Rollover and other misalignments

Pietro, Sandro, Roberto

ALCOR day - 25 February 2025

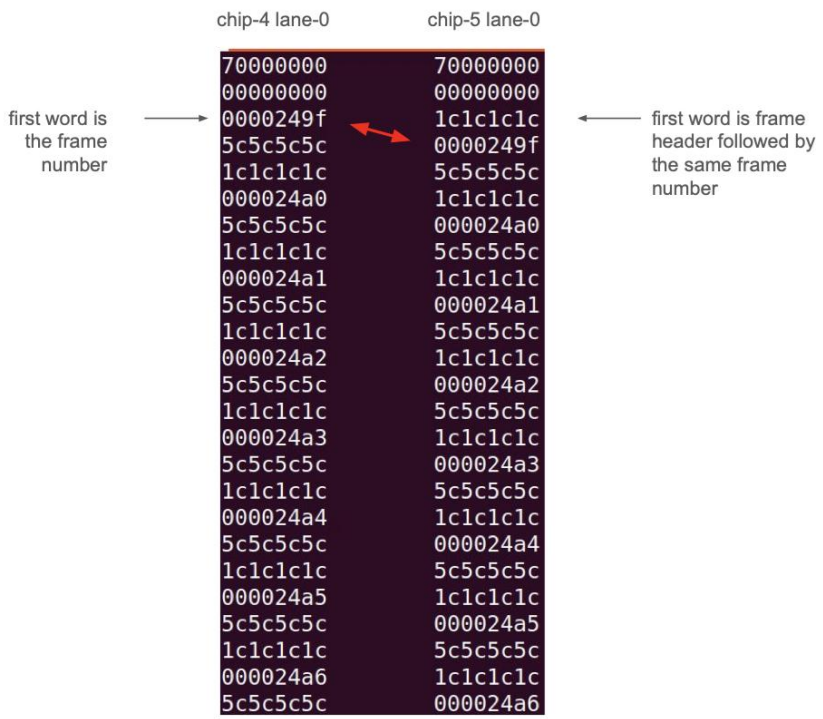
This presentation

- re-cap understanding + anomalies seen (up to Mid Feb)
- recent debug results [and conversations!]
- looking forward
  - synchronization in next test beam (and also in lab) with ALCOR v2
  - synchronization with ALCOR v3

Understanding up to few weeks ago

- At test beam (and also in lab) we observed in data several lanes having sometimes one rollover (RO) more, like the foreseen synchronization was not working properly.
- Note: all synch on data is based on RO.
- **Suggestion (Fabio)**: given coarse counter is started for each lane during setup phase (ECCR setting), send a soft reset continuously ("jamming") in such a way no chip reaches rollover. Init all chip, then remove jamming.
- jamming results presented 11/2 by Roberto.
- it "works" but anomalies were still found.

Anomaly #1
first frame mismatch



Anomaly #2
rollover mismatch

interpretation (Fabio): given SOFT reset is asserted completely asynchronously with respect to the state of coarse counter, depending on when the SR is sent (very close but before a rollover, immediately after etc.) we might have some instabilities.

Anomalies and test pulses

With test pulses

number of rollover mismatch

test pulse →

this run was selected as anomalous based on the mismatch on the total number of rollovers

test pulse words are not recorded in the same frame on all chips

Reminder:

(at each spill rising edge an SR is sent)

1. Jamming: using 20 kHz spill pulses with 500 ns width (*EC enable* bits stay at 0).

2. Data taking: 2kHz test pulses considering a spill width of 10 ms and lower frequency.

Our understanding since last ALCOR meeting (May):

after a SR is sent, **first word** out is a **frame header** (0x1C1C1C1C).

Our understanding after debug these days with FPGA + chats with Fabio:

- when a SR is sent the coarse counter goes to 0 while a frame is already opened, so **first word seen different from idle (in absence of pulses) is a RO** (0x5c5c5c5c) followed by words closing the frame + next frame header/new frame counter.
    - **Verified through a ILA IP core** set after deserialization inside the FPGA firmware.

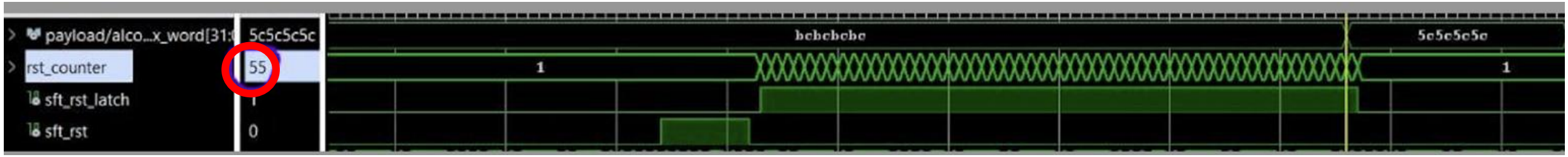
Check: what happens when a SR is sent?

Following chat with Fabio we understand that:

event words or **idle words** are sent **until the coarse counter reaches rollover**. At that point the RO (and frame closing) is sent. If this is true **we should count 32768 = 2^{15} clock cycles between the SR falling edge and the observation of the RO.**

We checked the above statement sending the **SR in a completely asynchronous way** with respect to the ALCOR internal coarse counter. **So the SR can be everywhere in the coarse counter range (0-32768).**

Findings: the **# of clock cycles varies a lot**! Also very few (55!) or O(33000)! This is surprising and not understood.



We then tried to verify without sending SR which is the distance between 2 ROs .

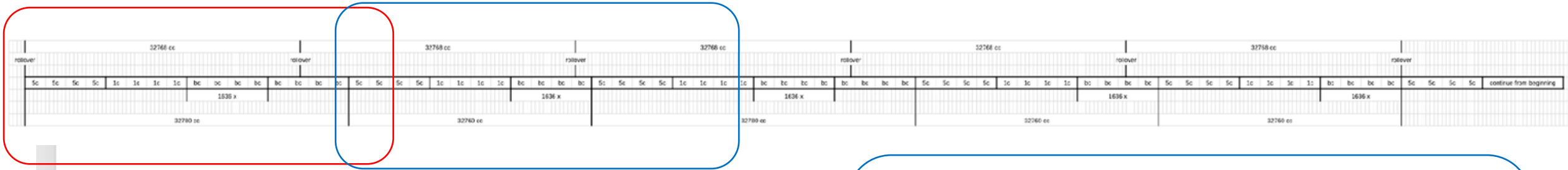
Check: time distance between rollover signals

Check: without soft reset is the rollover distance correct?

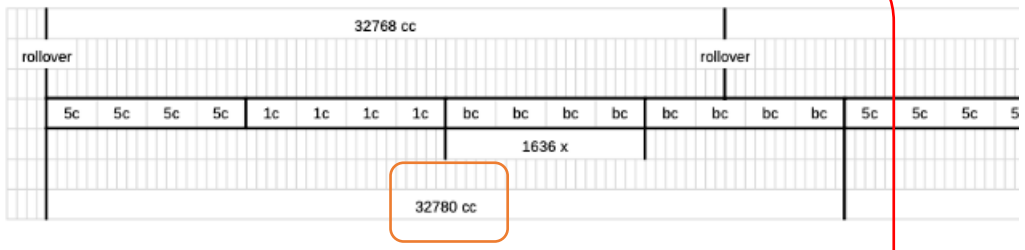
After data serialization we count the **clock cycles between two consecutive RO signals**.

Result: two different fixed values were measured: **32780 and 32760 cc**.

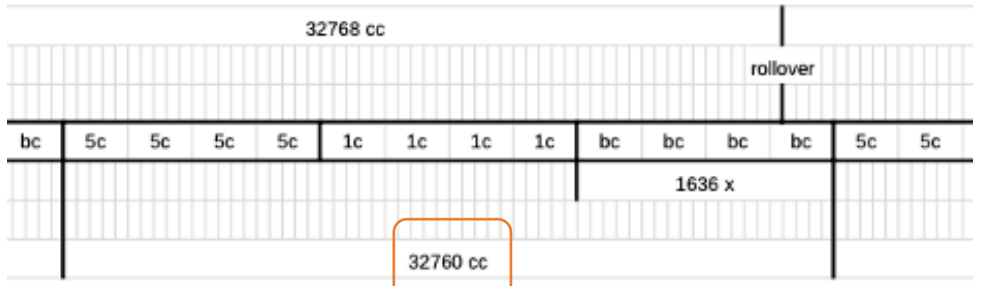
This is indeed consistent with the **10/8 encoding/decoding**!! The **rollover cannot be sent "immediately"** if there is something in the serializer (including an idle 0xbcbcbcbc).



distance between roll-over



time distance on 32-bit bus



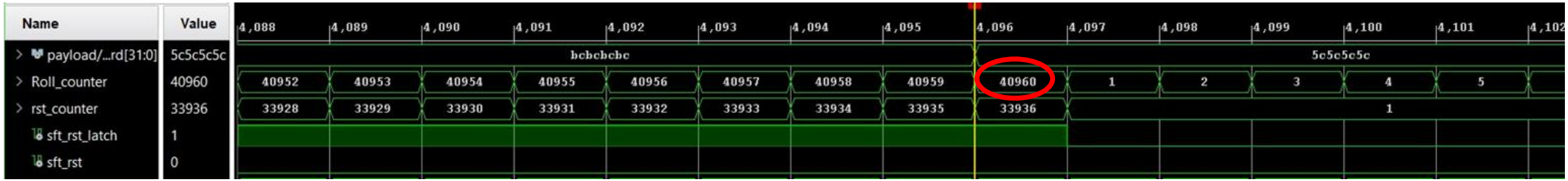
Check: controlling “phase” of SR assertion

- An internal **FPGA 15-bit counter** is reset at each SR or at its own rollover.
- The **SR** is sent to ALCOR **after a fixed number of clock cycles** (16384 and 8192 ccs)
- Result:
  - the time distance between the SR falling edge and the first RO word **floats around ~33945** (strange.. it should be close to 32768, we are still checking **why an offset of more than 1100 cc**). **No small values observed** as in asynchronous case.
  - the **distance between 2 ROs is however fixed** and the value is as expected.

Phase (cc)	RO distance (cc)	Notes
16384	49140 and 49160 (16384 + 32768 = 49152)	2 values due to 10/8 bit DESER
8192	40960 (8192 + 32768 = 40960)	fixed because is a multiple of 20

Phase = 8192 cc

Expected RO distance = 32768 + 8192 cc = **40960 cc**



The difference between 2 rollovers is exactly what we expected

Result: new anomalies

With Smart Reset

soft reset sent at ½ rollover

in this run it looks like the first word was not a rollover

in most chips the frame header is missing and the first word is the frame counter

In chip 2 the first word is the frame header

Lanes data for different chips seem starting incomplete, **without RO**!

In all lanes but one we lose also the frame header.

Result: new anomalies

With Smart Reset

soft reset sent at ½ rollover
anomaly 1740266061

the first word in most chip is a rollover

but in chip 2 there is a missing rollover and the first word is a frame header

test pulse is aligned in the same frame

All lanes but one seem starting complete!
One lane loses the rollover.

Final check and conclusions/questions

Final check: using again SR at ½ rollover,

we verified again **through a ILA IP core set** after deserialization inside the FPGA firmware that the **first word non IDLE after a SR** is **always** a **rollover**!

Current conclusions/questions:

1. We still need to investigate what happens with SR sent in different positions of the coarse counter range, and specifically very “close” to normal rollover. **One of the tests done (slide 7) might indicate we have instability regions.**

2. Monitoring ALCOR output just after deserialization: tests indicate **no RO are lost after SR**.

3. However **data anomalies are seen in the data** even controlling when the SR is sent (slide 10, 11) at **half of ½ rollover**. We will **move debug on FPGA and readout** to understand where they are generated!

Backup

ALCOR Soft Reset and EIC orbit

Current scheme:

- Input clock: 98.52 MHz → 394.08 MHz ($ALCOR_{clk}$ = 4 x EIC_{clk})
- 1 RevTick (“Revolution Tick”) orbit counter/bunch reset every EIC_{clk} 1260 ticks = 12.789 us
- → 1 RevTick (Soft Reset) every 5040 $ALCOR_{clk}$
- SR would reach ALCOR @ 78.2 kHz

Jamming:

Jamming sequence

- program KC705
- initialise ALCOR chips
  - make sure that ECCR *"Column 0 Enable"* and *"Column 1 Enable"* bits stay zero
  - needed to modify programs for that, not big issue
- turn ON the Jamming pulser
- set KC705 in SPILL mode (mode = 0x5)
  - this will trigger a SOFT RESET at each SPILL signal
  - ALCOR chips are "jammed", frame counters do not increase
- enable columns in ECCR
  - manually set ECCR *"Column 0 Enable"* and *"Column 1 Enable"* to one
- reset KC705 SPILL mode (mode = 0x0)
  - stop sending SOFT RESET signals
  - ALCOR chips are "unjammed" all together, frame counters start increasing
- turn OFF the Jamming pulser
- ready to collect data with synchronised frames