Status report on common simulation (and firmware) framework (for mini-DAQ) + points for discussion…

LHCb Electronics Upgrade Meeting

10 October 2013

Federico Alessio, Guillaume Vouters

What is this? (I)

Common simulation framework

to:

develop code for TELL40

(including sub-detectors' user code)

develop code for TFC

and

study resource usage

What is this? (II)

BUT ALSO

to develop FE logic by verifying its compatibility with the rest of the system (TFC and TELL40) and study resources usage

FE digital logic and packing algorithms/protocols should be validated in simulation in its proper environment!
This means: with TFC and TELL40 + (LLI and DAQ).

This framework is also for you!

General philosophy in our simulation endeavour

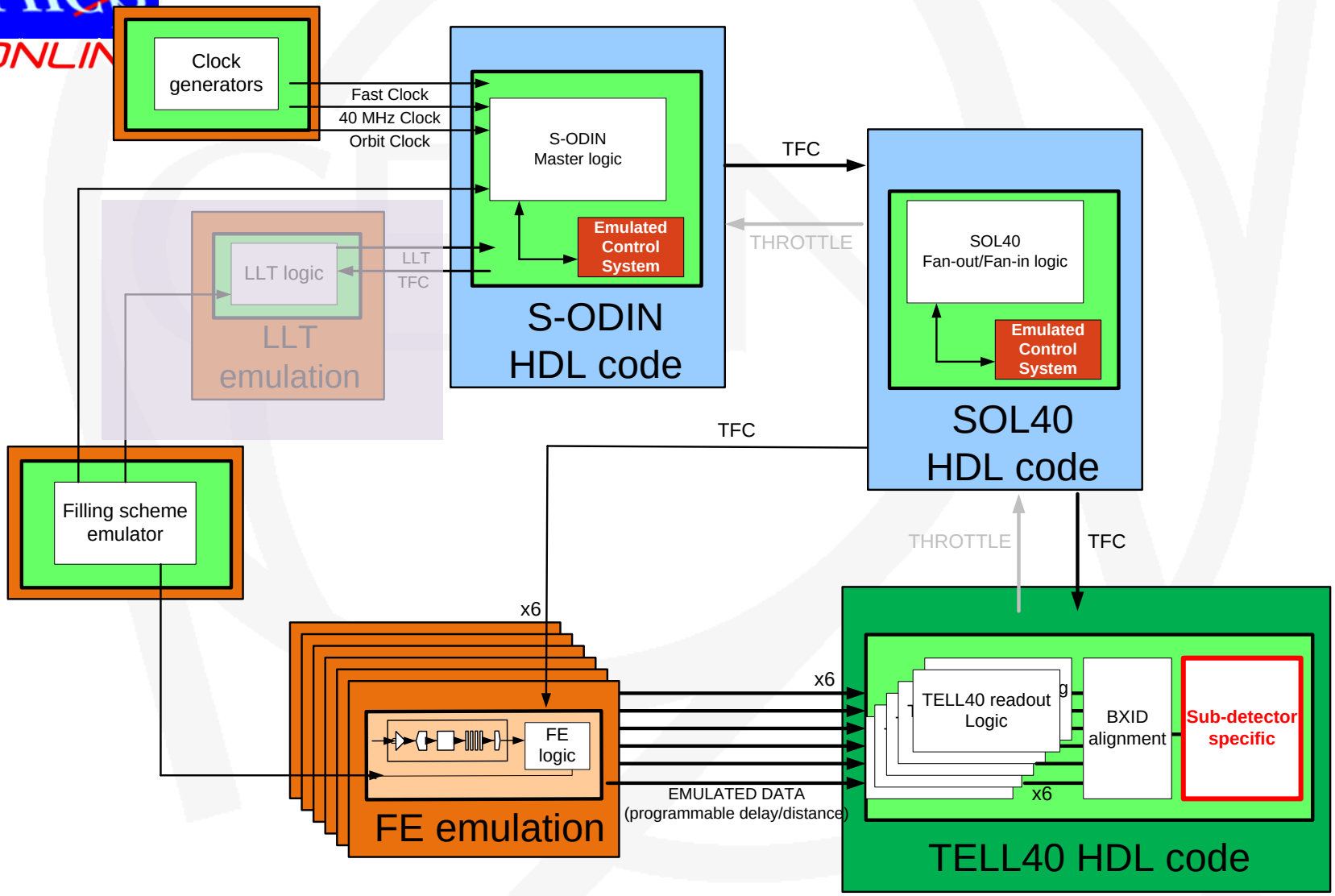
Configurable with many different parameters

Possibility to **change and modify** files locally and globally

Easy, reliable and controlled work environment

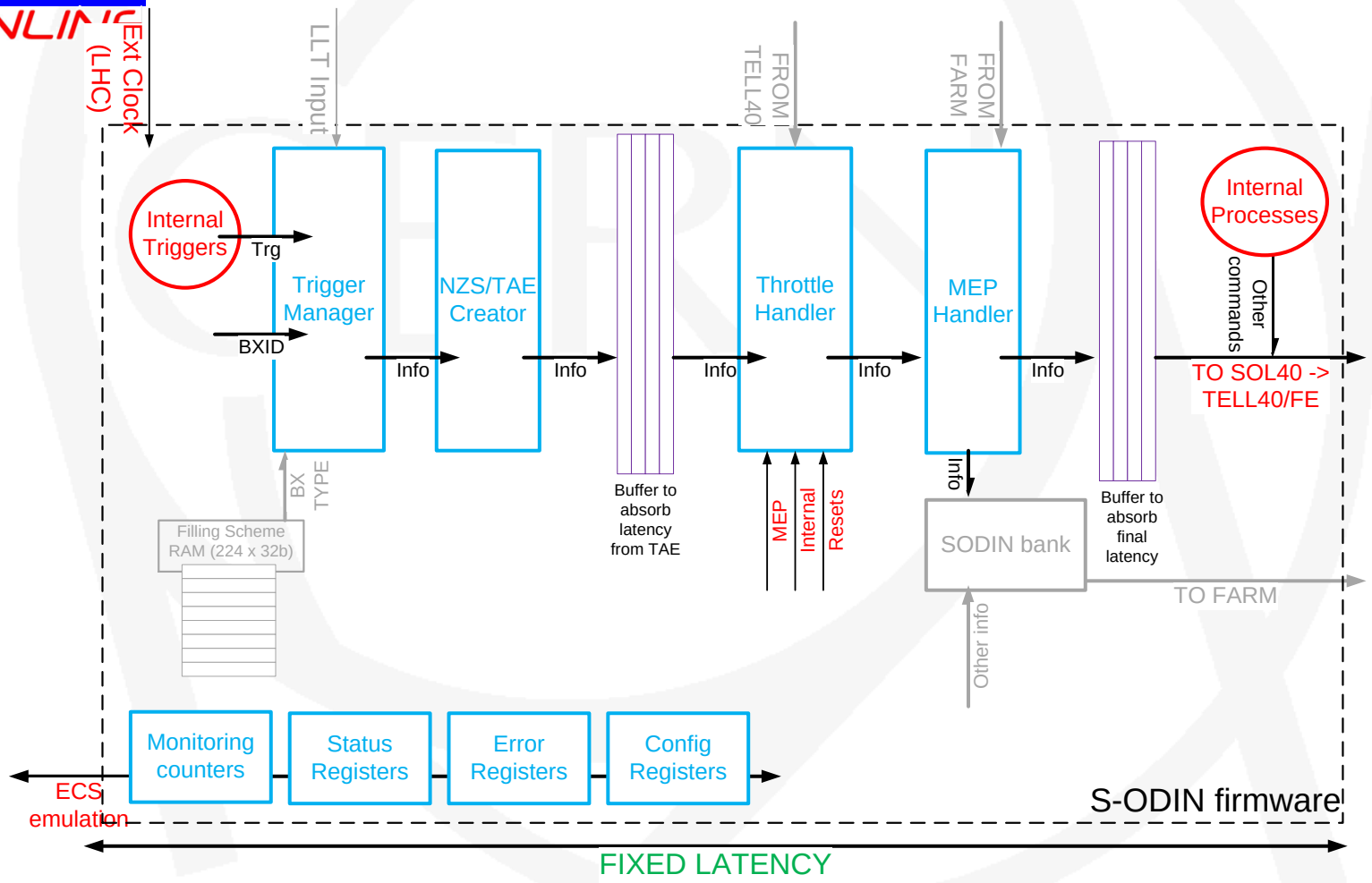
Define and study **needed resources**

What we have today



S-ODIN HDL code

(this is not an emulation, this is real and synthesizable code)



For details, see [LHCb-PUB-2012-001]

Fast commands available

(this is not an emulation, this is real and synthesizable code)

to TELL40

63 .. 52	51	50	49 .. 18	17 .. 14	13 .. 10
BXID(11..0)	Reserve	MEP Accept	MEP Dest(31..0)	Trigger Type(3..0)	Calibration Type(3..0)

9	8	7	6	5	4	3	2	1	0
Synch	Snapshot	Trigger	BX Veto	NZS Mode	Header Only	BE Reset	FE Reset	EID Reset	BXID Reset

to FE

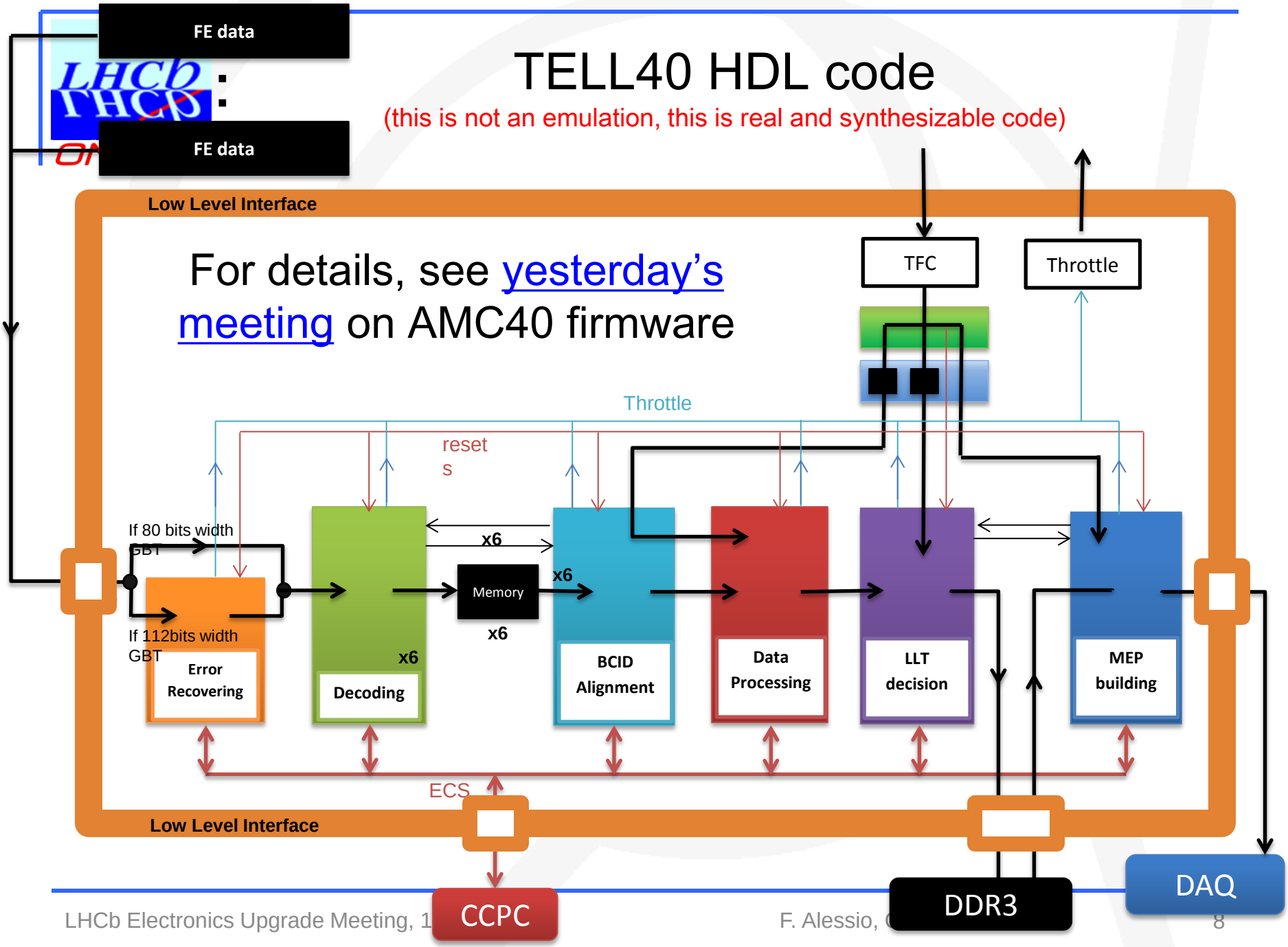
23 .. 12	11	10	9	8 .. 5	4	3	2	1	0
BXID(11..0)	Reserve	Synch	Snapshot	Calibration Type(3..0)	BX Veto	NZS Mode	Header Only	FE Reset	BXID Reset

Periodicity, rates, codes are all configurable
(just like in the current TFC!)

For details, see LHCb-PUB-2012-017

TELL40 HDL code

(this is not an emulation, this is real and synthesizable code)



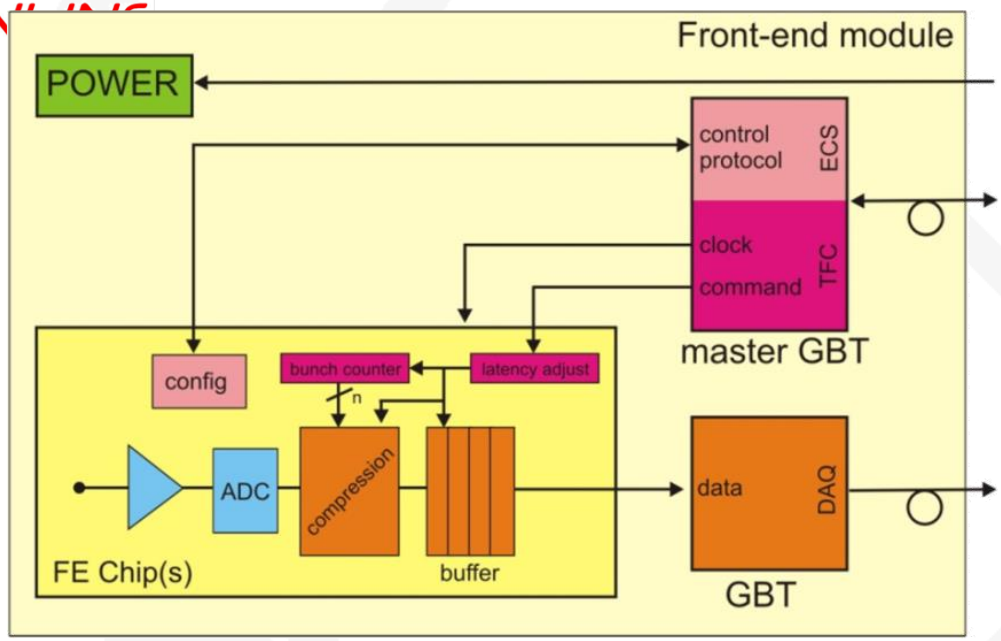
FE emulation

We needed some emulation code for the FE to test our own

- ✓ TFC commands and TFC logic make sense
- ✓ TELL40 decoding and BXID alignment
- ✓ Estimate needed resources at FE, TELL40, TFC
  - Study alternative solutions
  - Compilation performed for Stratix V

For details, see LHCb-INT-2013-015

Reminder: your (generic) FE



NO TRIGGER to FE!
→ Only commands, clock and slow control

For details, see
LHCb-INT-2011-011

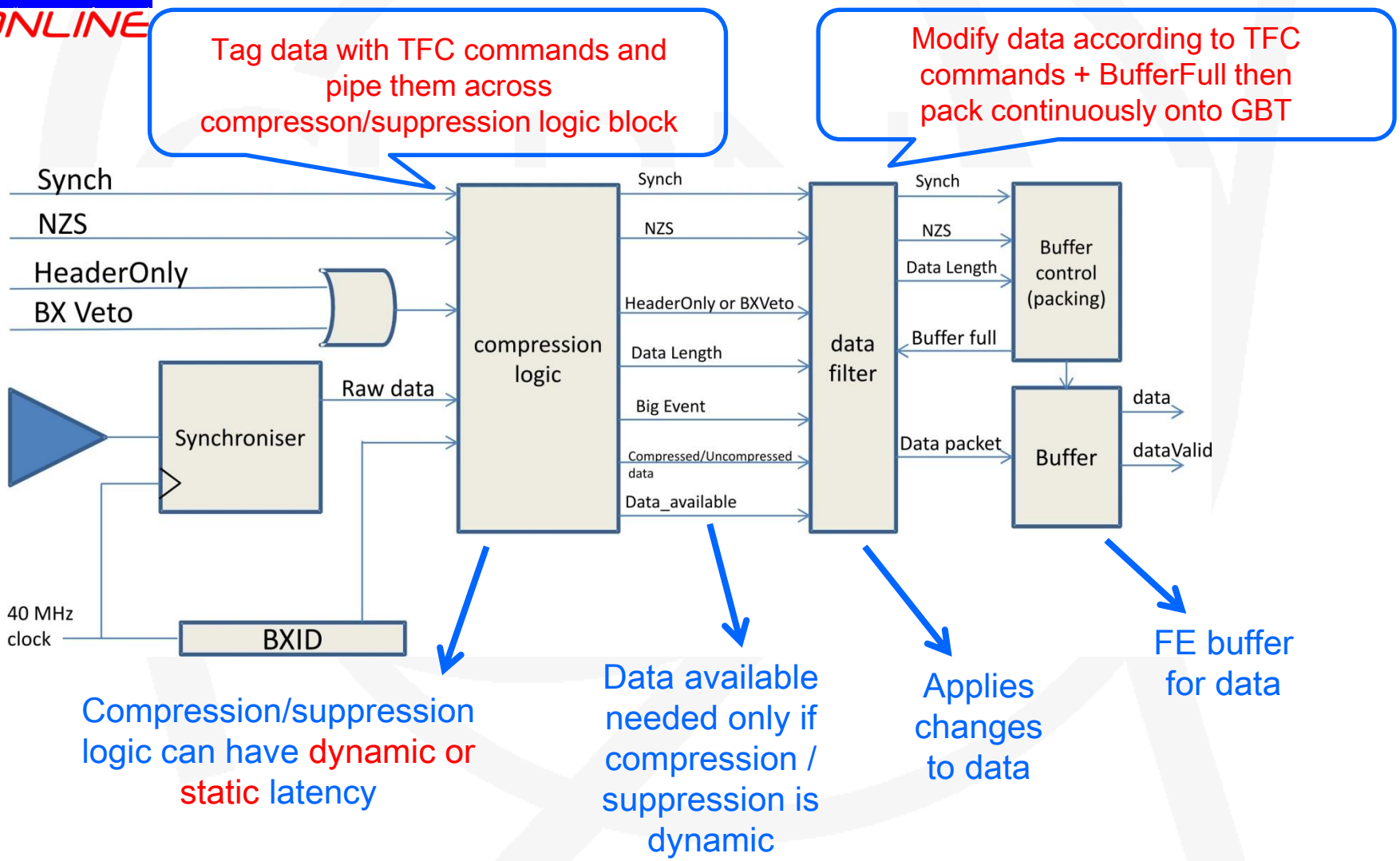
Compress (zero-suppress) data already at the FE

- reduce # of links from ~80000 to ~12500 (~20 MCHF to ~3.1 MCHF)
- data driven readout (asynchronous) + variable latencies!

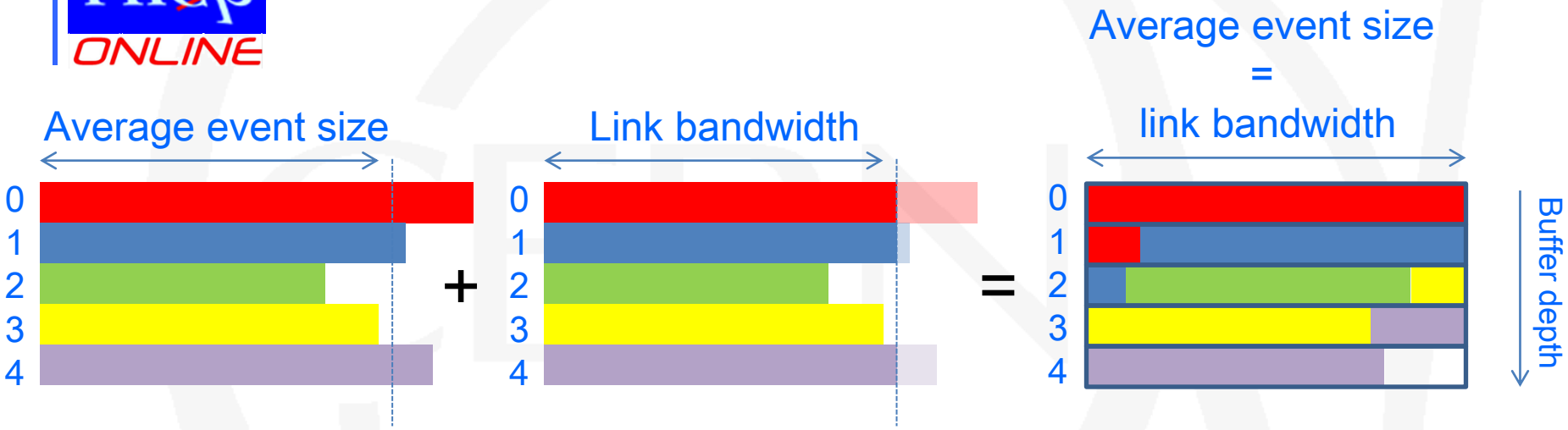
Efficiently use data link bandwidth

- pack data on data link continuously with elastic buffer
- extensive use of GBT (robust FEC vs WideBus mode)
  - ✓ evaluate choices based on complexity vs robustness

Reminder: generic FE data flow scheme

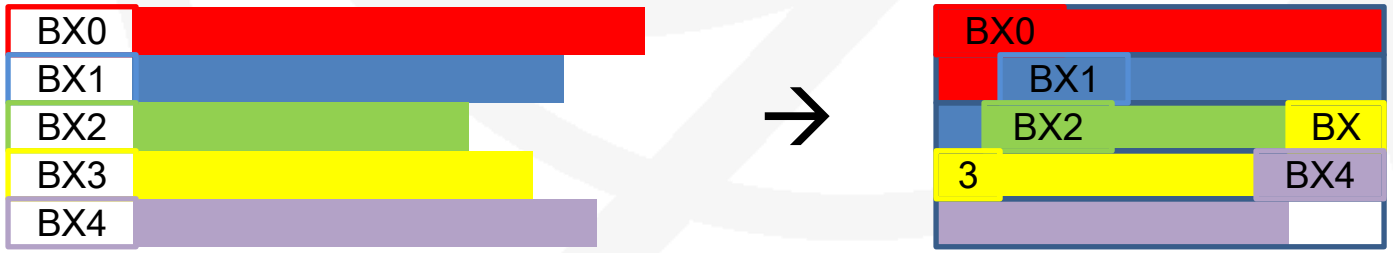


Reminder: dynamic packing algorithm

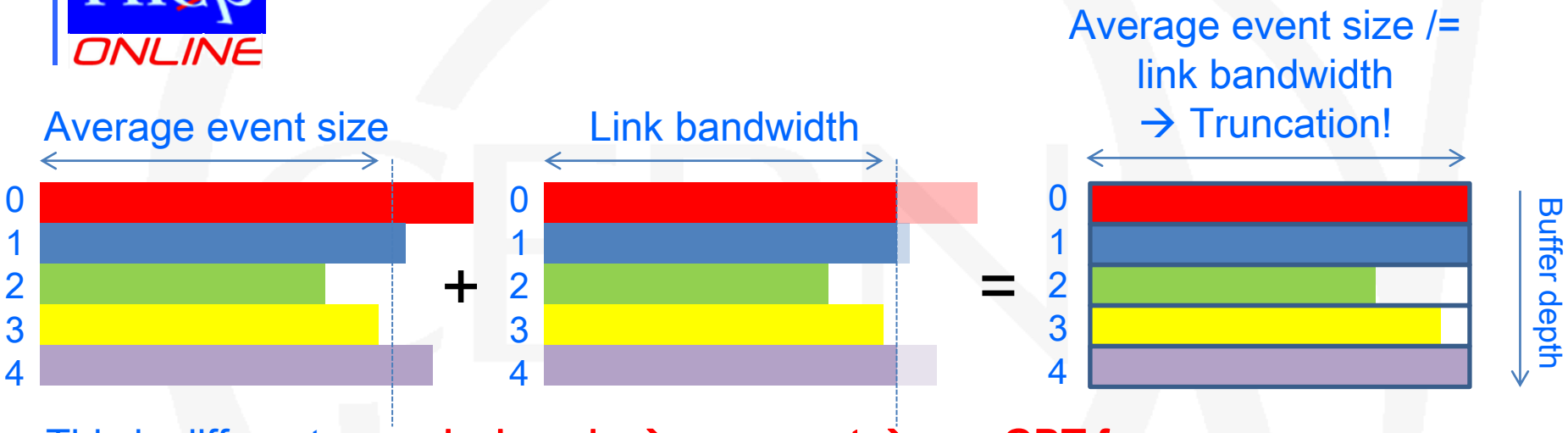


Header is the unique identifier for each event in frame

- ✓ Compulsory (tag for each crossing), partly programmable (must contain length of frame+BXID)
- ✓ Difficult buffer management, but almost no truncation.
- ✓ Flexible against occupancy problem (what if your estimate is wrong?).
- ✓ Maximum exploitation of bandwidth.
- ✓ Readout Board uses Header to decode and separate frames → lots of resources.

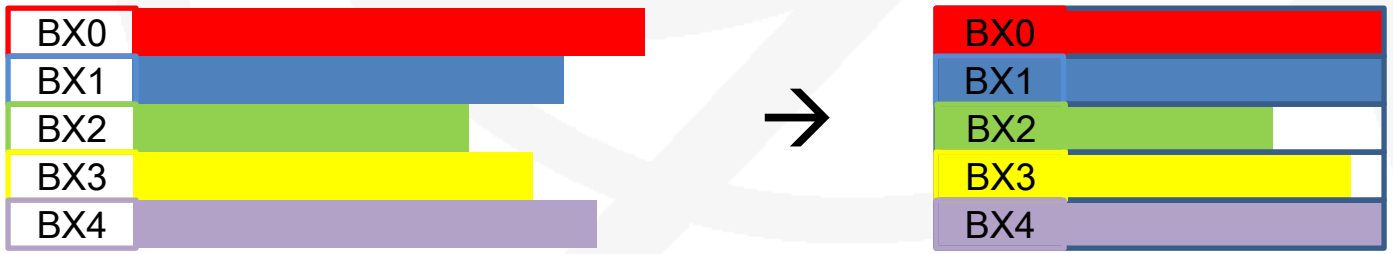


Reminder: fixed packing algorithm



This is different: **one clock cycle → one event → one GBT frame**

- ✓ Header more flexible: you can add addresses, hitmaps…
- ✓ Very simple buffer management, but truncation has to happen eventually.
- ✓ Not flexible against occupancy problem (again, what if your estimate is wrong?).
- ✓ Loses a bit of bandwidth as empty spaces must be padded to be sent out.
- ✓ Readout Board uses Header to decode and separate frames → much fewer resources



Reminder: fixed vs variable length header in dynamic packing

Dynamic packing with fixed length header.

Header field		
BXID	**information**	**Length**
BXID [11:0]	X bits	Y bits

Dynamic packing with dynamic length header (fully flexible!)

Synch	**Header Only or BX Veto or BufferFull**	**NZS**	**Output of data filter**			
			Header field			**Data field**
			BXID	**No data**	**Length**	
1	X	X	BXID [11:0]	X	X	Synch pattern
0	1	X	BXID [n:0]	1	X	No data
0	0	1	BXID [n:0]	0	NZS code (unique)	Uncompressed data
0	0	0	BXID [n:0]	0	Data length [m:0]	Compressed data

What we have for the FE emulation

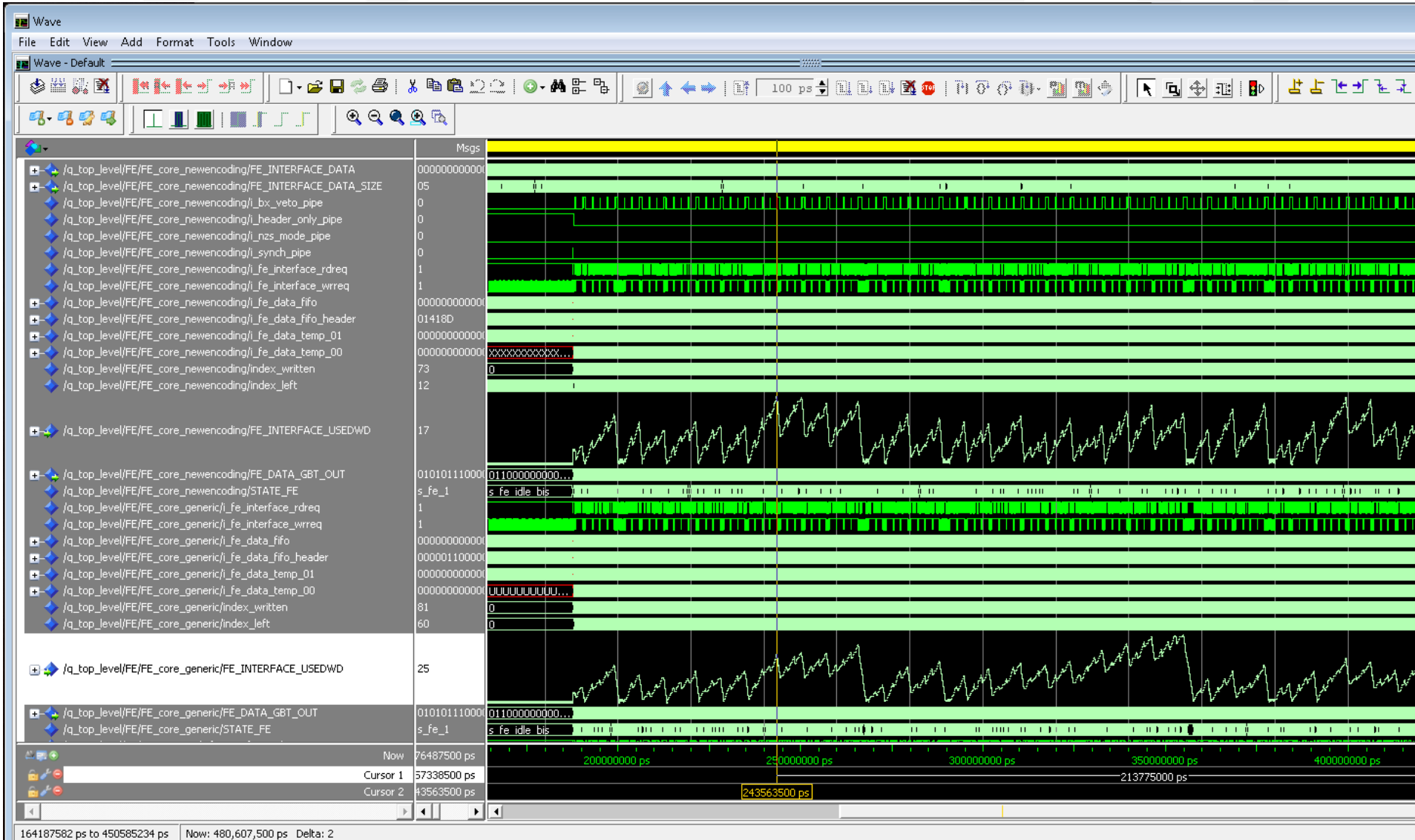
Implemented generic code for FE encoding/digital logic:

✓ Programmable
- Number of channel and size of channels
- Buffer depth
- GBT width frame (80 or 112 bits)
- Header fields
- Introduce bugs in a controlled way
  - skip BXID, swap BXID etc…

✓ Synthesizable
- Estimate resources in FE

Can emulate ANY FE packing algorithm (I mean it)!

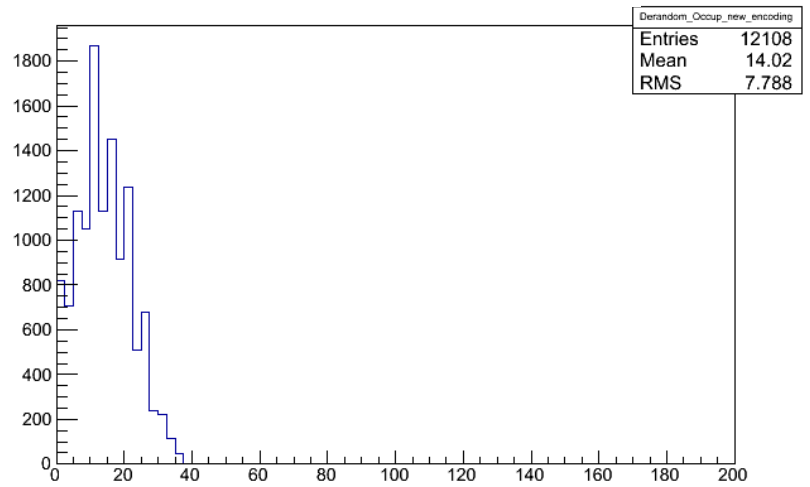
Studied differences in efficiency



Studied differences in efficiency

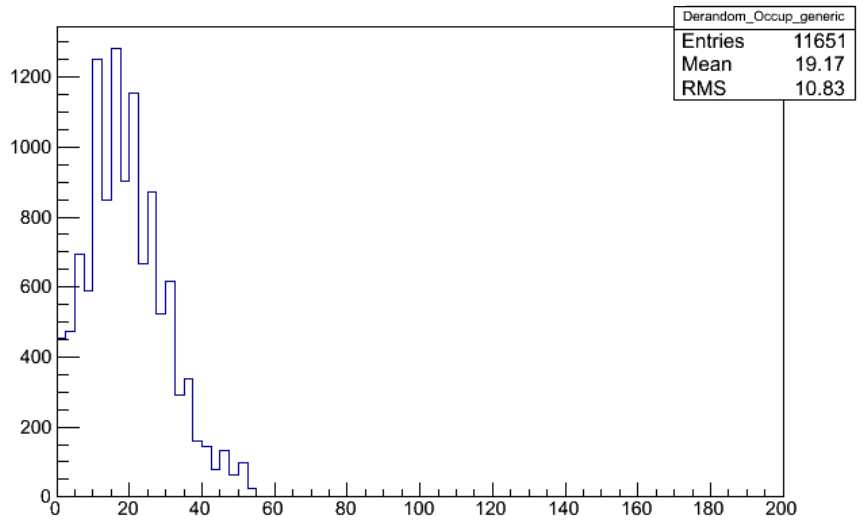
This is just an example:
500 channels of 4 bits each, occupancy 3.5%, buffer depth 160, 12 bits of BXID

Dynamic with dynamic header



Buffer occupancy over 500 us

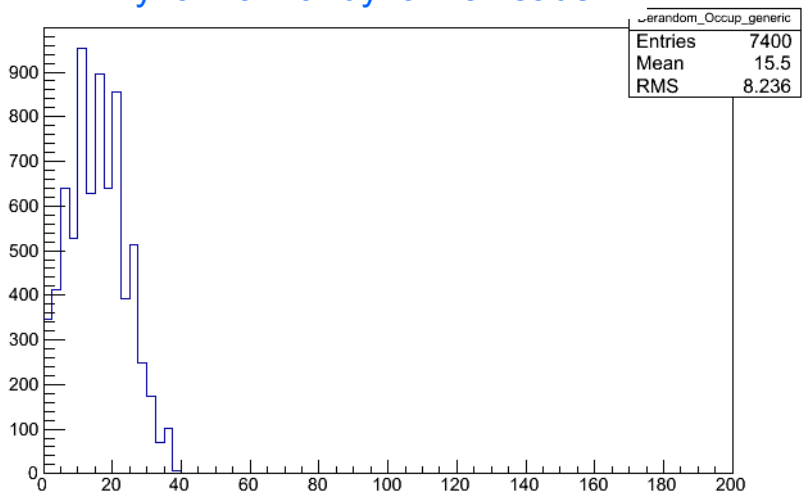
Dynamic with fixed header



Studied differences in efficiency

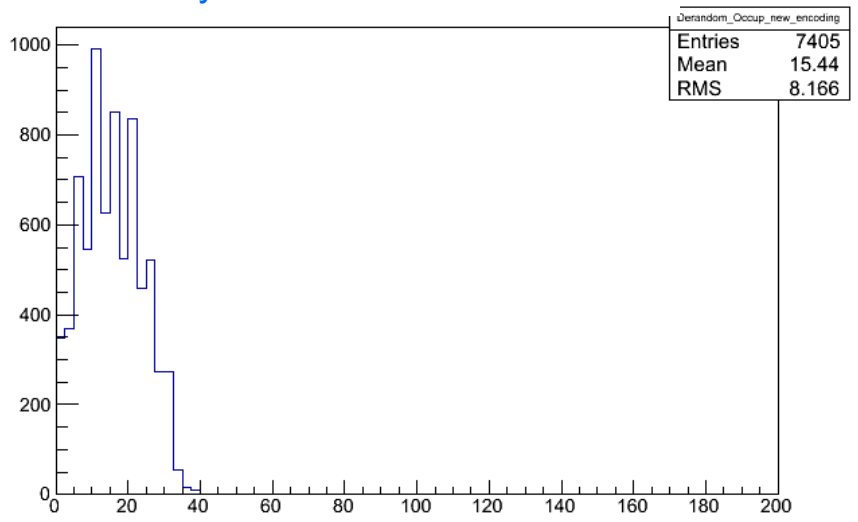
This is just another example:
500 channels of 4 bits each, occupancy 4%, buffer depth 160, 4 bits of BXID

Dynamic with dynamic header

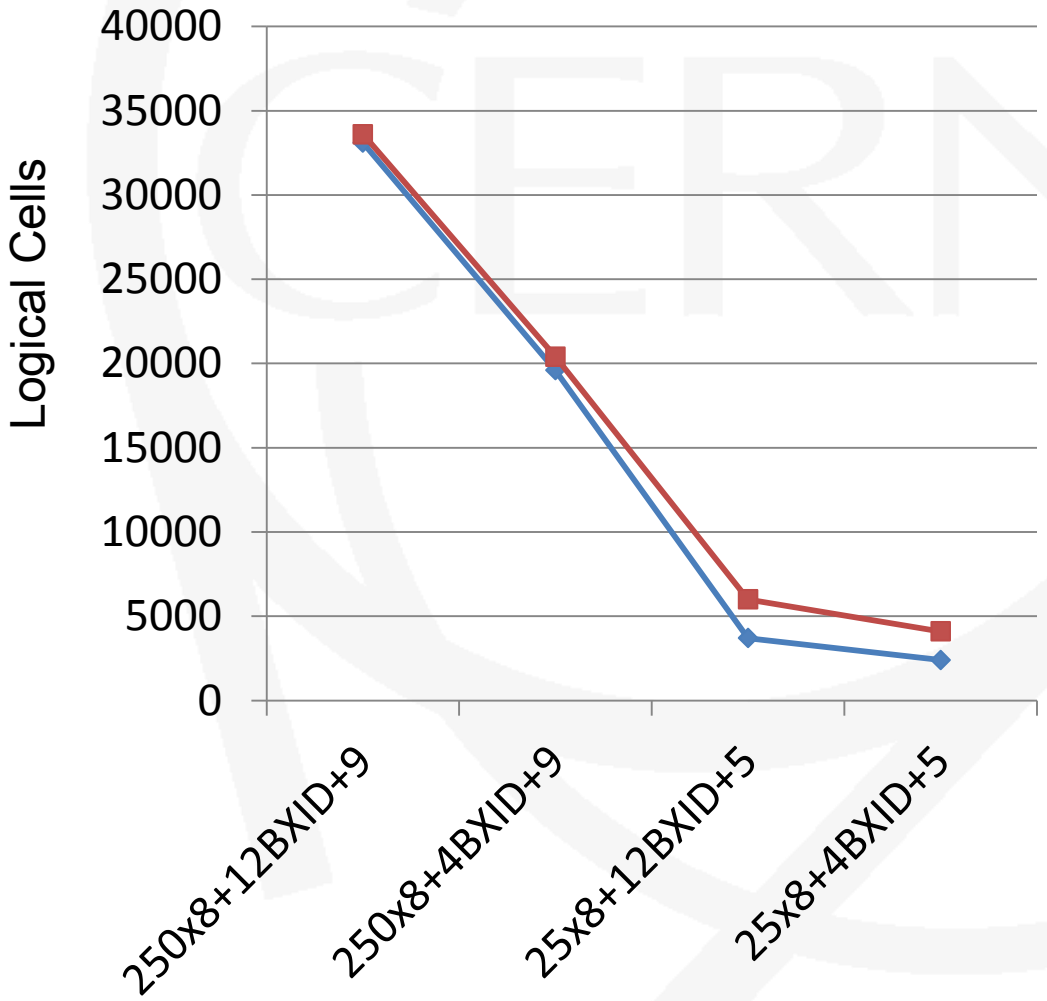


Buffer occupancy over 500 us

Dynamic with fixed header



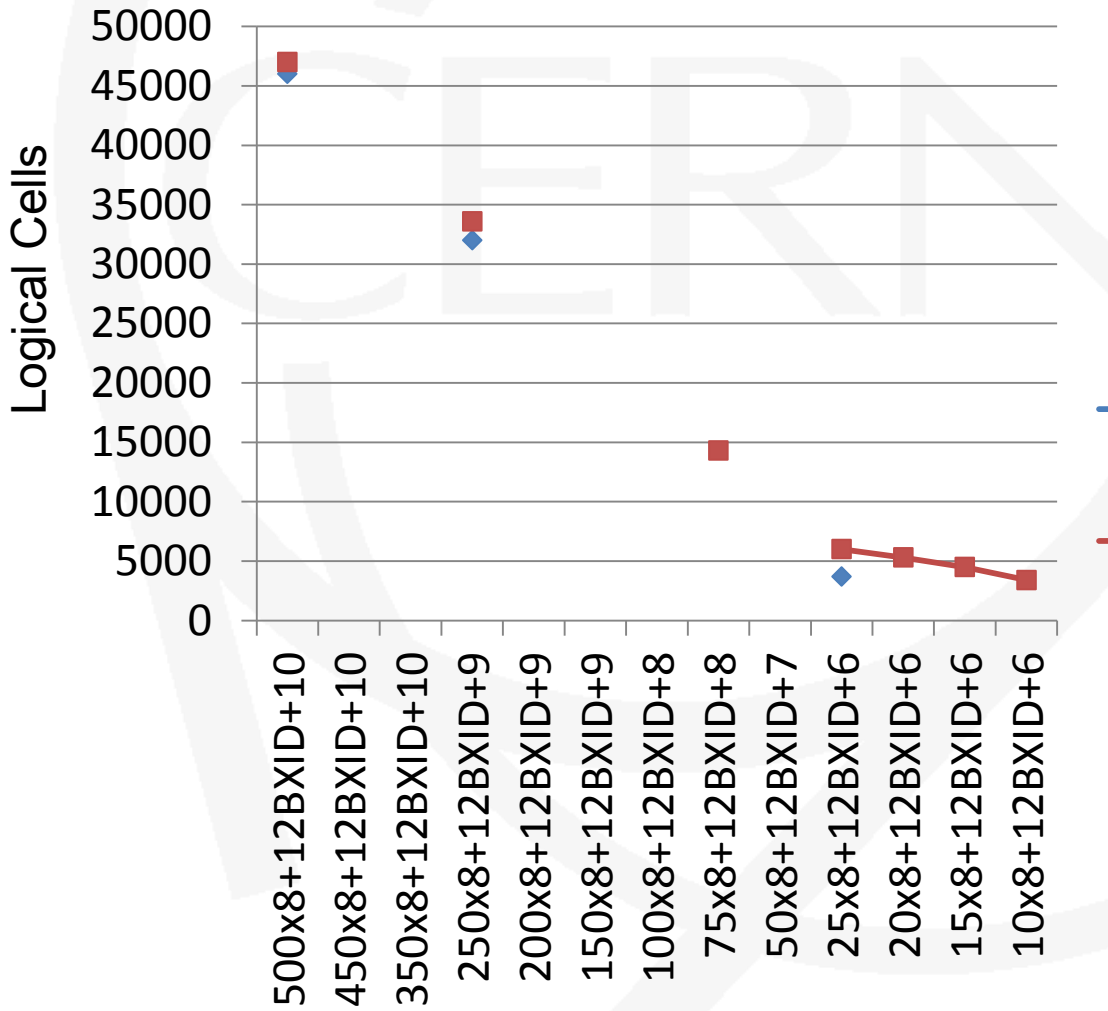
Compared resources needed for different encodings



This is for the ENCODING.
This is per GBT link!

Dynamic encoding might help you save in fibers, but the cost will rise in FPGA/ASICs resources!

Compared resources needed for different encodings



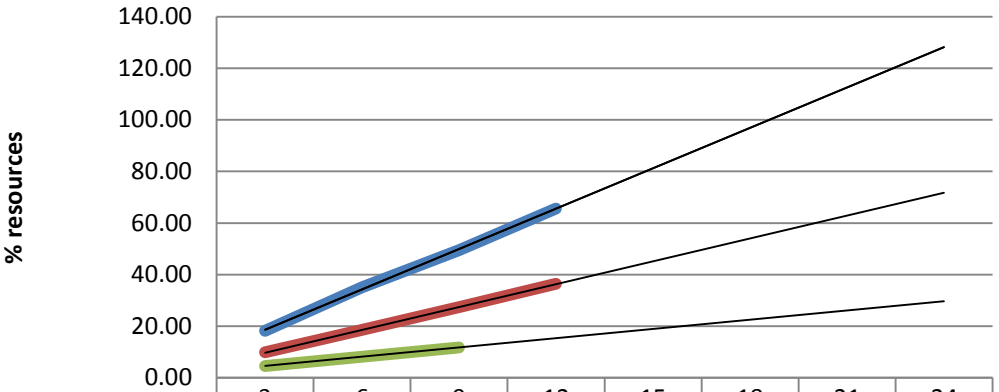
This is for the ENCODING.
This is per GBT link!

NB: fixed encoding is 460 LC!
→ 10-100x less
→ CALO & MUON use case - they need fixed latency for the LLT!

Studied impact on TELL40 resources

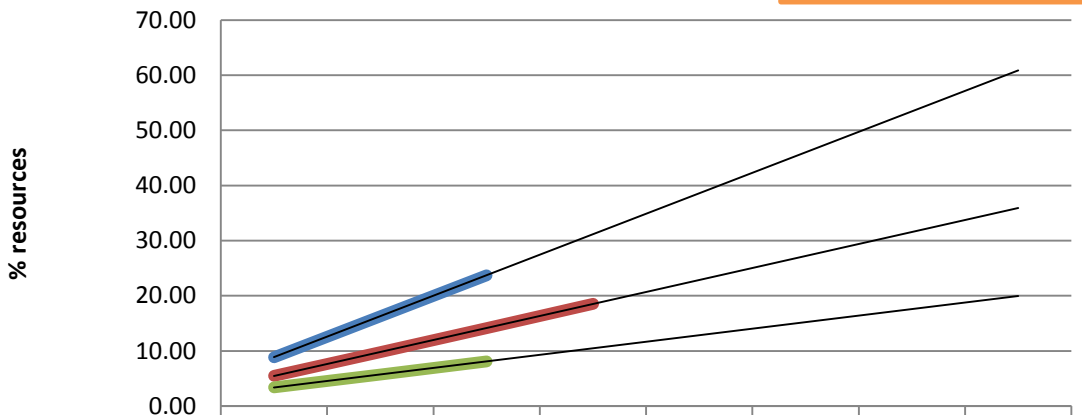
This is for the DECODER in TELL40.

Dynamic with dynamic header



	3	6	9	12	15	18	21	24
GBT 120 bits	18.20	35.07	49.48	65.55				
GBT 80 bits	9.87	18.57	27.34	36.47				
GBT 40 bits	4.59	8.16	11.77					

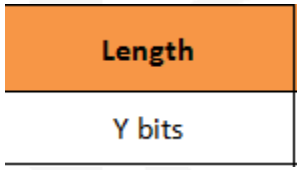
Dynamic with fixed header



	3	6	9	12	15	18	21	24
GBT 120 bits	8.86	16.29	23.73					
GBT 80 bits	5.49	9.77	14.16	18.53				
GBT 40 bits	3.38	5.72	8.12					

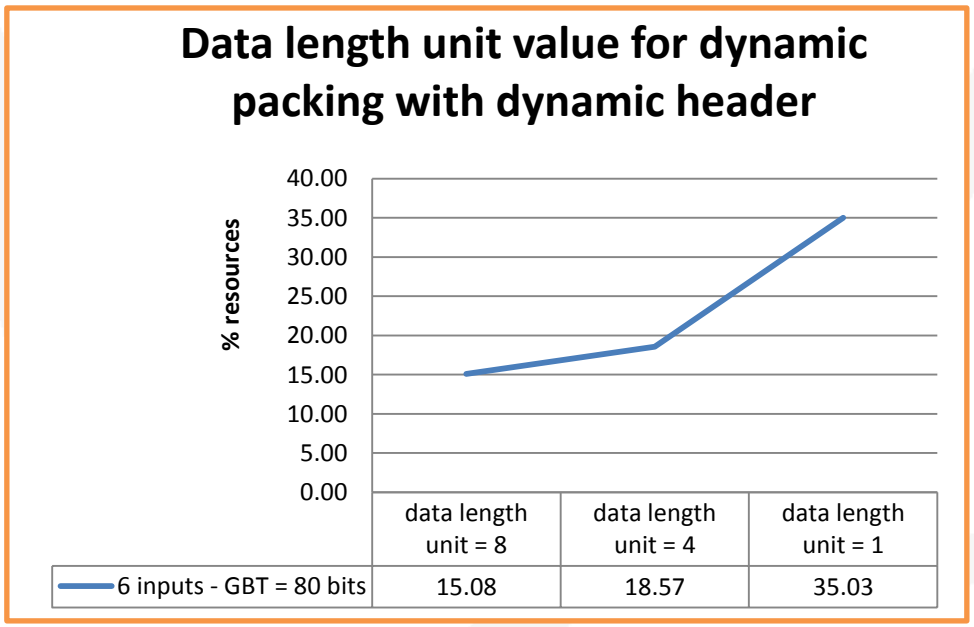
Studied impact on TELL40 resources

Length field will likely contain the number of channels hit
(not the length of the data word – that would require more bits)
Each channel has a "data length unit value" (i.e. size of each channel)



Ex: Length (8 bits) is 0x0A = 10
If data length unit value = 1 : real data length = 10bits
If data length unit value = 4 : real data length = 40bits
If data length unit value = 8 : real data length = 80bits

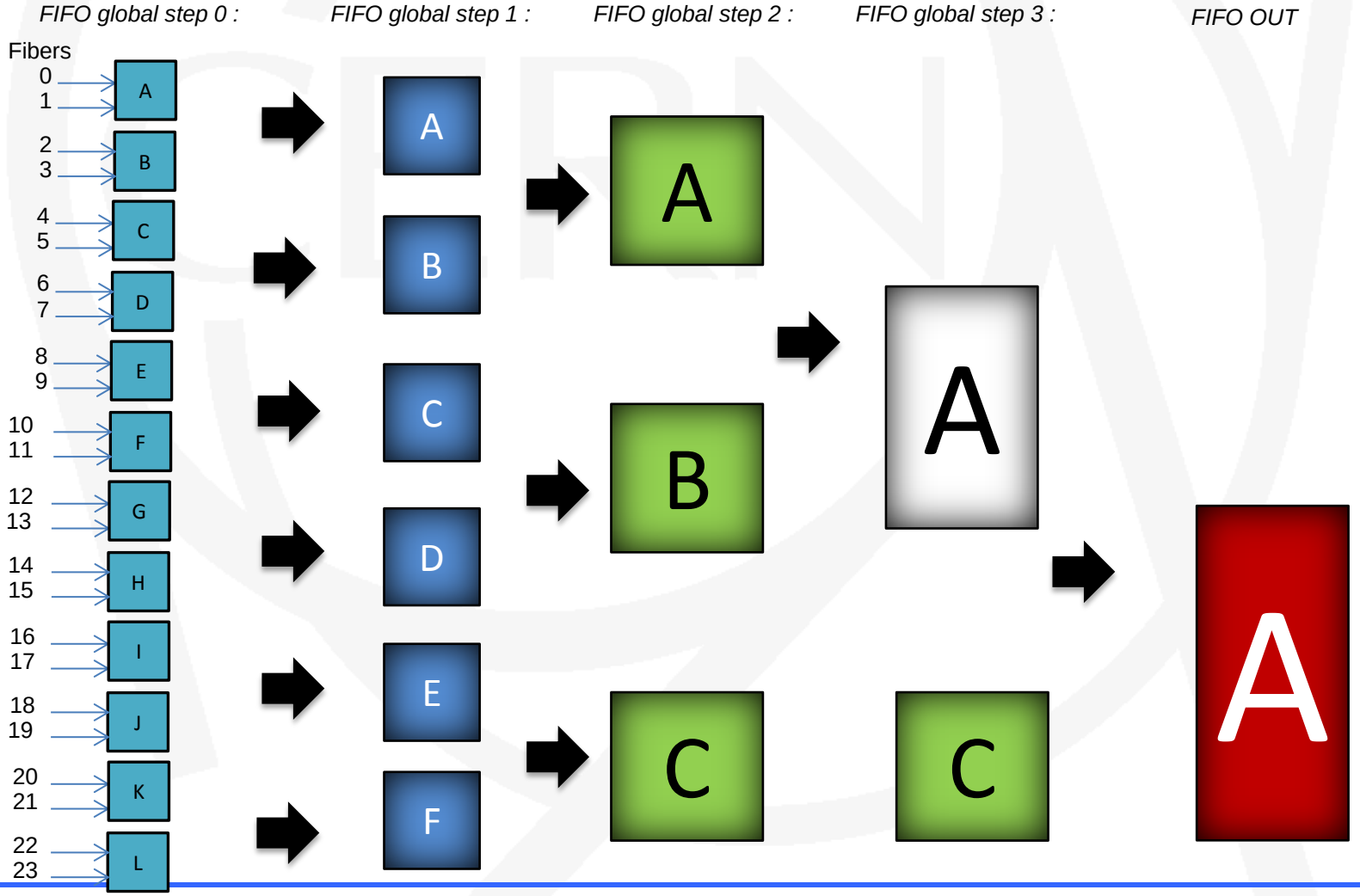
Test done with dynamic packing with dynamic header



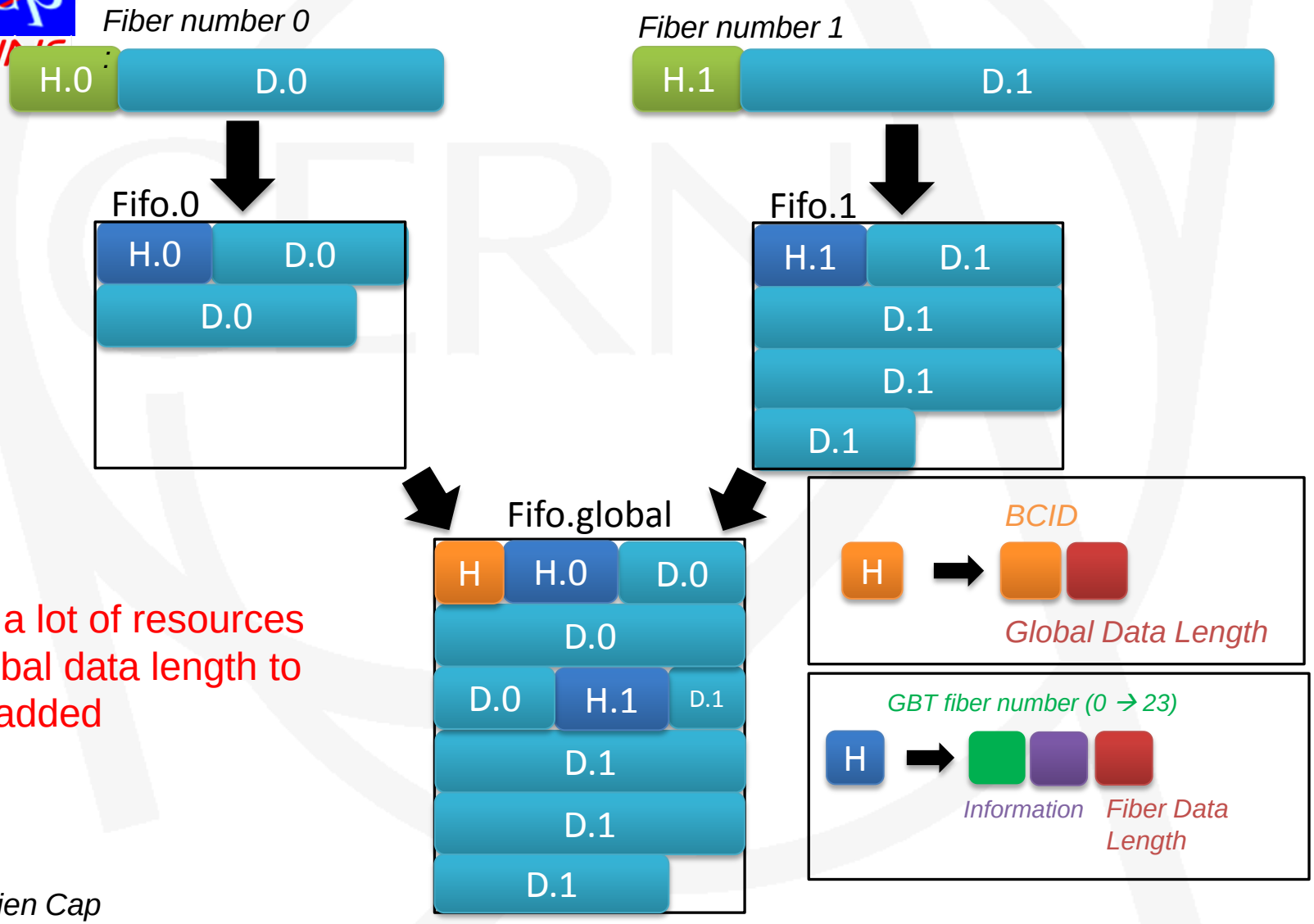
The data length unit value should be bigger or equal to 4.

We should forbid smaller than 4.

Studied impact on TELL40 resources



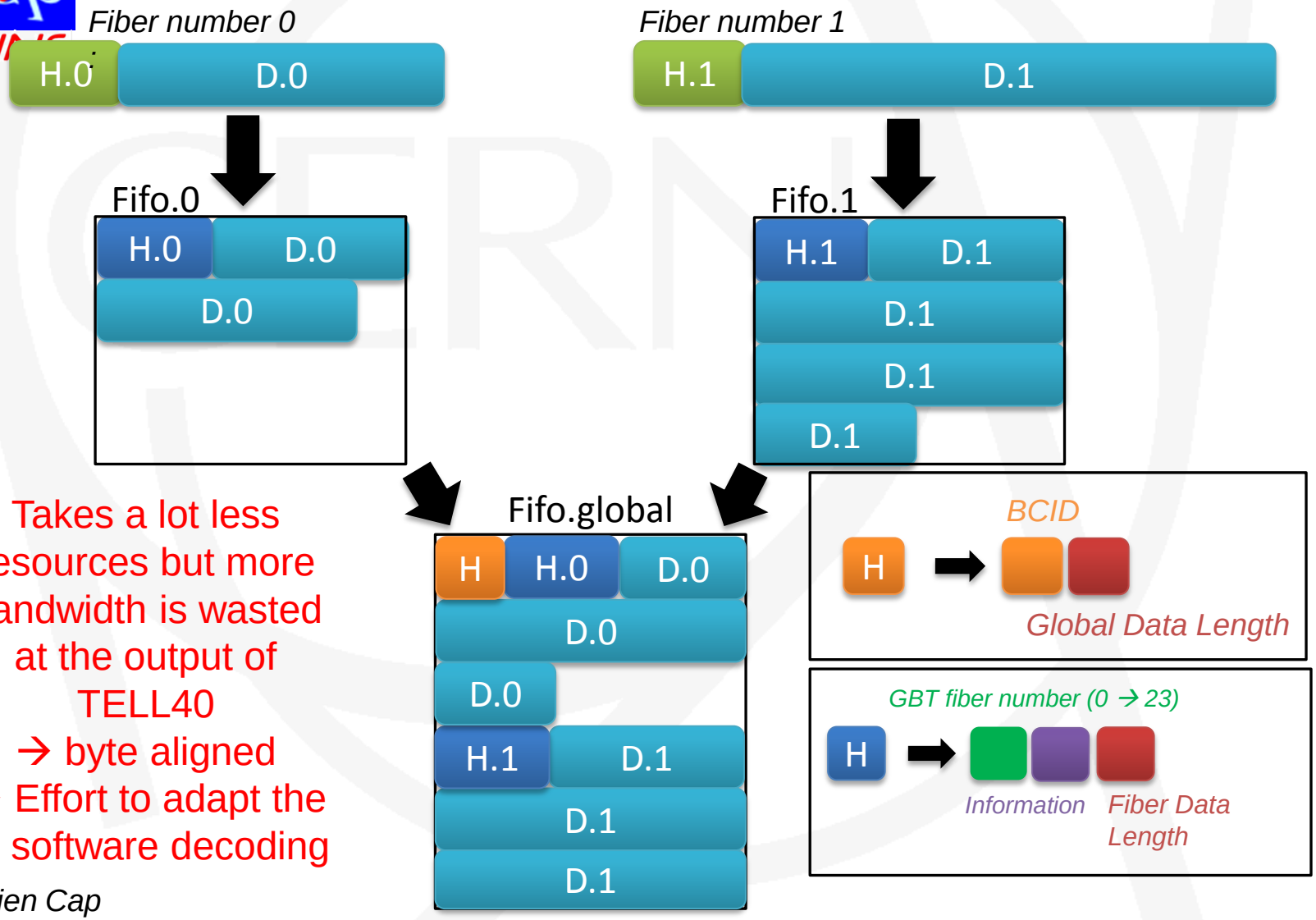
Studied impact on TELL40 resources



Takes a lot of resources
→ Global data length to be added

Sebastien Cap

Studied impact on TELL40 resources



Takes a lot less resources but more bandwidth is wasted at the output of TELL40
→ byte aligned
→ Effort to adapt the software decoding

Sebastien Cap

Is that all?

Not yet…

We have an easy, reliable and robust way to start the simulation and share files/components

- ✓ Scripts to launch simulation environment (and compilation) in ModelSim, start up top modules, include entities, make connections between components, make it easy for the users
- ✓ LHCb upgrade GIT repository to share files, track versions … (see yesterday's meeting for more information) https://indico.cern.ch/conferenceDisplay.py?confId=254741
- ✓ FORGE website to share documentation … https://lbredmine.cern.ch/projects/amc40/documents

What's next?

We will make it available by the next electronics meeting to sub-detectors to start developing and test their code!

(just in time for Christmas)

Firmware for Mini-DAQ

Compilation of first TFC + TELL40 firmware

Preparation of simulation and compilation framework

Done!

Integrate LLI and DAQ core

Meeting in preparation, by November

(Basic) software to control Mini-DAQ

Meeting next week, ready by Xmas

Test & deploy in a (virtual) gift package

(for the hardware, see Jean-Pierre's presentation)

Before concluding

FE emulators are there only to help us develop our code…

… you will be requested to test and insert your own FE code in the simulation framework …

… you will be requested to test and insert your own TELL40 “user data processing code” in the simulation framework as well.

(We simply eased your job by having a reliable simulation environment).

Conclusion

Lots of progress over the summer:

- ✓ TELL40 and TFC firmware ready
- ✓ DAQ core is ready (Paolo)
- ✓ LLI is ready (Jean-Pierre et al.)

} to be put together now.

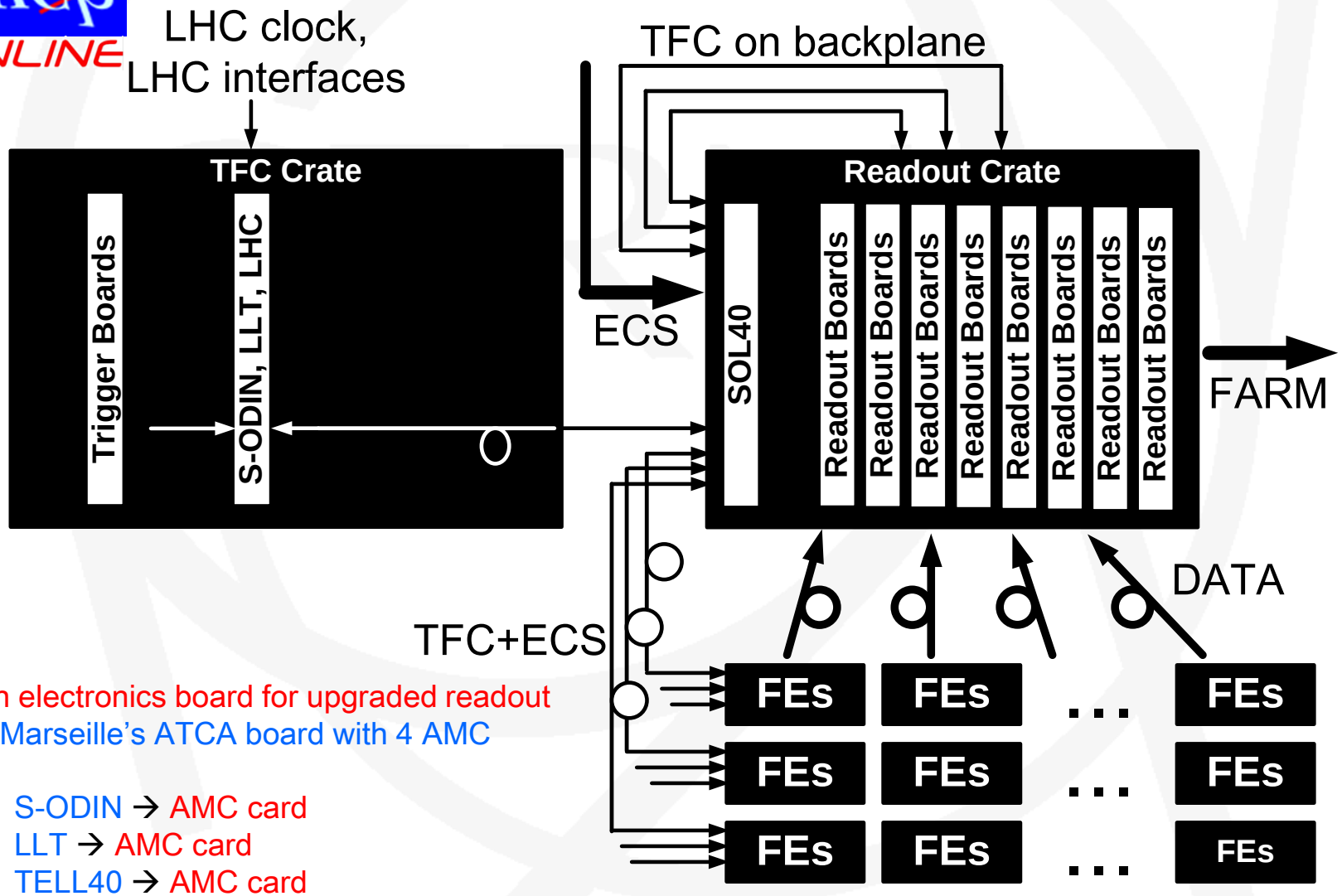
- ✓ Extensive simulation framework basically ready
- ✓ First compilation attempt for Mini-DAQ successful
- ✓ More studies will be done on resource usage and algorithms and will be circulated.

If you have requests, please get in contact with us.

However, you will surely hear from us (beware)… ☺

Qs & As?

The upgraded physical readout slice



Common electronics board for upgraded readout system: Marseille's ATCA board with 4 AMC cards

- S-ODIN → AMC card
- LLT → AMC card
- TELL40 → AMC card
- LHC Interfaces → specific AMC card

Latest S-TFC protocol to TELL40

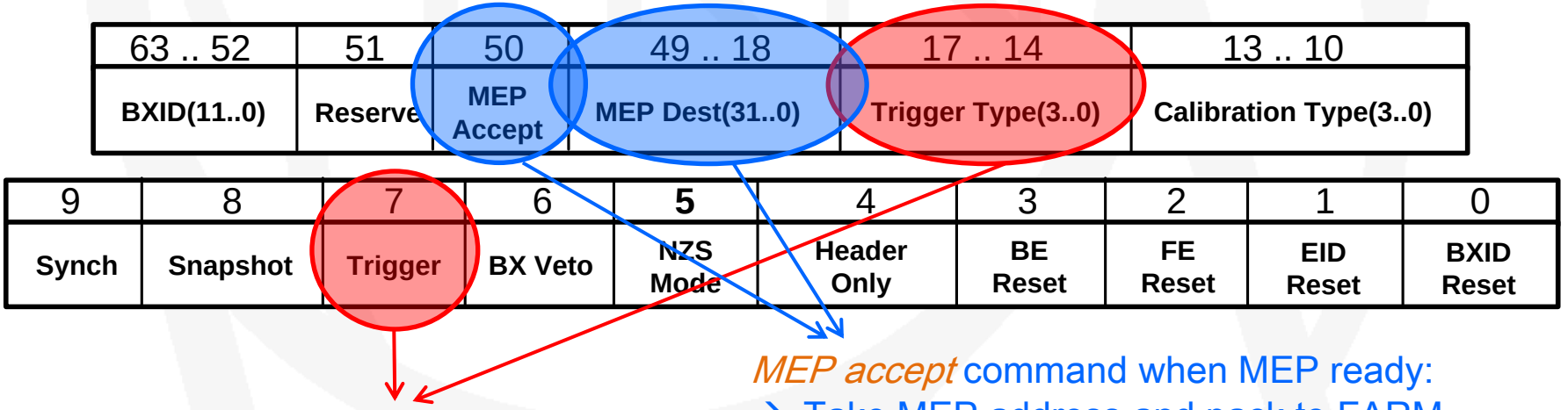
We will provide the TFC decoding block for the TELL40: VHDL entity with inputs/outputs

✓ «Extended» TFC word to TELL40 via SOL40:

→ 64 bits sent every 40 MHz = 2.56 Gb/s (on backplane)

→ packed with 8b/10b protocol (i.e. total of 80 bits)

→ no dedicated GBT buffer, use ALTERA GX simple 8b/10b encoder/decoder



MEP accept command when MEP ready:

→ Take MEP address and pack to FARM

→ No need for special address, dynamic

Constant latency after BXID

✓ THROTTLE information from each TELL40 to SOL40:

- no change: 1 bit for each AMC board + BXID for which the throttle was set

→ 16 bits in 8b/10b encoder

→ same GX buffer as before (as same decoder!)

S-TFC protocol to FE, no change

- ✓ TFC word on downlink to FE via SOL40 embedded in GBT word:
  - → 24 bits in each GBT frame every 40 MHz = 0.98 Gb/s
  - → all commands associated to BXID in TFC word

23 .. 12	11	10	9	8 .. 5	4	3	2	1	0
BXID(11..0)	**Reserve**	**Synch**	**Snapshot**	**Calibration Type(3..0)**	**BX Veto**	**NZS Mode**	**Header Only**	**FE Reset**	**BXID Reset**

Put local configurable delays for each TFC command

- GBT does not support individual delays for each line
- Need for «local» pipelining: detector delays+cables+operational logic (i.e. laser pulse?)
  - → DATA SHOULD BE TAGGED WITH THE CROSSING TO WHICH IT BELONGS!

TFC word will arrive before the actual event takes place

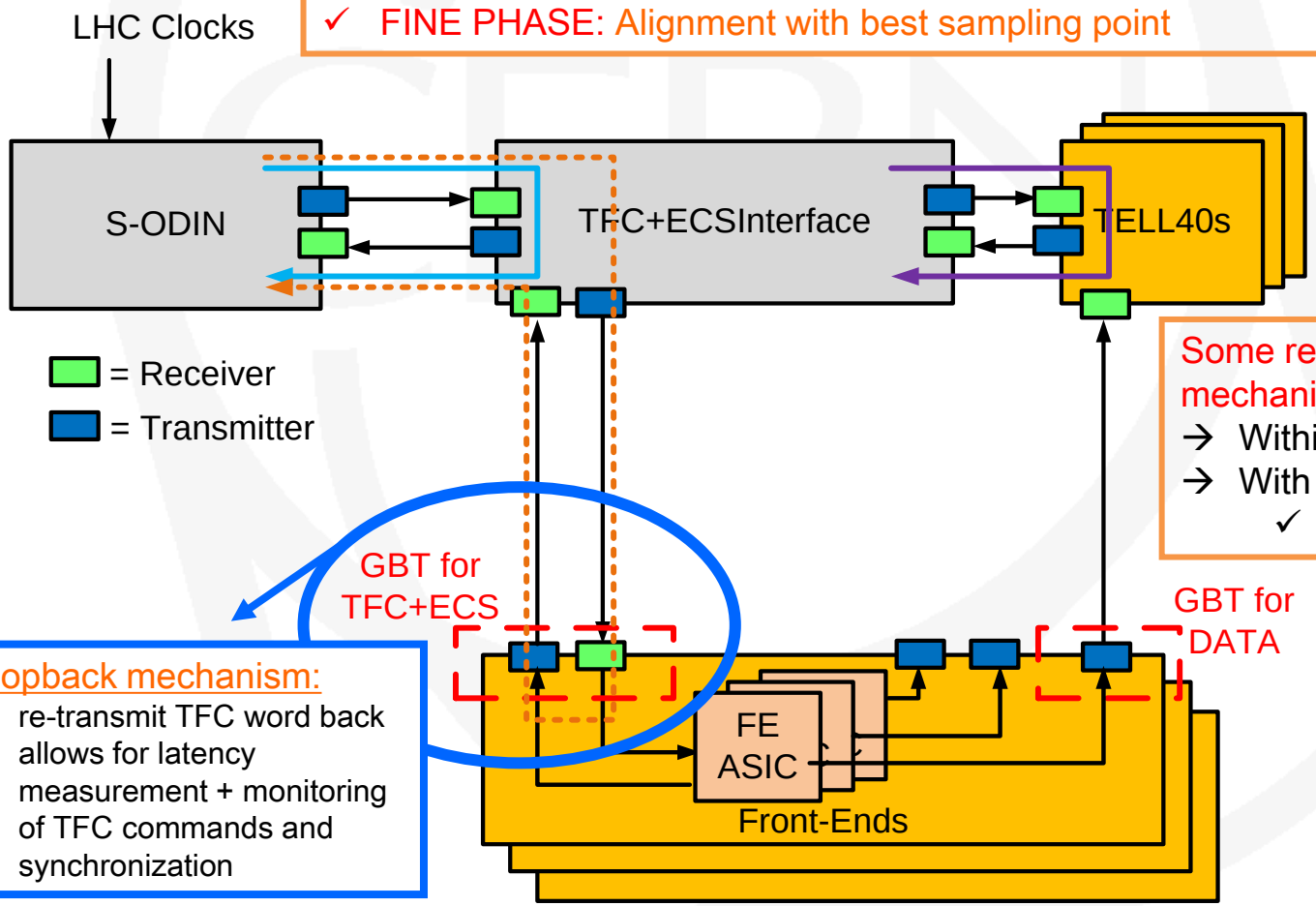
- To allow use of commands/resets for particular BXID
- Accounting of delays in S-ODIN: for now, 16 clock cycles earlier + time to receive
- Aligned to the furthest FE (simulation, then in situ calibration!)

TFC protocol to FE has implications on GBT configuration and ECS to/from FE

- see specs document!

Timing distribution

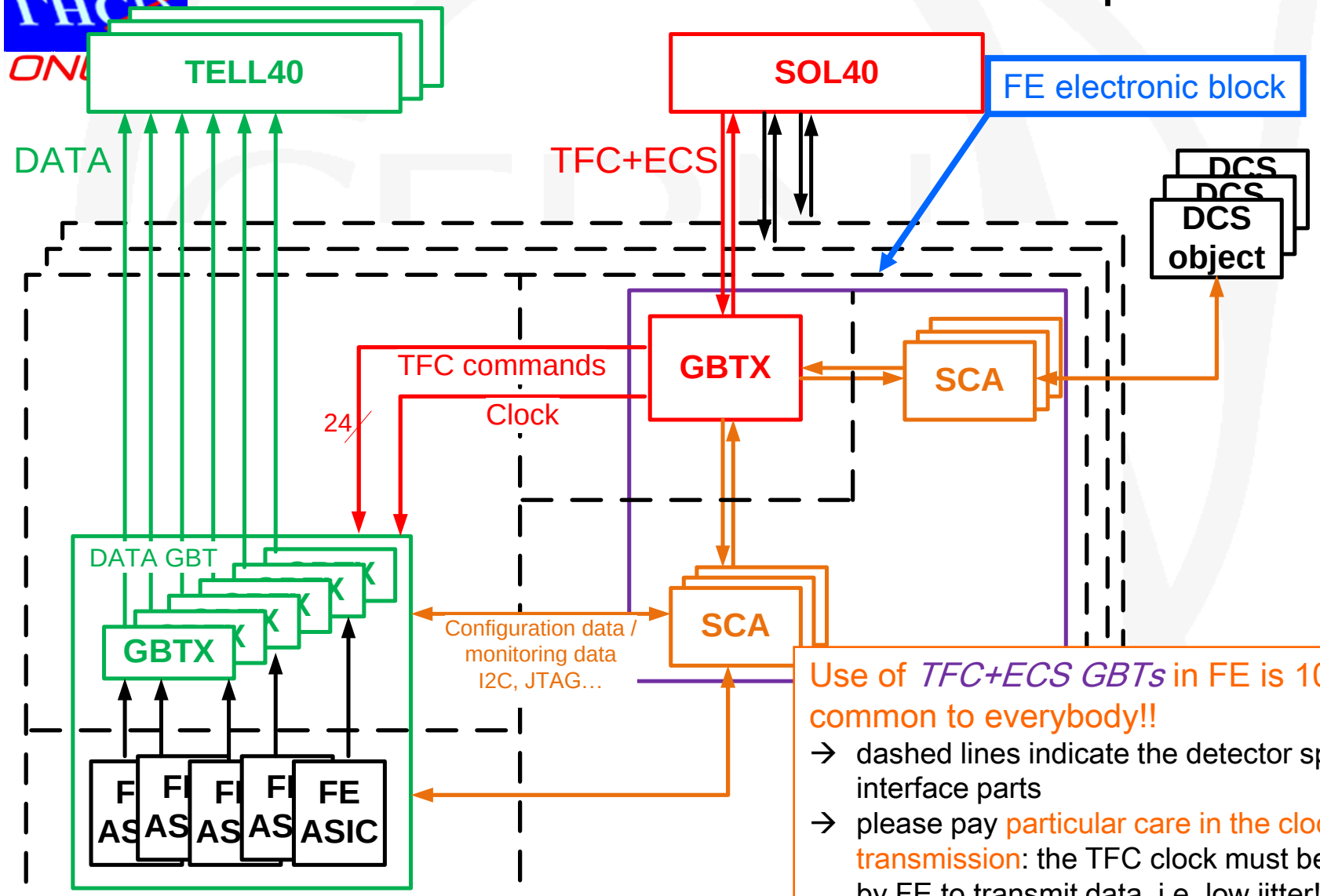
From TFC point of view, we ensure constant:
- ✓ LATENCY: Alignment with BXID
- ✓ FINE PHASE: Alignment with best sampling point



Some resynchronization mechanisms envisaged:
- → Within TFC boards
- → With GBT
  - ✓ No impact on FE itself

Loopback mechanism:
- → re-transmit TFC word back
- → allows for latency measurement + monitoring of TFC commands and synchronization

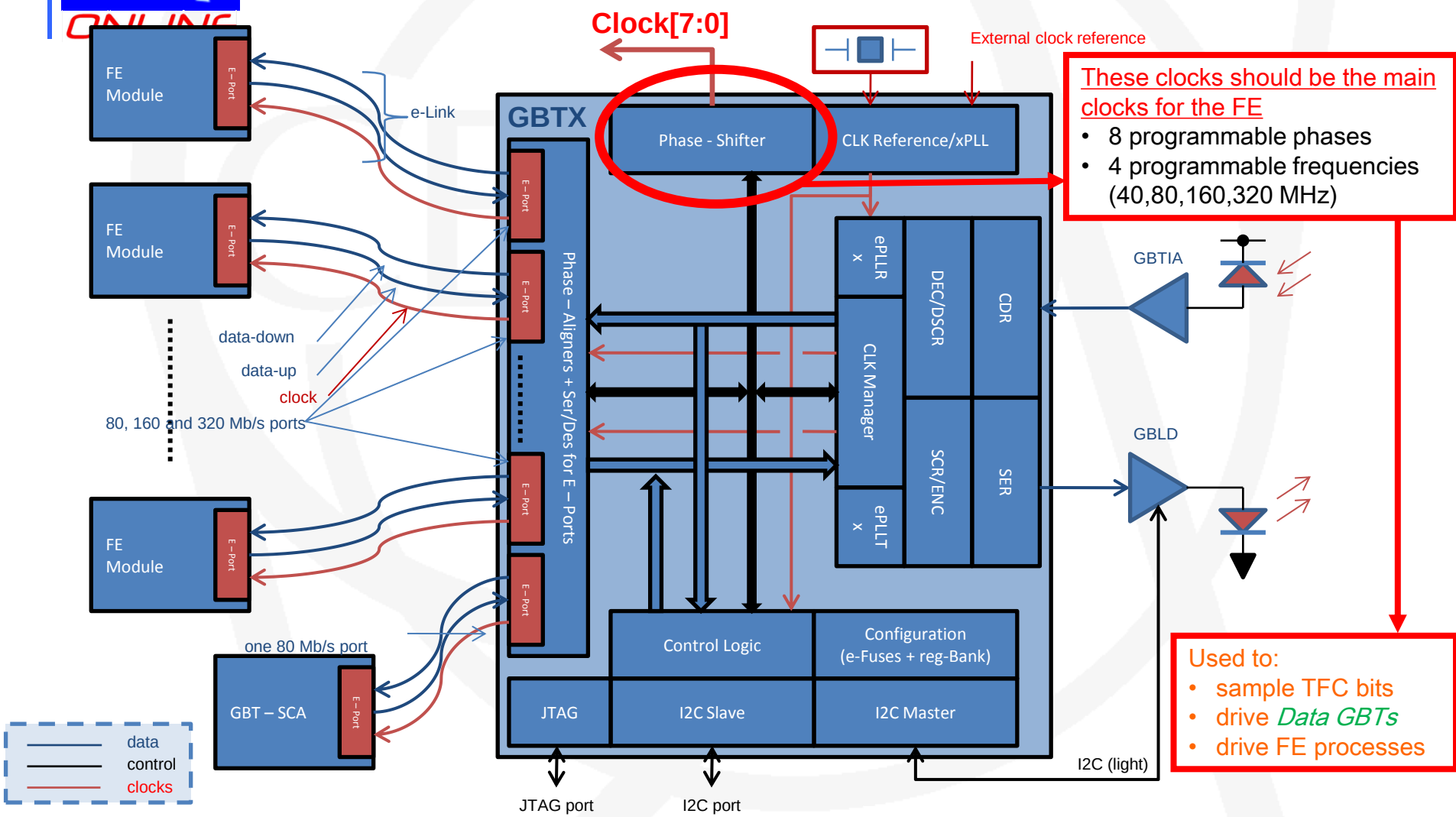
How to decode TFC in FE chips?



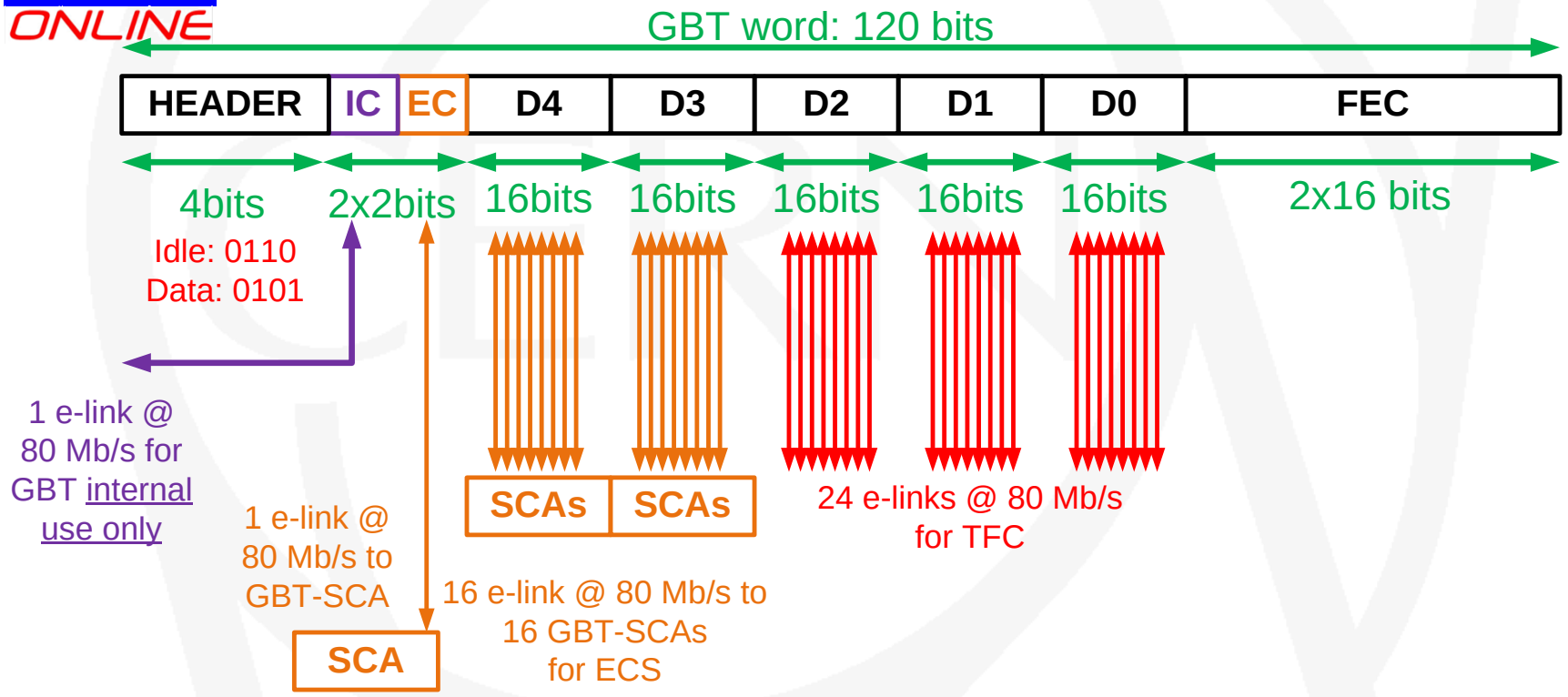
Use of *TFC+ECS GBTs* in FE is 100% common to everybody!!

- → dashed lines indicate the detector specific interface parts
- → please pay particular care in the clock transmission: the TFC clock must be used by FE to transmit data, i.e. low jitter!
  - ✓ Kapton cable, crate, copper between FE ASICs and GBTX

The TFC+ECS GBT



The TFC+ECS GBT protocol to FE

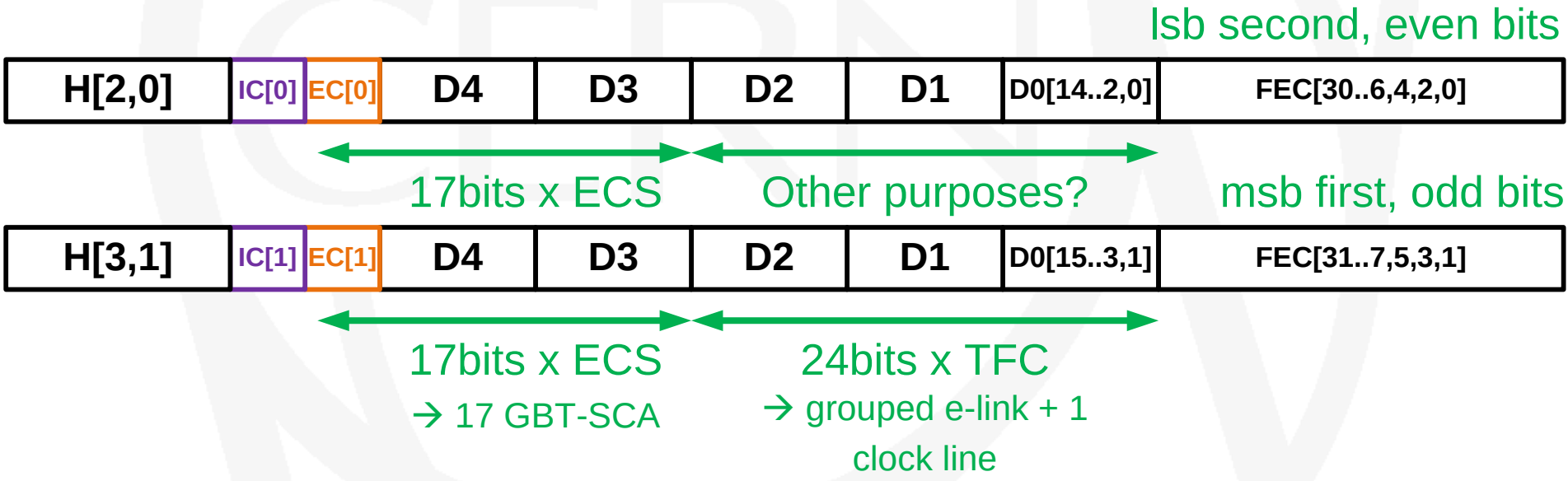


→ TFC protocol has direct implications in the way in which GBT should be used everywhere

- 24 e-links @ 80 Mb/s dedicated to TFC word:
  - ✓ use 80 MHz phase shifter clock to sample TFC parallel word
- TFC bits are packed in GBT frame so that they all come out on the same clock edge
  - ✓ We can repeat the TFC bits also on consecutive 80 MHz clock edge if needed

→ Leftover 17 e-links dedicated to GBT-SCAs for ECS configuring and monitoring (see later)

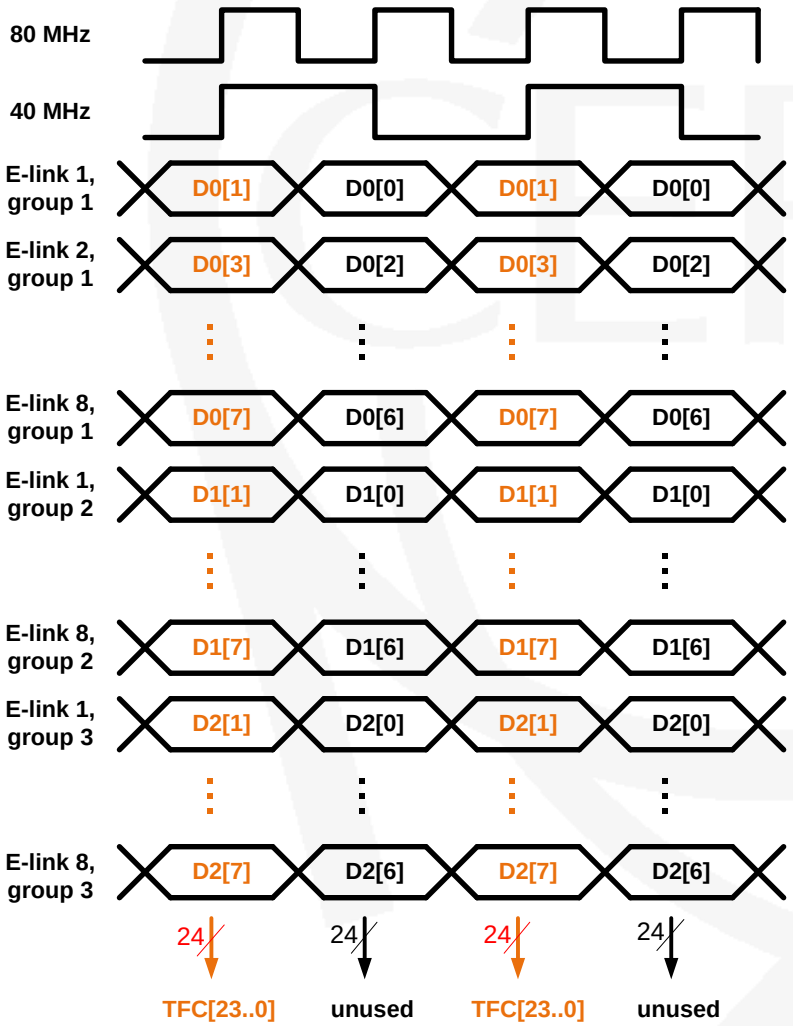
Words come out from GBT at 80 Mb/s



In simple words:

- Odd bits of GBT protocol on rising edge of 40 MHz clock (first, msb),
- Even bits of GBT protocol on falling edge of 40 MHz clock (second, lsb)

TFC decoding at FE after GBT



This is crucial!!

→ we can already specify where each TFC bit will come out on the GBT chip

→ this is the only way in which FE designers still have minimal freedom with GBT chip

- ✓ if TFC info was packed to come out on only 12 e-links (first odd then even), then decoding in FE ASIC would be mandatory!
- ✓ which would mean that the GBT bus would have to go to each FE ASIC for decoding of TFC command

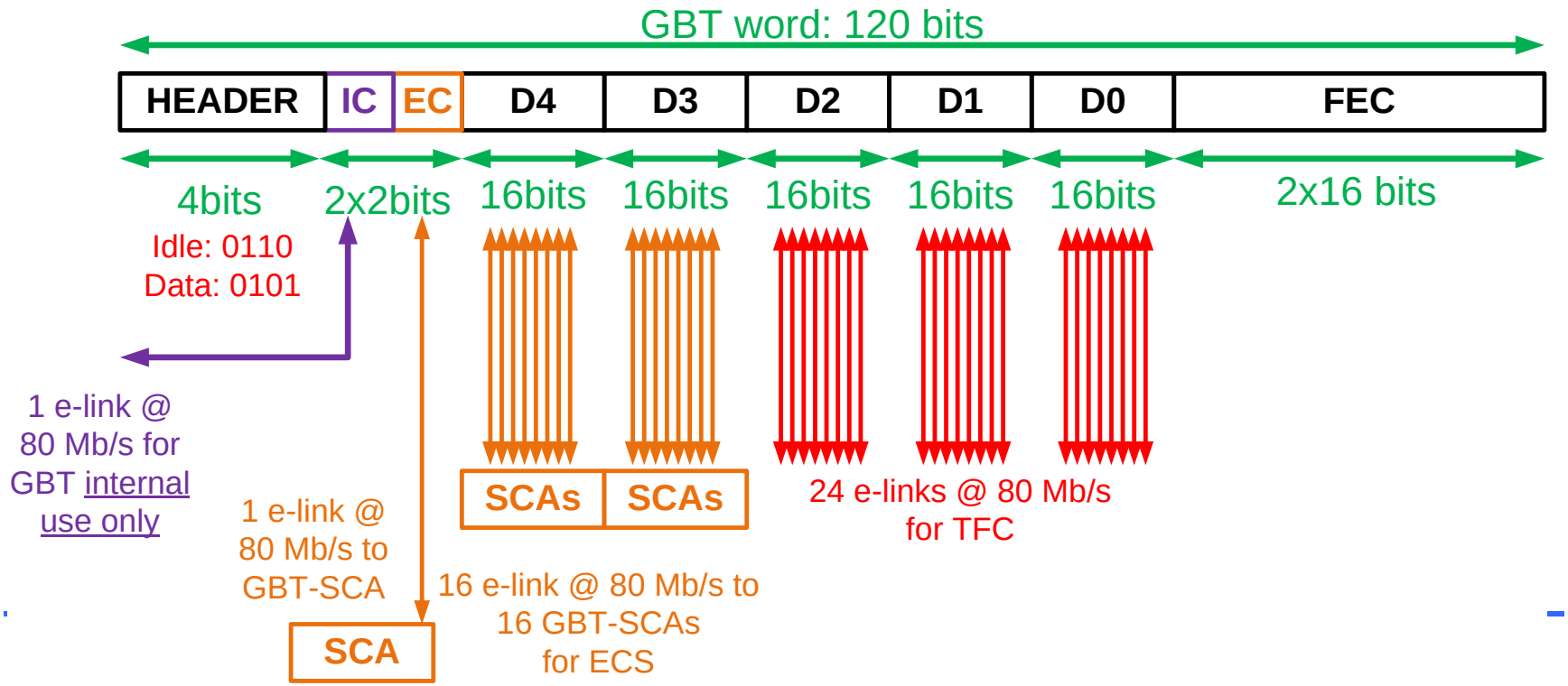
→ there is also the idea to repeat the TFC bits on even and odd bits in TFC protocol

- ✓ would that help?
- ✓ FE could tie logical blocks directly on GBT pins…

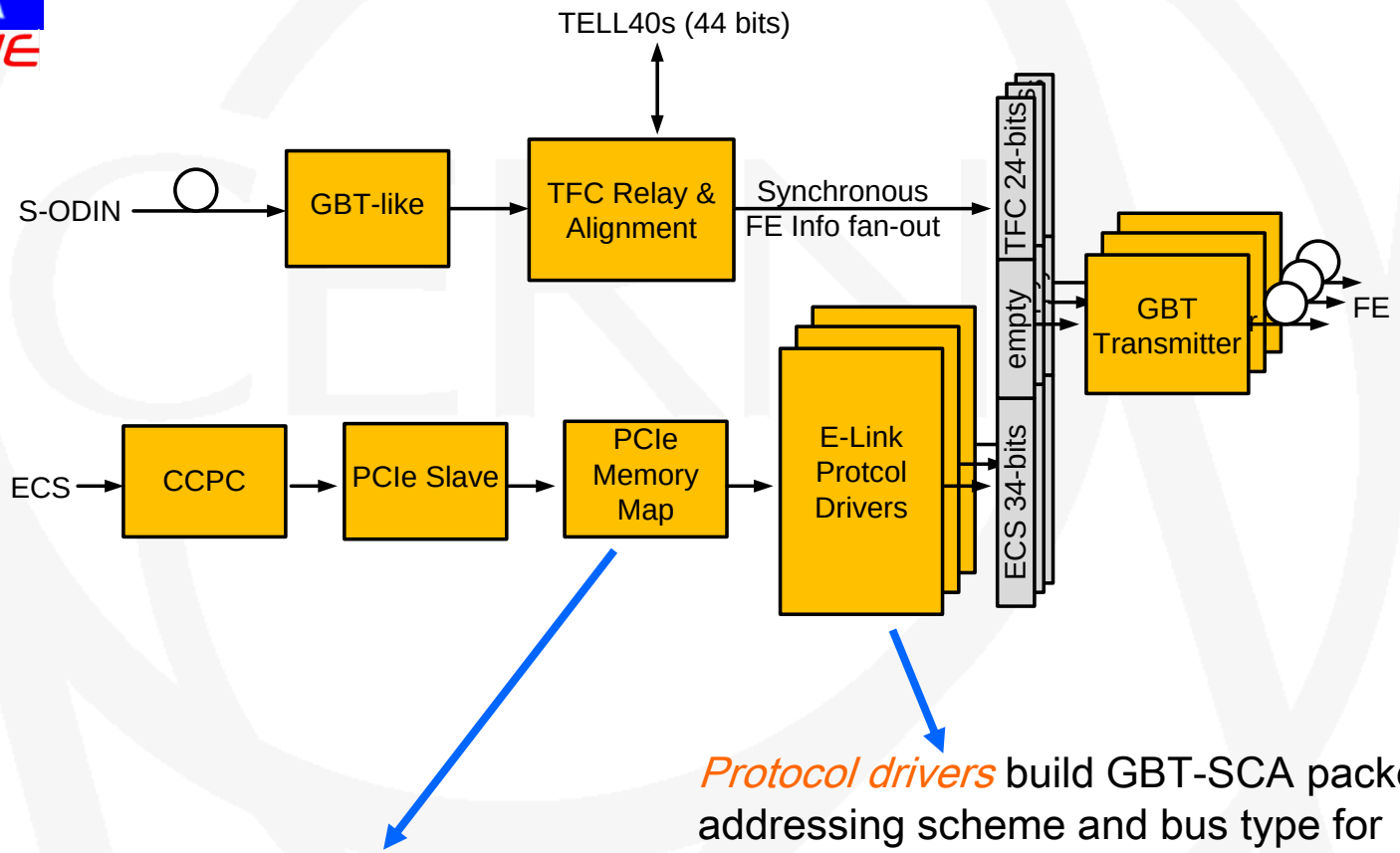
Now, what about the ECS part?

Each pair of bit from ECS field inside GBT can go to a GBT-SCA

- One GBT-SCA is needed to configure the *Data GBTs* (EC one for example?)
- The rest can go to either FE ASICs or DCS objects (temperature, pressure) via other GBT-SCAs
  - ✓ GBT-SCA chip has already everything for us: interfaces, e-links ports ..
    - → No reason to go for something different!
  - ✓ However, «silicon for SCA will come later than silicon for GBTX»…
    - → We need something while we wait for it!



SOL40 encoding block to FE!

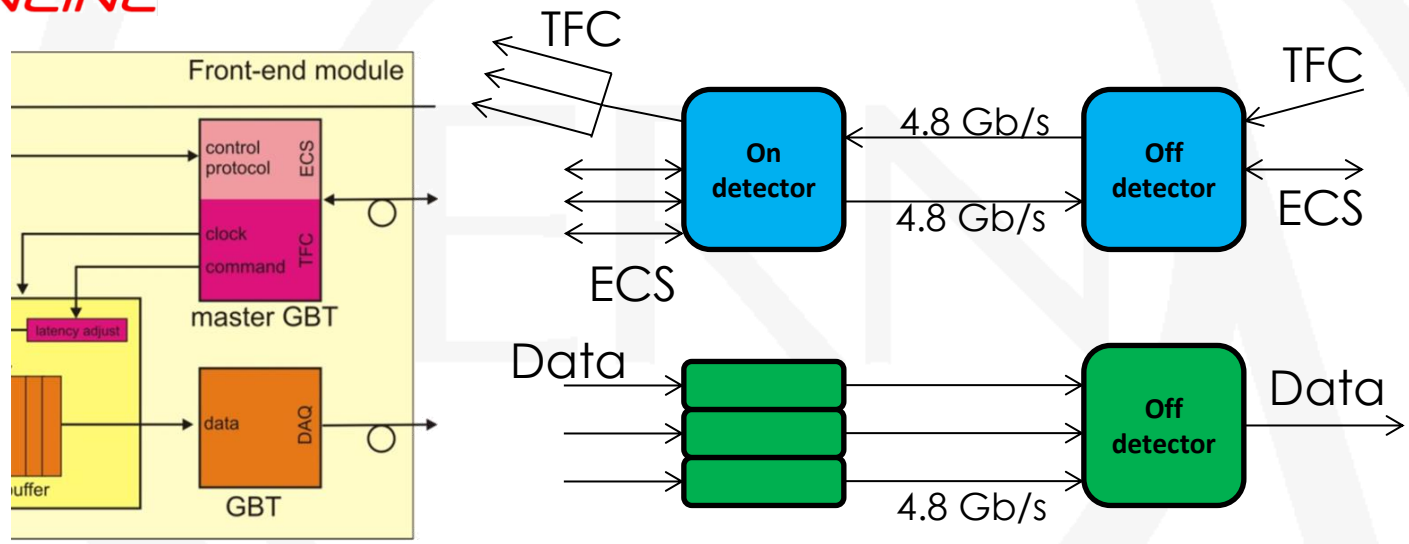


Memory Map with internal addressing scheme for GBT-SCA chips + FE chips addressing, e-link addressing and bus type: *content of memory loaded from ECS*

Protocol drivers build GBT-SCA packets with addressing scheme and bus type for associated GBT-SCA user busses to selected FE chip

→ *Basically each block will build one of the GBT-SCA supported protocols*

Fast & Slow Control to FE



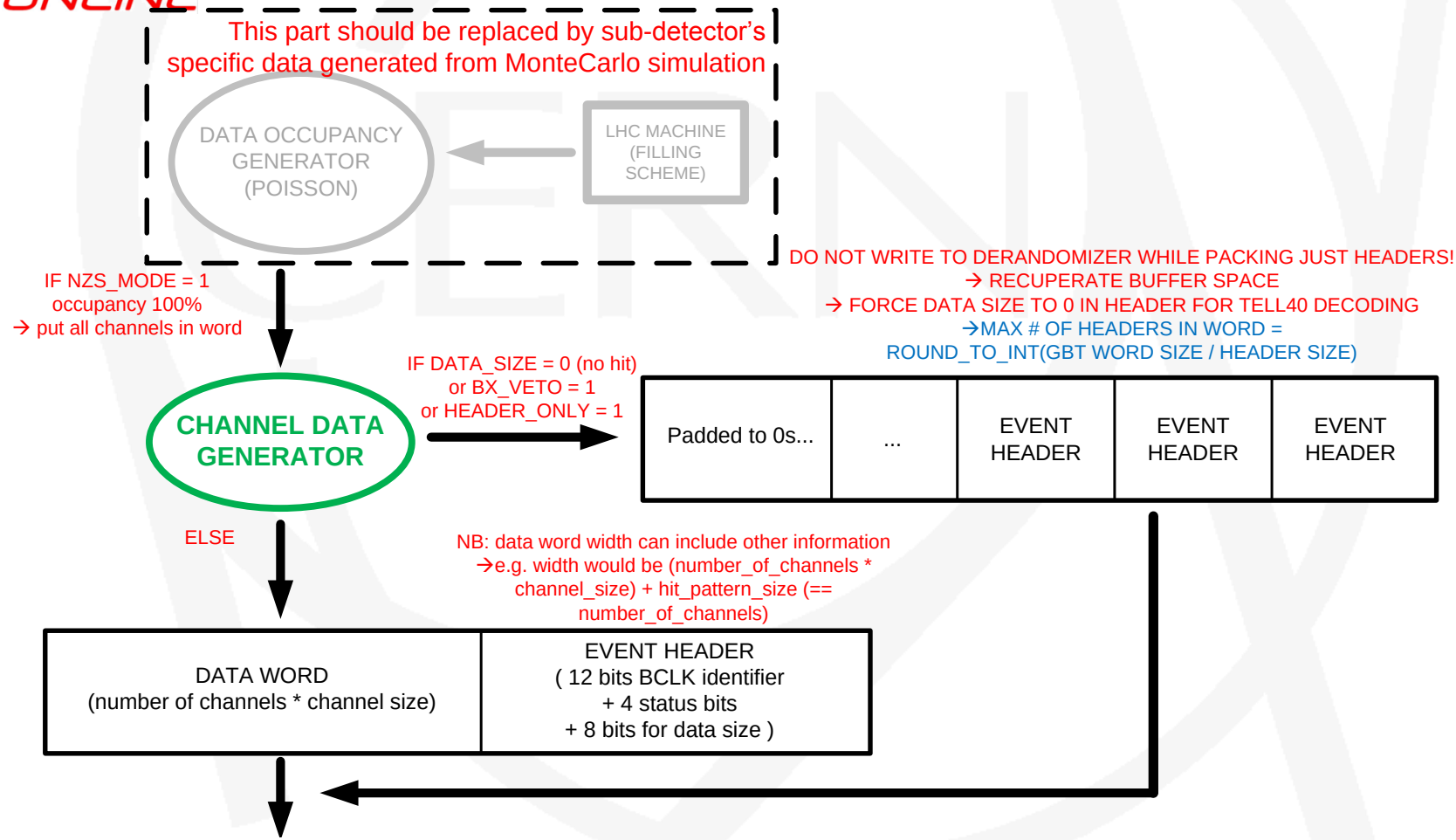
Separate links between controls and data

- A lot of data to collect
- Controls can be fanned-out (especially fast control)

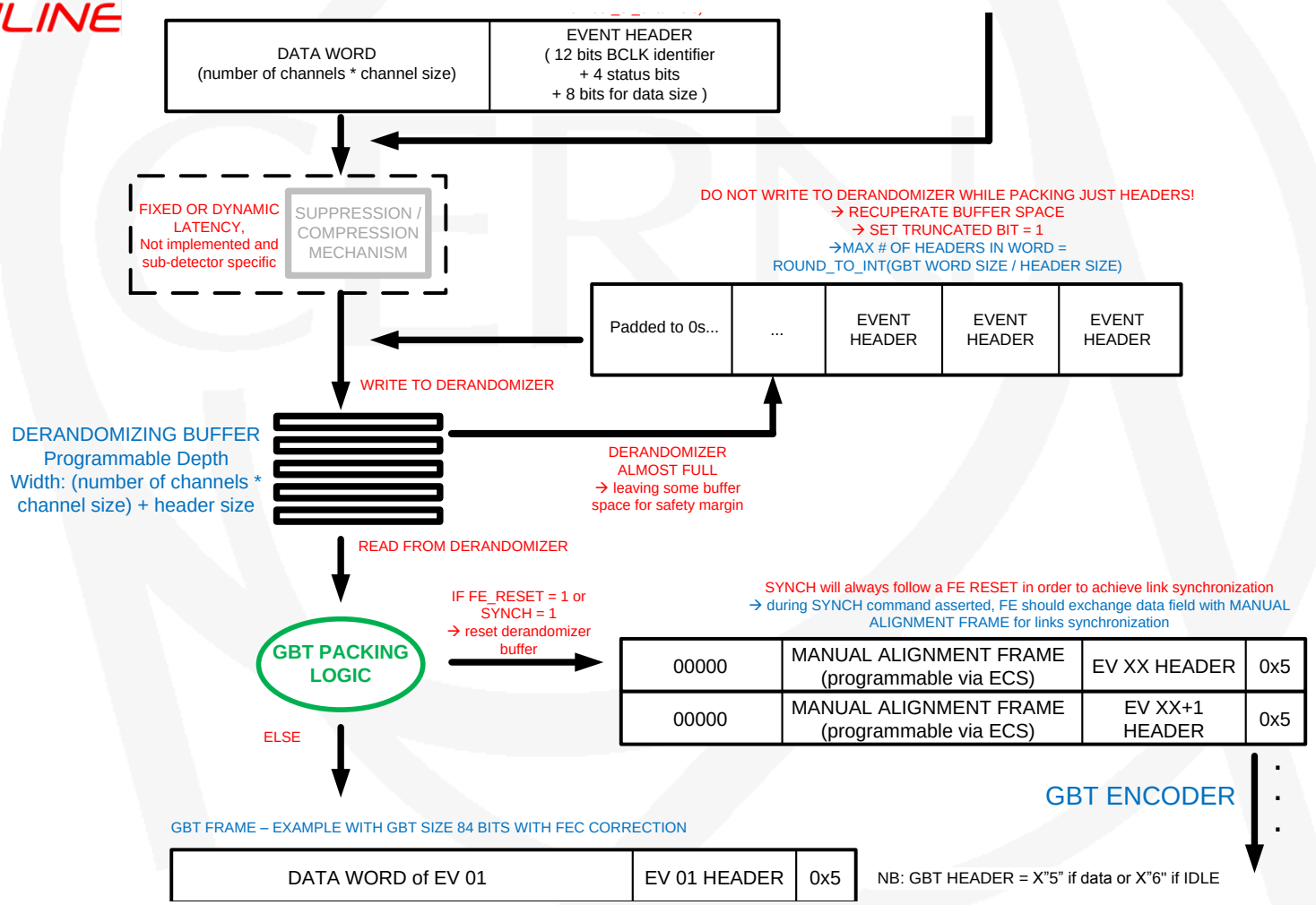
Compact links merging Timing, Fast and Clock (TFC) and Slow Control (ECS).

- Extensive use of GBT as Master GBT to drive Data GBT (especially for clock)
- Extensive use of GBT-SCA for FE configuration and monitoring

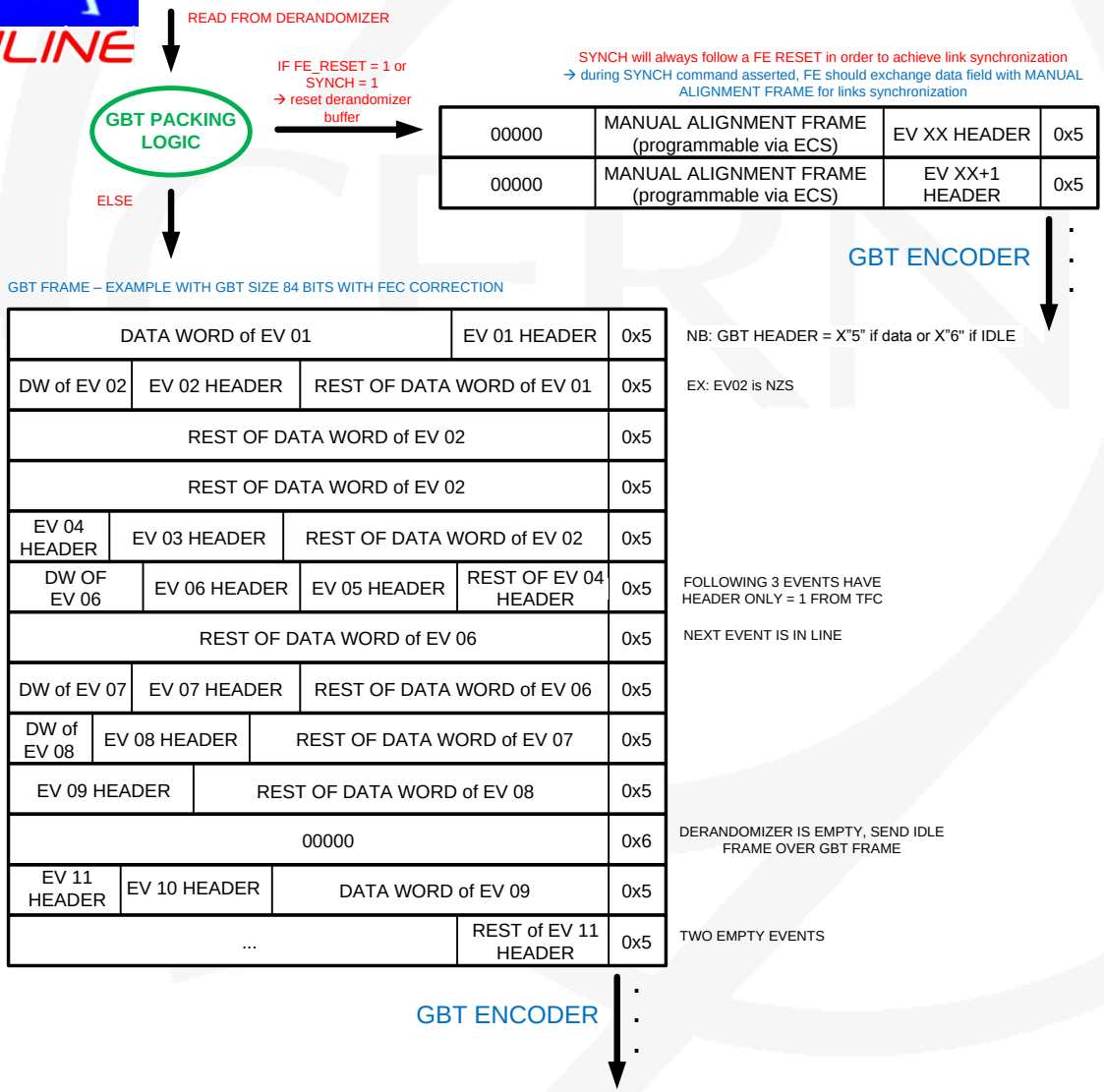
The code: FE data generator



The code: FE buffer manager



The code: GBT dynamic packing



Very important to analyze simulation output bit-by-bit and clock-by-clock!

The code: configuration

FE generic data generator is fully programmable:

- ✓ Number of channels associated to GBT link
- ✓ Width of each channel
- ✓ Derandomizer depth
- ✓ Mean occupancy of the channels associated to GBT link
- ✓ Size of GBT frame (80 bits or WideBus + GBT header 4 bits)

Extremely flexible and easy to configure with parameters

Covers almost all possibilities (almost…)

- ✓ Including flexible transmission of NZS and ZS

Including TFC commands as defined in specs

- ✓ Study dependency of FE buffer behaviour with TFC commands
- ✓ Study effect of packing algorithm on TELL40
- ✓ Study synchronization mechanism at beginning of run
- ✓ Study re-synchronization mechanism when de-synchronized
- ✓ Etc… etc… etc…

And it is fully synthesizable… ☺

Conclusions

Packing mechanism as specified in our document is feasible.

- ✓ Will be used temporarily to emulate FE generated data in global readout and TFC simulation.

However, very big open questions:

- ✓ Is your FE compatible with such scheme? What about such code in an ASIC?
- ✓ Behaviour of FE derandomizer will strongly depend on your compression or suppression mechanism.
  - If dynamic could create big latencies
  - If your data does not come out of order can become quite complicated…
- ✓ Behaviour of FE derandomizer will strongly depend on TFC commands
  - FE buffer depth should not rely on having a BX VETO! Aim at a bandwidth for fully 40 MHz readout → BX VETO solely to discard events synchronously.
  - What about SYNCH command? When do you think you can apply it? Ideally after derandomizer and after suppression/compression, but…
- ✓ How many clock cycles do you need to recover from an NZS event?
  - Can you handle consecutive NZS events?

Old TTC system support and running two systems in parallel

We already suggested the idea of a *hybrid system*:

reminder: L0 electronics relying on TTC protocol

→ part of the system runs with old TTC system

→ part of the system runs with the new architecture

How?

1. Need connection between S-ODIN and ODIN (bidirectional)
   → use dedicated RTM board on S-ODIN ATCA card
2. In an early commissioning phase ODIN is the master, S-ODIN is the slave
   → S-ODIN task would be to distribute new commands to new FE, to new TELL40s, and run processes in parallel to ODIN
   → ODIN tasks are the ones today + S-ODIN controls the upgraded part
   ✓ In this configuration, upgraded slice will run at 40 MHz, but positive triggers will come only at maximum 1.1MHz…
   - Great testbench for development + tests + apprenticeship…
   - Bi-product: improve LHCb physics programme in 2015-2018…
3. In the final system, S-ODIN is the master, ODIN is the slave
   → ODIN task is only to interface the L0 electronics path to S-ODIN and to provide clock resets on old TTC protocol