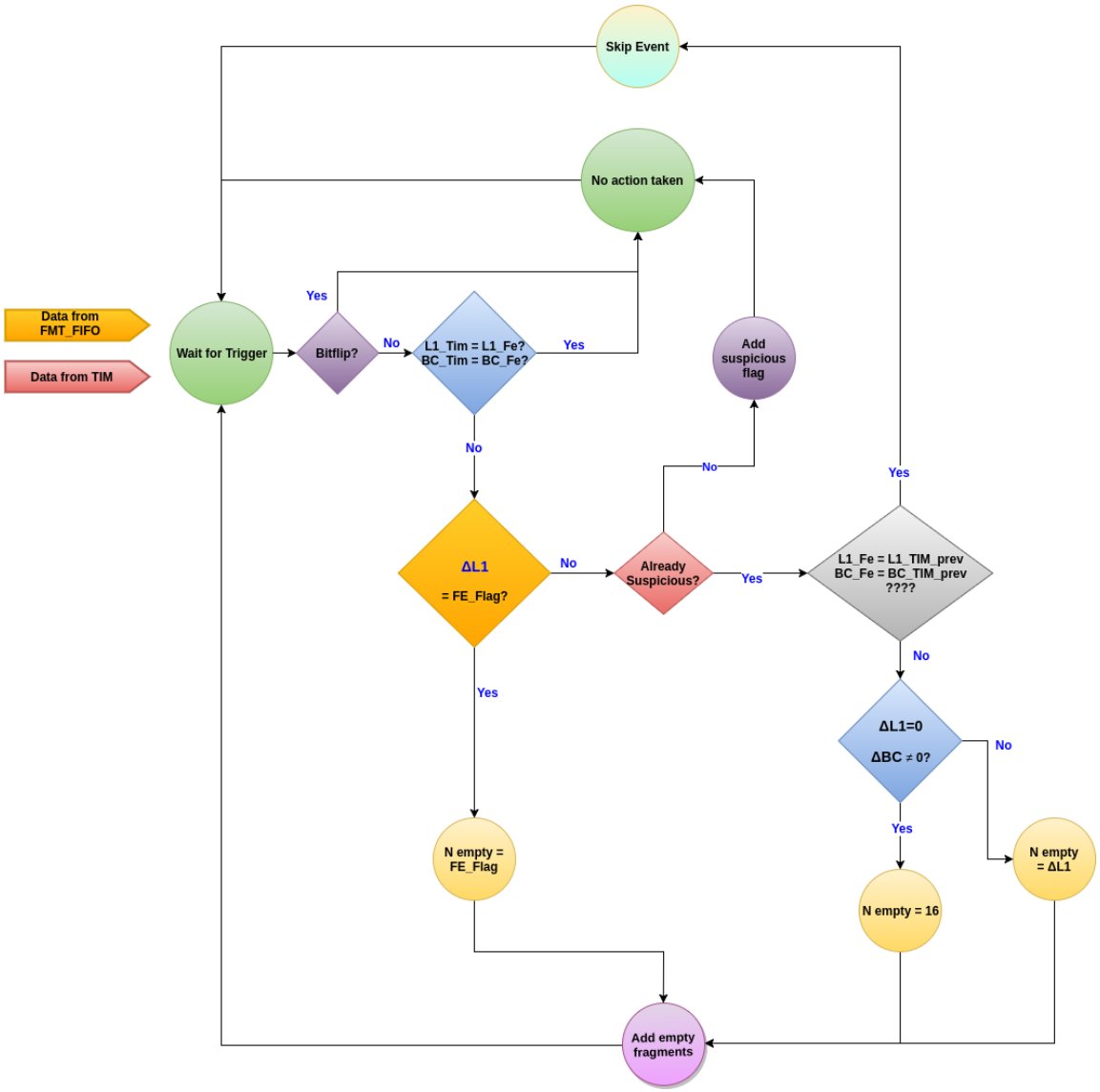
- sometimes the module add an **extra** (random????) event between two consecutive events;
- this behavior has been observed only at **high trigger rate** and **high occupancy**.

How to solve it

- Not possible to **online** recognize Extra Events;
- the extra event will be flagged as **suspicious** and desynchronized;
- the following event can be **identified** since it will have:
  1. same L1ID as previous TIM one;
  2. same BCID as previous TIM one;
- skipping the following event will resynchronize the following events.



Proposed Algorithm



Monitoring Desynchronization

Skipped triggers monitoring

At the same time it is possible to **monitor** the number of pending triggers (not yet processed) inside the ROD

- +1 for incoming trigger
- -1 when front-end data arrives
- **not reliable** (edge effect due to propagation of signal)
- useful for **monitoring** (should be comparable to number of skipped trigger)
- possibility to **protect** the front-ends (send busy after certain threshold)
- **tunable** (threshold = 16? 15? 14?)