



# TFC update and TFC simulation testbench

LHCb Electronics Upgrade Meeting  
14 February 2012

---

Federico Alessio  
Richard Jacobsson

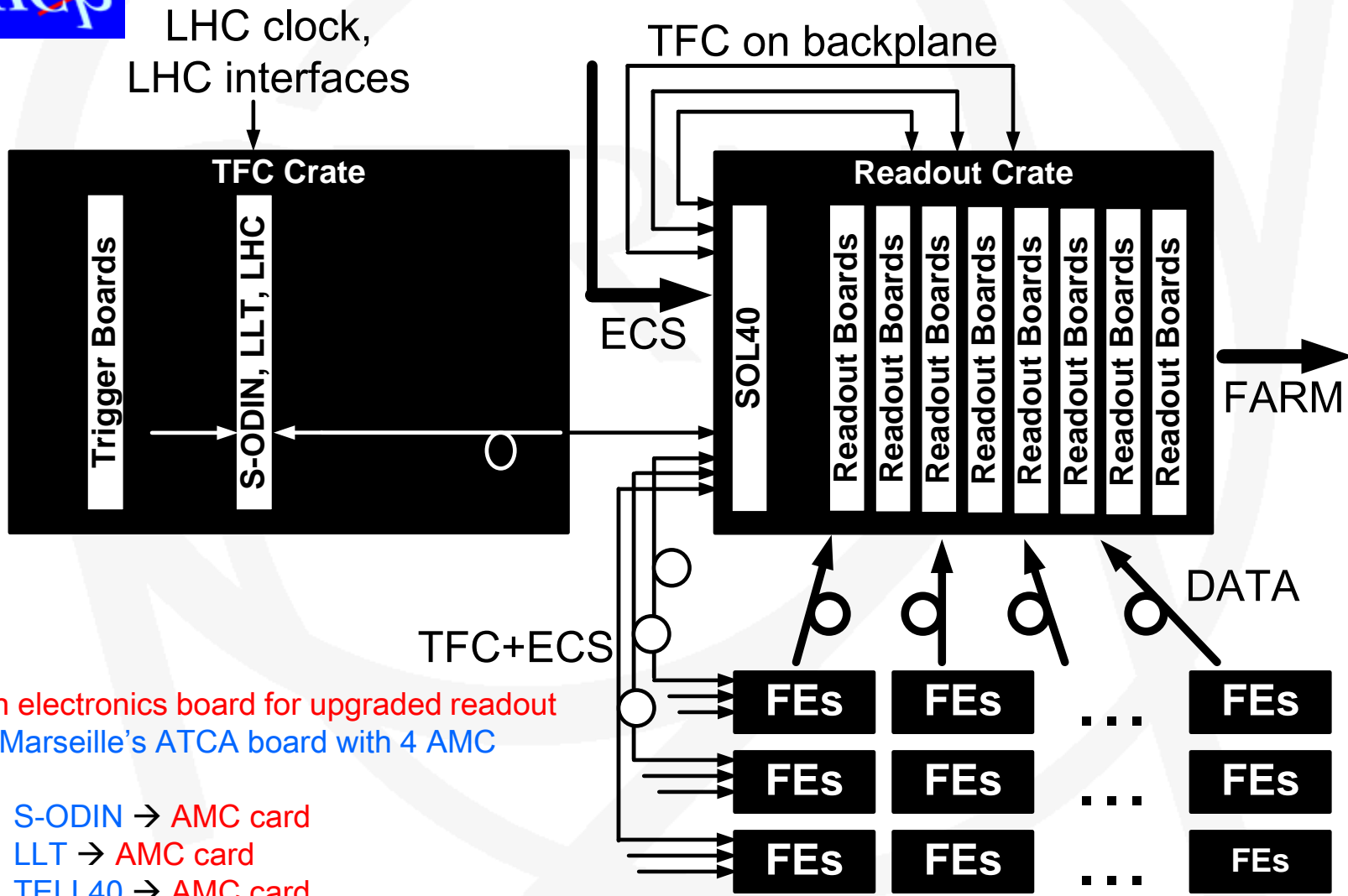
---



# Outline

- ✓ Modifications in SOL40-TELL40 protocol
- ✓ First version of S-ODIN firmware
  - Block diagram
  - Latency
  - Plans
- ✓ TFC simulation testbench
  - Requirements
  - Status
  - Plans

# The upgraded physical readout slice



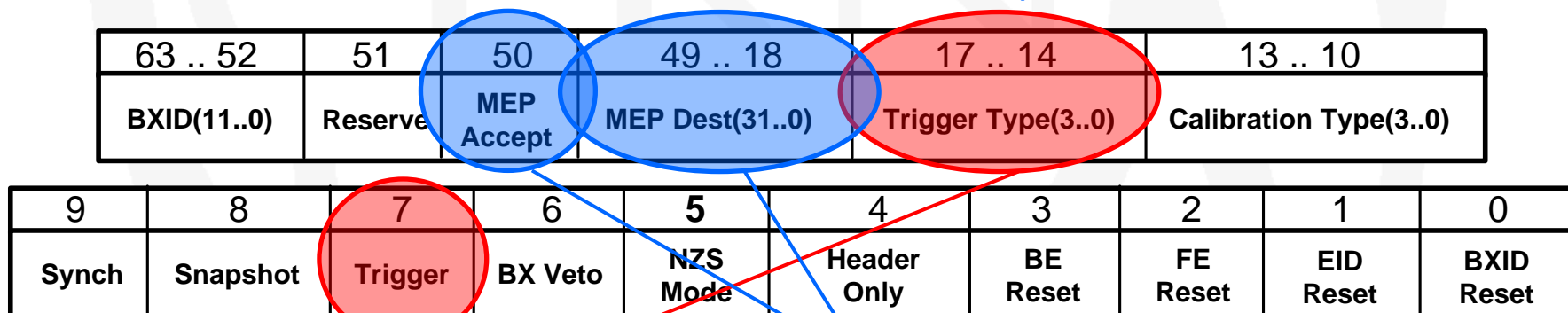
Common electronics board for upgraded readout system: Marseille's ATCA board with 4 AMC cards

- 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



Constant latency after BXID

MEP accept command when MEP ready:  
 → Take MEP address and pack to FARM  
 → No need for special address, dynamic

- ✓ 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!



# SODIN firmware v1r0

## SODIN firmware v1r0 in preparation

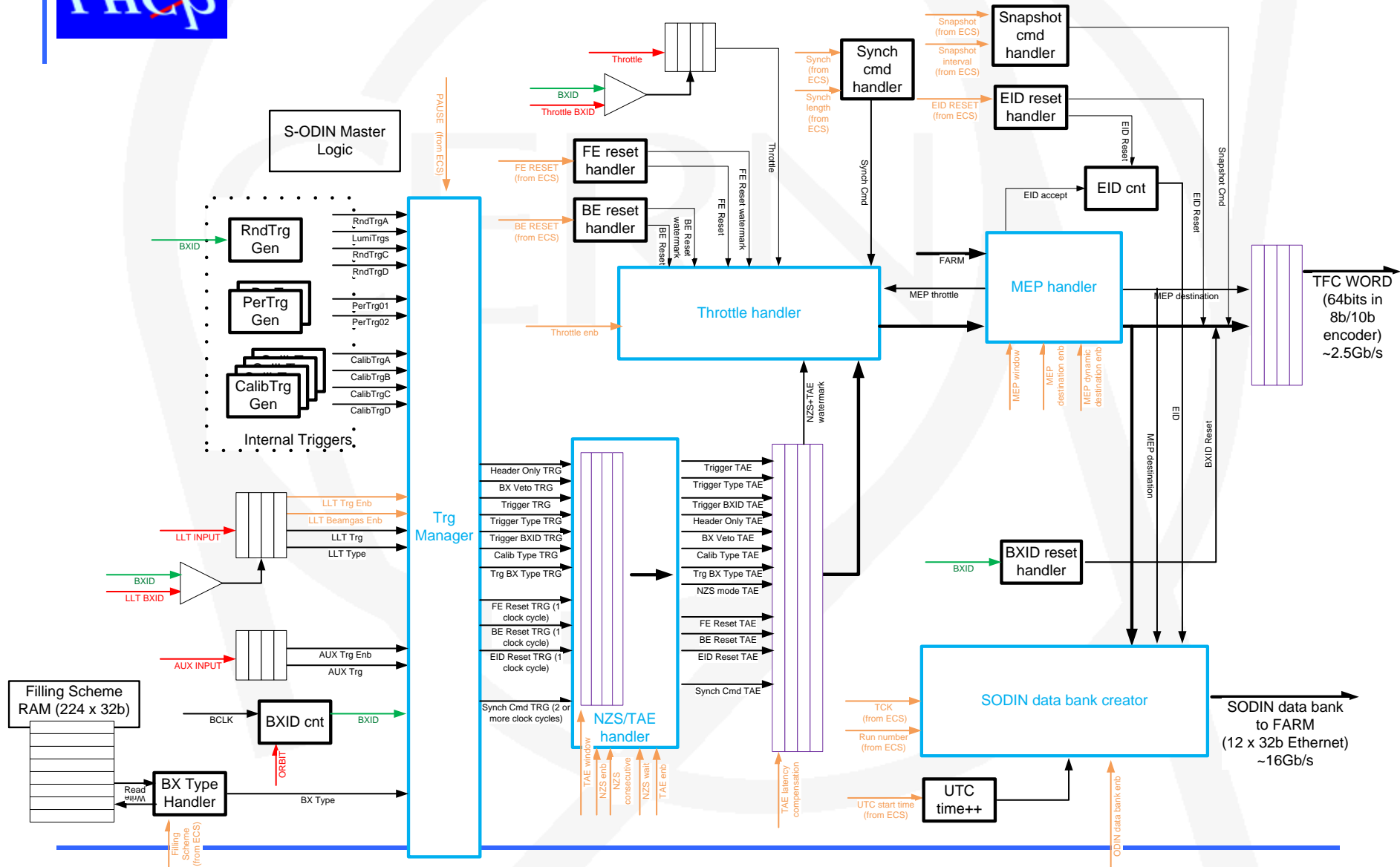
- first version ready now
- has been rewritten completely wrt to current ODIN:
  - obviously kept same philosophy (from Richard's successes)
    - ✓ handling of destinations of MEPs will be the same
    - ✓ data bank from sw point of view will look the same as current ODIN (see EDMS #704084)

## SODIN v1r0 will be available for mini-DAQ

- with basics features described in our specs documents (see LHCb-PUB-2012-001 and LHCb-PUB-2012-017)
- Your FE must cope with them ... !



# SODIN firmware v1r0 – block diagram





# SODIN firmware v1r0 – latency

Latency in ODIN =

# clock cycles between LLT yes and output of TFC word

→ Entirely programmable

- 3 buffers/pipelines to allow to change it at different stages
  - TAE half window (to change TFC word in the past...)
  - TAE/NZS buffer
  - Output buffer
- Allows to change delays later on...

Minimum latency =

max half window TAE + 7 clk cycles for internal processing

- Decided for max 31 consecutive TAE events (15 half window)
- Simulated latency = 22 clock cycles  
(see picture later)





# SODIN firmware v1r0 – plans

1. SODIN v1r0 is the version for the *mini-DAQ (AMC + AMC\_TP from Marseille)*
  - first version based on final version, but smaller, simpler and robust
    - ✓ to be expanded in collaboration with sub-detectors/TELL40
  - being finalized **now** using simulation testbench (see later)
2. Will need to be validated with the hardware
  - ~2 weeks/1 month of tests with Marseille's AMC + LLI + QSYS
3. Will need some kind of control system for it
  - to be organized, but supposedly based on PVSS (*ops, sorry WinCC...*)
  - hence, *WinCC* panels and scripts similar to current *PVSS* ODIN
4. SOL40 v1r0 fw will be done in parallel
  - and packaged together with SODIN for the mini-DAQ

Generally we should aim at a first version v1r0 for each fw in the mini-DAQ

→ A robust basic first version for each fw so that FE ppl can develop and test their FE → **availability early on after hardware is ready!**



# TFC simulation testbench

## Revived old simulation done in 2009

→ see <http://indico.cern.ch/conferenceDisplay.py?confId=71146>

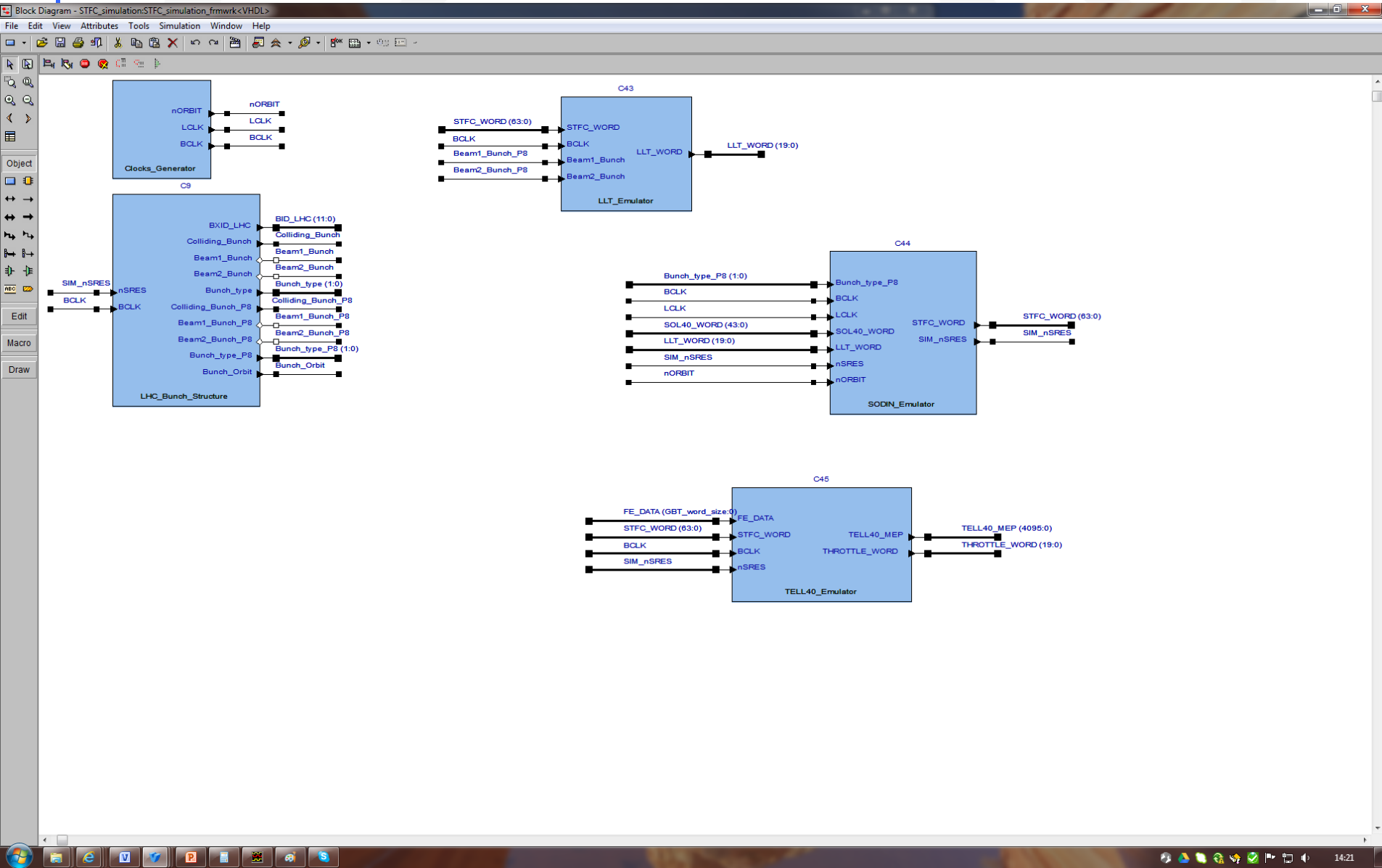
→ attached to this agenda too

## *Generic simulation testbench* for the TFC development

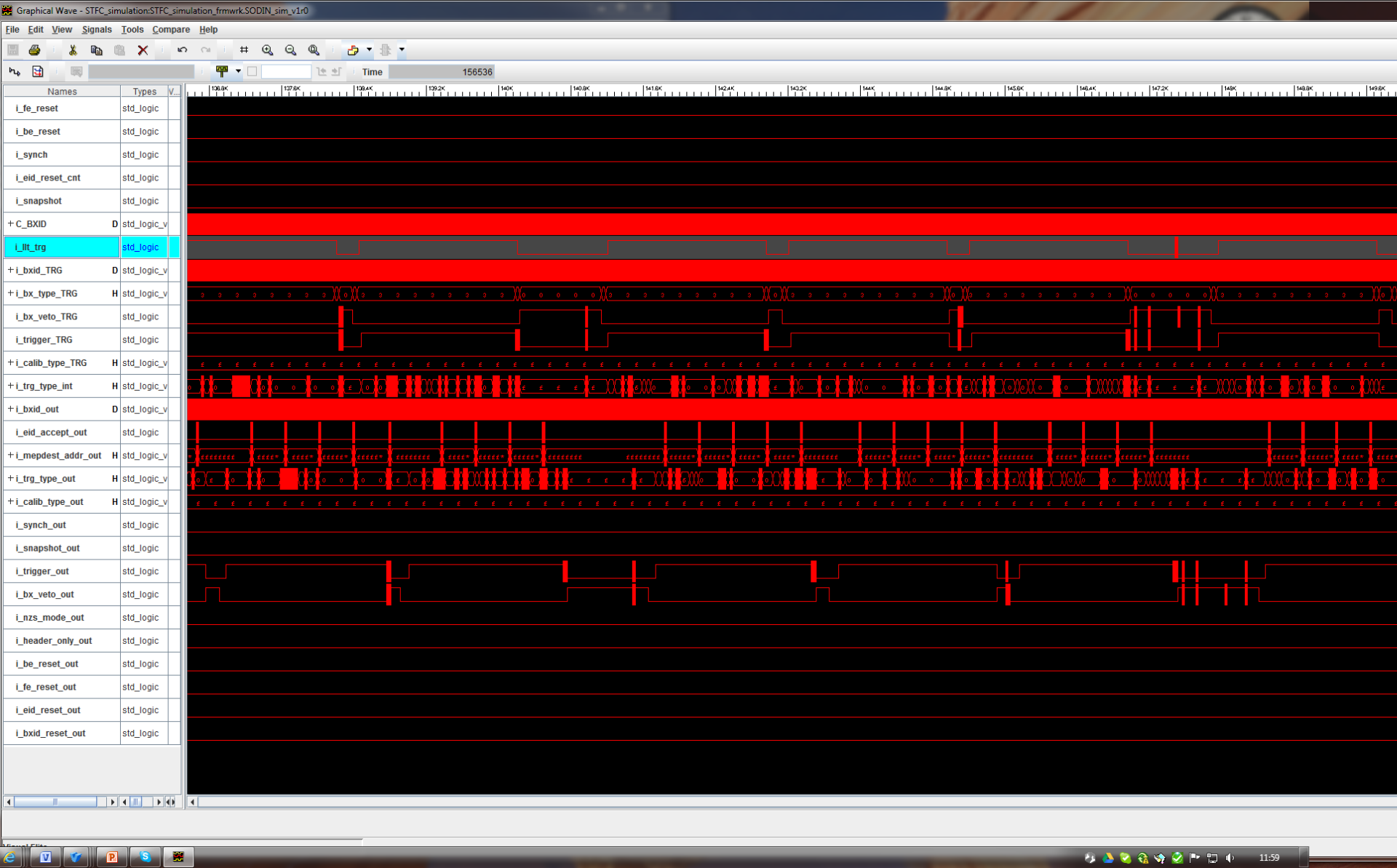
- Using VisualElite from Mentor Graphics, but can be done also in ModelSim
- Synthesizable “clock-level fidel” simulation of SODIN and SOL40
  - ✓ Emulation of other components
- Based on “configurable parameters” to easily test the system
- Easy to implement components to test various protocols/buffer occupancies/logical blocks
- Easy to include real TELL40 logic, FE logic,GBTs...
  - ✓ Already implemented FE, TELL40, LLT, LHC filling scheme *emulations*
  - ✓ *Emulate* behavior of FARM and control system (high-level emulation)

NB: will be needed anyway for the development and maintenance of SODIN and SOL40 fw!

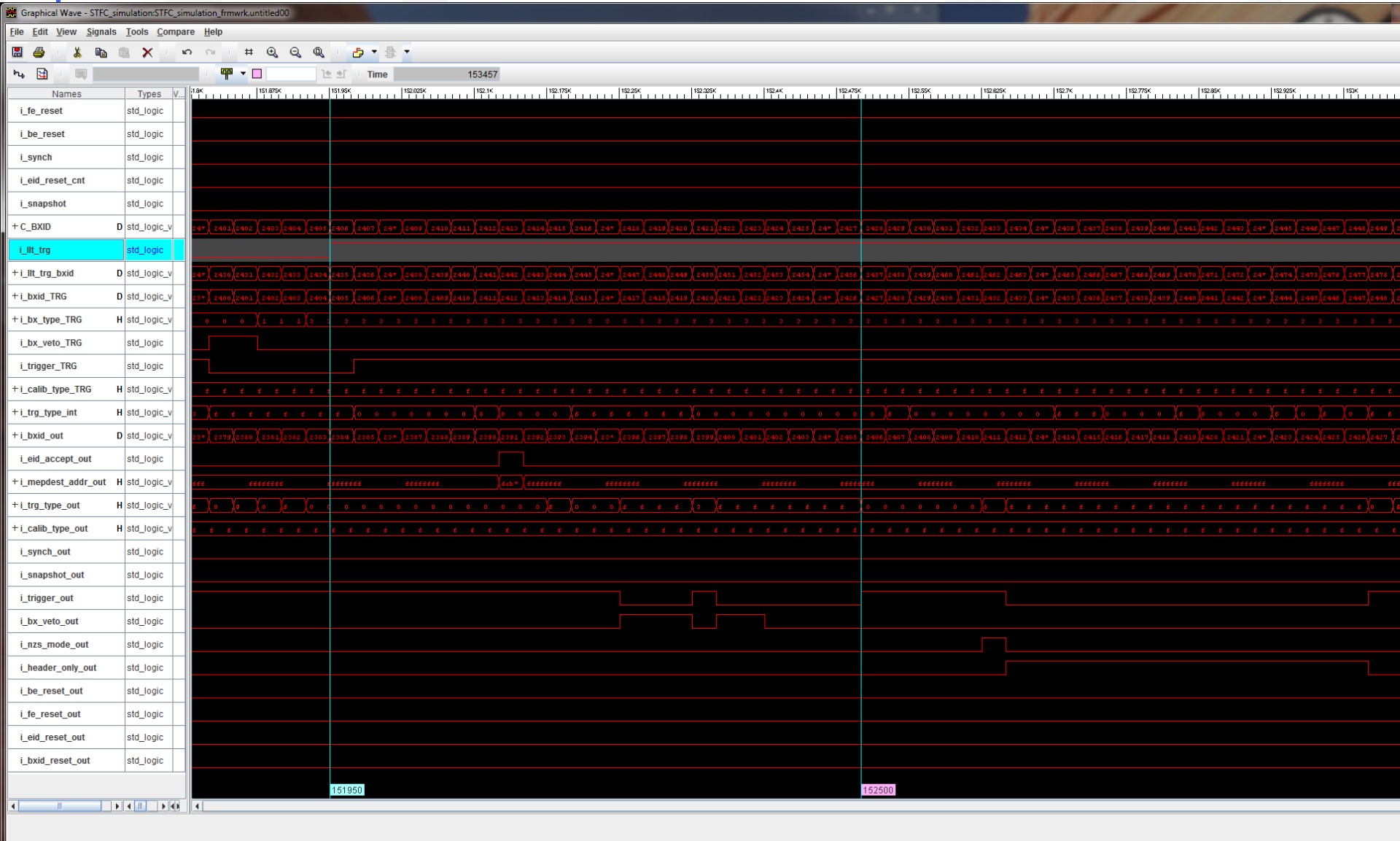
# Example - SODIN



# Example - SODIN



# Example - SODIN





# TFC simulation testbench - plans

## Plan to expand it including:

- SOL40 fw
- TELL40 emulation
  - ✓ eventually replaced by synthesizable TELL40 fw
- FE generic channel emulation
  - ✓ eventually this can be replaced by real FE code
  - here formal request to FE people to provide their own code ☺
- GBT and GBT-SCA emulations
  - ✓ GBT already available (since 2008, thanks to Marseille)
  - ✓ GBT-SCA discussion with PH-ESE to be done

NB 1: simulation of buffers, logic and protocols can easily be done here

- at clock-level

NB 2: does not exclude/compete with simulation with QSYS/ModelSim

- aims at integrating it (by including GBT-SCAs, FE codes ...)

# Qs & As?

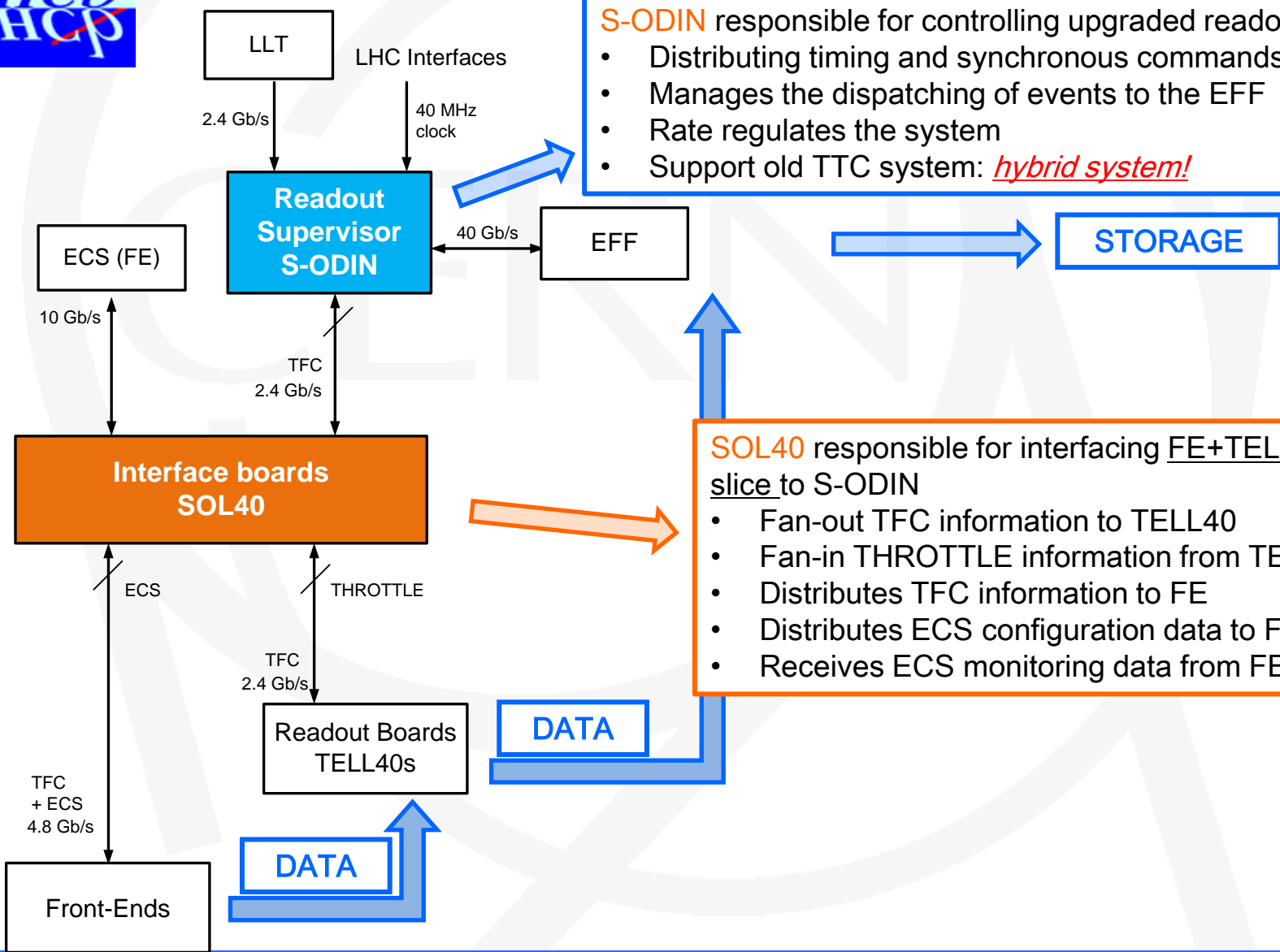


# System and functional requirements

1. **Bidirectional** communication network
2. Clock jitter, and phase and latency control
  - ✓ At the FE, but also at TELL40 and between S-TFC boards
3. **Partitioning** to allow running with any ensemble and parallel partitions
4. **LHC** interfaces
5. Events **rate control**
6. **Low-Level-Trigger** input
7. Support for **old TTC-based** distribution system
8. **Destination control** for the event packets
9. Sub-detectors **calibration triggers**
10. **S-ODIN data bank**
  - ✓ Information about transmitted events
11. Test-bench **support**



# The S-TFC system at a glance



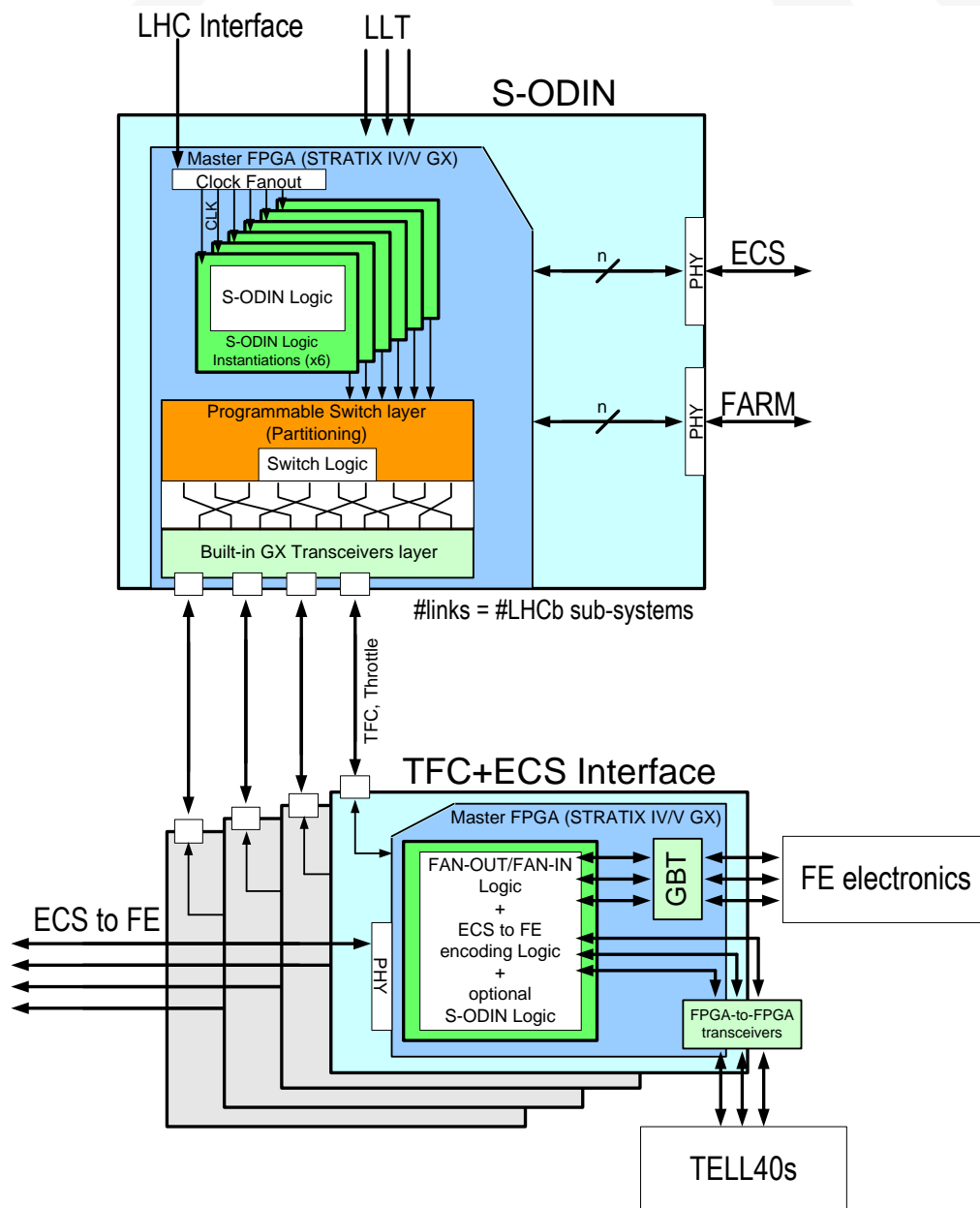
**S-ODIN** responsible for controlling upgraded readout system

- Distributing timing and synchronous commands
- Manages the dispatching of events to the EFF
- Rate regulates the system
- Support old TTC system: *hybrid system!*

**SOL40** responsible for interfacing FE+TELL40 slice to S-ODIN

- Fan-out TFC information to TELL40
- Fan-in THROTTLE information from TELL40
- Distributes TFC information to FE
- Distributes ECS configuration data to FE
- Receives ECS monitoring data from FE

# S-TFC concept reminder





# Partitioning

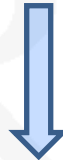
Partitioning is assured by having:

- ✓ Many instances of S-ODIN inside main FPGA
  - Only one is the LHCb Master
- ✓ Switching is done inside main FPGA
  - Simply ensure that TFC information are sent to right output
- ✓ TFC+ECSInterface «interfaces» S-ODIN with partitioned FE+TELL40 slice(s)
  - Logical distribution of TFC+ECSInterfaces in TELL40s crates
  - Important to respect the «logical concept» of partitioning
    - Should not span over different sub-systems with same TFC+ECSInterface...
    - Need at least one dedicated TFC+ECSInterface for each sub-system

# «BX VETO»

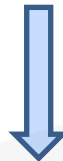
@ 40 MHz

S-ODIN vetoes the readout of an event



Based on filling scheme

- Used to control the rate of readout while  $< 30\text{MHz}$
- INDEPENDENT FROM LLT DECISION!



FE can use this info to recuperate time for processing events

- Only header for vetoed events
- Flexible packing of data into GBT frame

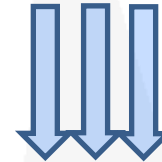
# Sending a «LLTyes»

@ 40 MHz

S-ODIN receives decisions from LLT



Special triggers



S-ODIN aligns and applies priority scheme on trigger types

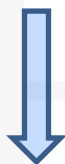
Rate regulation  
(next slide) →



S-ODIN sends out a “LLTyes” to TELL40 at  
a fixed latency wrt BXID!

@ 40 MHz

TELL40 raises the throttle bit



SOL40 compiles a throttle word with BXID and sends it to S-ODIN



MEP request scheme  
(next slide)



S-ODIN rejects event(s) until throttle is released  
→ In practice: the subsequent “LLTyes”(s) become “LLTno”(s)!



# S-TFC readout control sequences

## “BXID” and “BXID Reset”

- Every TFC word carries a BXID for synchronicity checks of the system
- A BXID Reset is sent at every turn of the LHC (orbit pulse)
  - ✓ Should only reset the internal bunch counter of the FE

## “FE RESETS”

- Bit set for one clock cycle in TFC word
- Reset of FE operational logic for data processing, formatting, transmission...
  - ✓ Should not touch the internal bunch counter
  - ✓ FE electronics should be back asap:
    - S-ODIN will ensure no data is being accepted during the FE reset process
    - wait to the slowest by setting Header Only bit, i.e. no data is recorded at FE
  - ✓ Reset TELL40 data input logic: the same bit is sent to TELL40 for same BXID

## “HEADER ONLY”

- **Idling the system**: only header (or few bits) in data word if this bit is set
  - ✓ Multiple purposes: set it during reset sequence, during NZS transmission, during TAE mode...

## “BX VETO”

- Based **exclusively** on filling scheme, processing of that particular event is inhibited
  - ✓ Only header (or few bits) in data word if this bit is set
  - ✓ Allows “recuperating” buffer space in a LHC-synchronous way if taken into account



# S-TFC readout control sequences

## “CALIBRATION TYPE” COMMAND

- Used to take data with special trigger pulses (periodic, calibration)
  - ✓ Dedicated 4 bits: i.e. 4 different calibration commands possible
  - ✓ Dynamic association to be used for calibration and monitoring
    - **Absolute need** of delays to account for each individual delay in the detectors
  - ✓ S-ODIN overrides LLT decision at TELL40
    - Periodic or calibration higher priority (but lower rate, max 11.245kHz each!)

## “NZS MODE”

- Read out (all) FE channels non-zero suppressed
  - ✓ Packing of full set of bits in many consecutive GBT frames:
    - ✓ **Needs buffering at FE (same buffer as data or different buffer?)**
- Possible to have also multi-NZS readout: *consecutive NZS events*
  - ✓ S-ODIN will take care of setting Header Only bit for a defined set of clock cycles later to allow recuperating buffer space (programmable as well, to the slowest of the detector)

*What are the sub-detector requests in term of NZS?*

*What do you need and what do you want to do?*

## “SNAPSHOT”

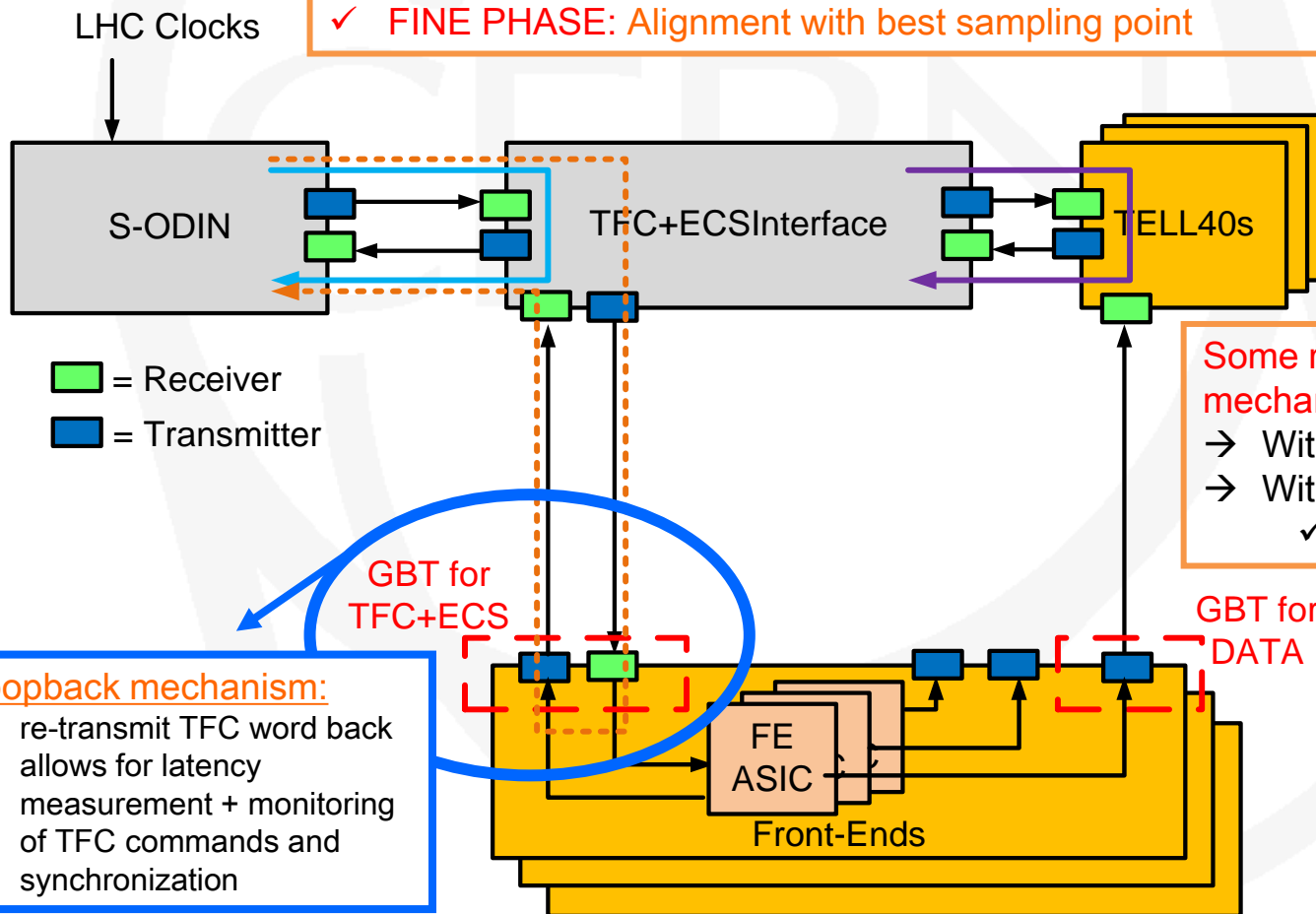
- Read out all status and counter registers in a “latched” way
  - ✓ Latch monitoring registers on snapshot bit, which is set periodically (programmable) and also single shot
  - ✓ When snapshot bit is received, send all data via ECS field in TFC on best effort



# Timing distribution

From TFC point of view, we ensure constant:

- ✓ LATENCY: Alignment with BXID
- ✓ FINE PHASE: Alignment with best sampling point



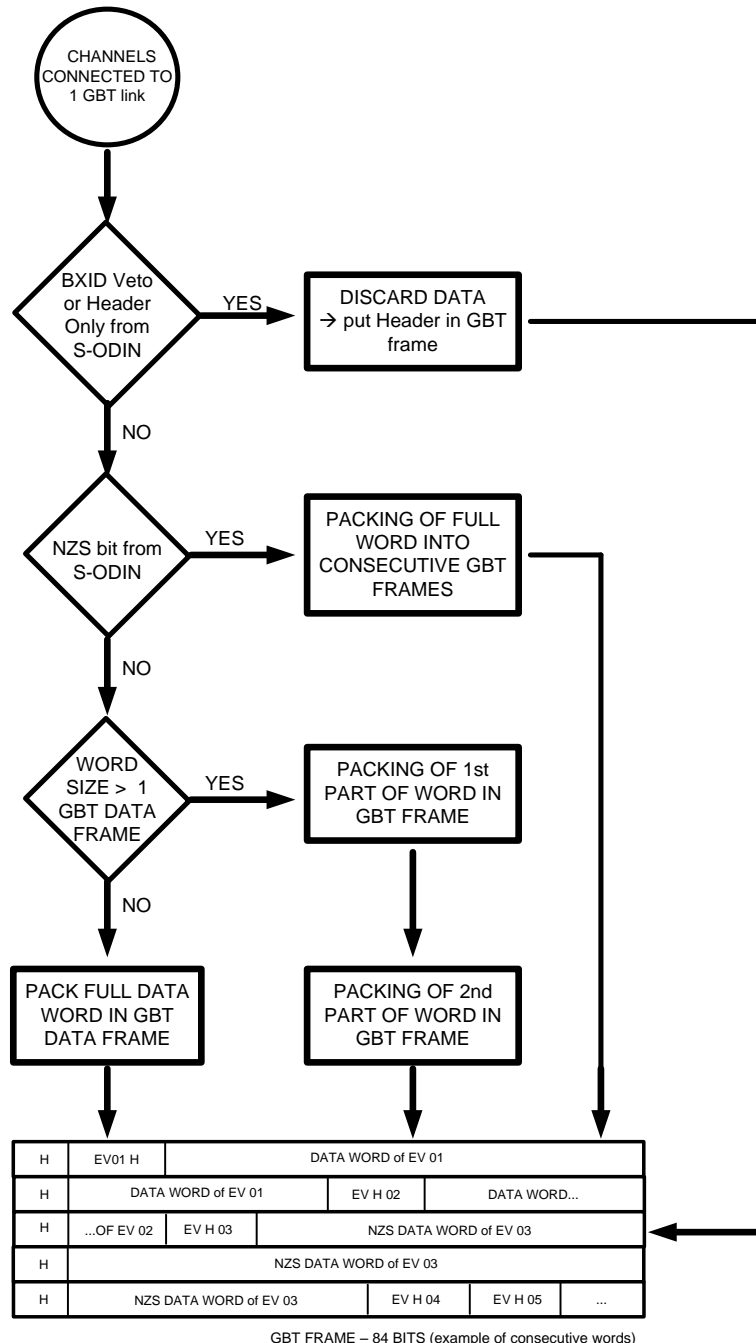
Some resynchronization mechanisms envisaged:

- 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

# Example of readout mechanism

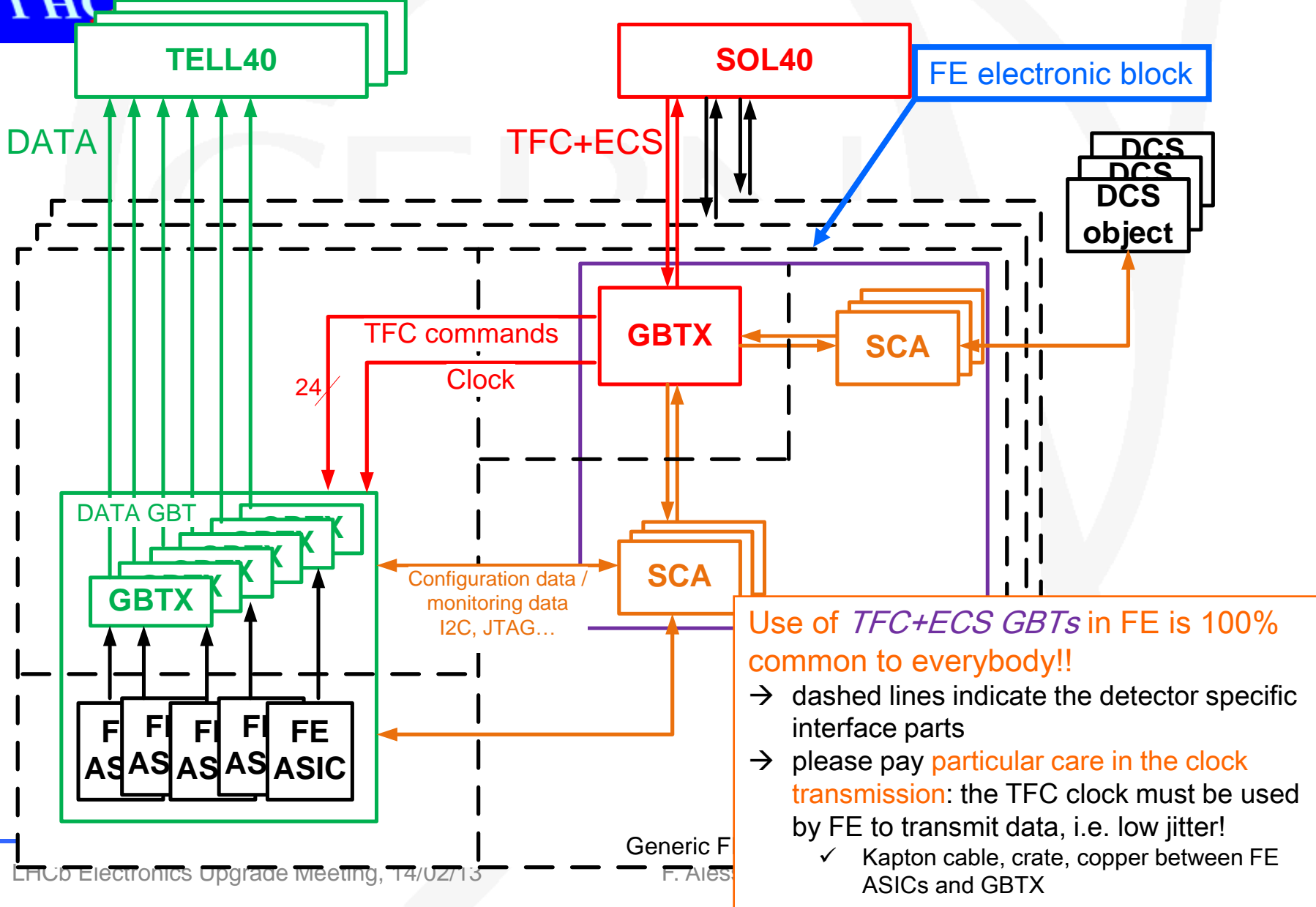


This scheme implies that the GBT bandwidth is exploited to its maximum:

- Pack of events consecutively independently from GBT boundaries
- Needs buffering to keep events stored waiting to be sent off
- Header needs more information than just BXID
  - ✓ length of word, type of word...
- Needs a well-defined truncation scheme and boundaries

GBT WORD 1  
 GBT WORD 2  
 GBT WORD 3  
 GBT WORD 4  
 GBT WORD 5

# How to decode TFC in FE chips?

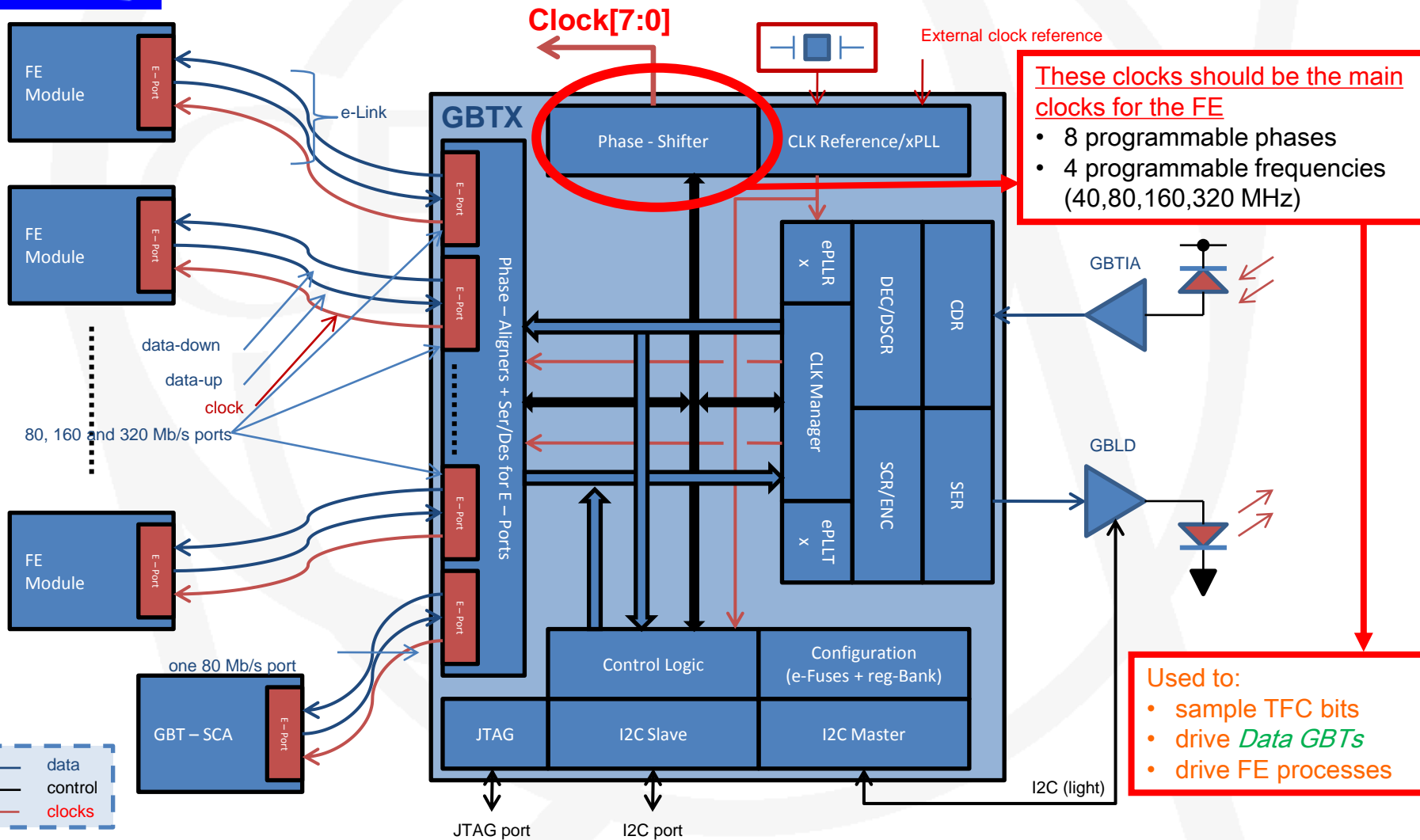


Use of *TFC+ECS* GBTs in FE is 100% common to everybody!!

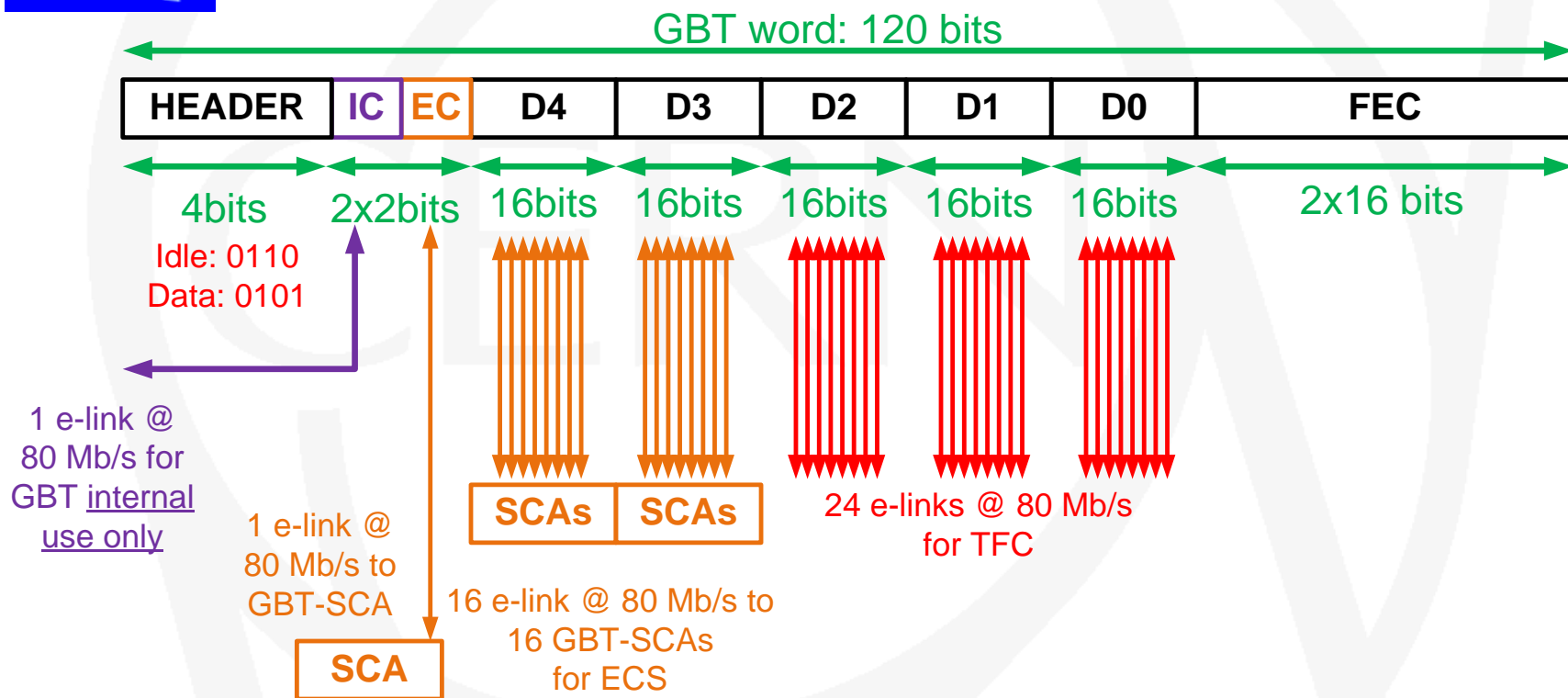
- dashed lines indicate the detector specific interface parts
- please pay **particular care in the clock transmission**: the TFC clock must be used by FE to transmit data, i.e. low jitter!
  - ✓ Kapton cable, crate, copper between FE ASICs and GBTX



# 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

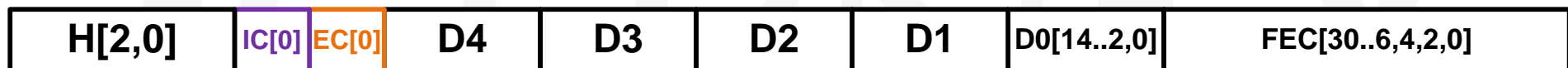
- 24 e-links @ 80 Mb/s dedicated to TFC word:
  - ✓ use 80 MHz phase shifter clock to sample TFC parallel word
- TFC bits are packed in GBT frame so that they all come out on the same clock edge
  - ✓ We can repeat the TFC bits also on consecutive 80 MHz clock edge if needed

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

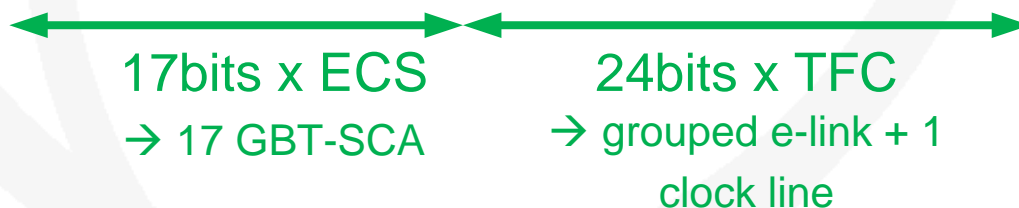
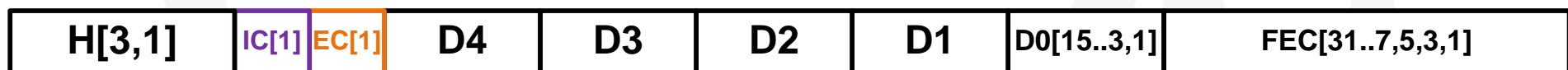


# Words come out from GBT at 80 Mb/s

lsb second, even bits



msb first, odd bits

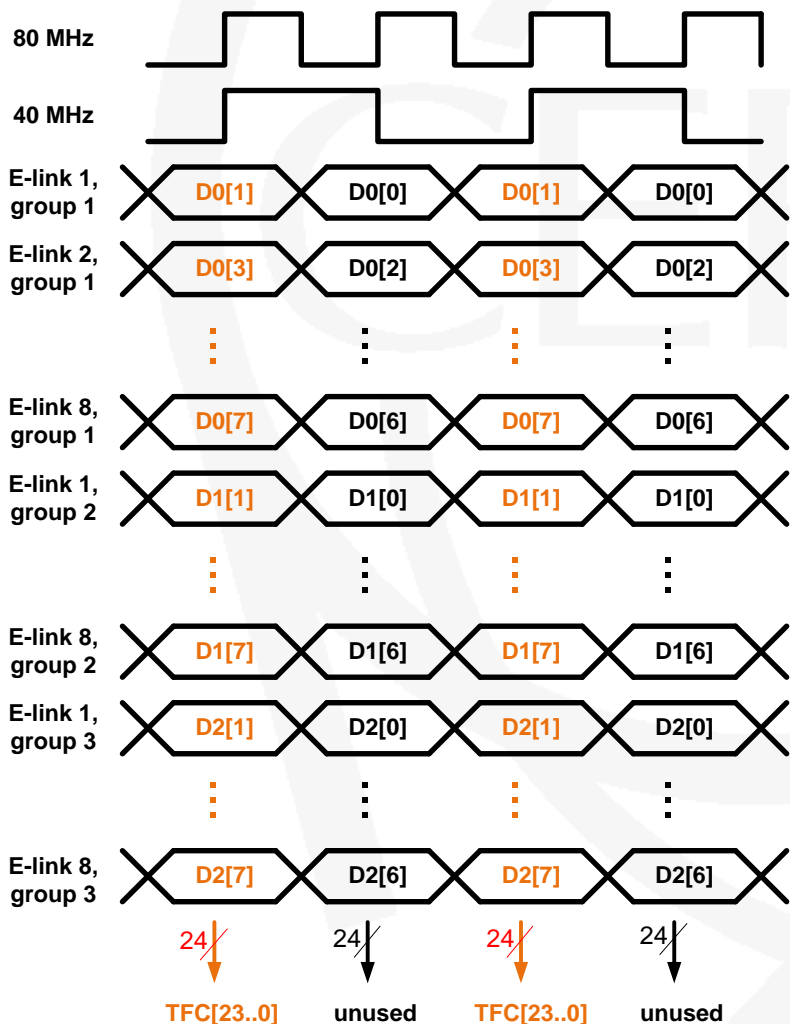


In simple words:

- 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

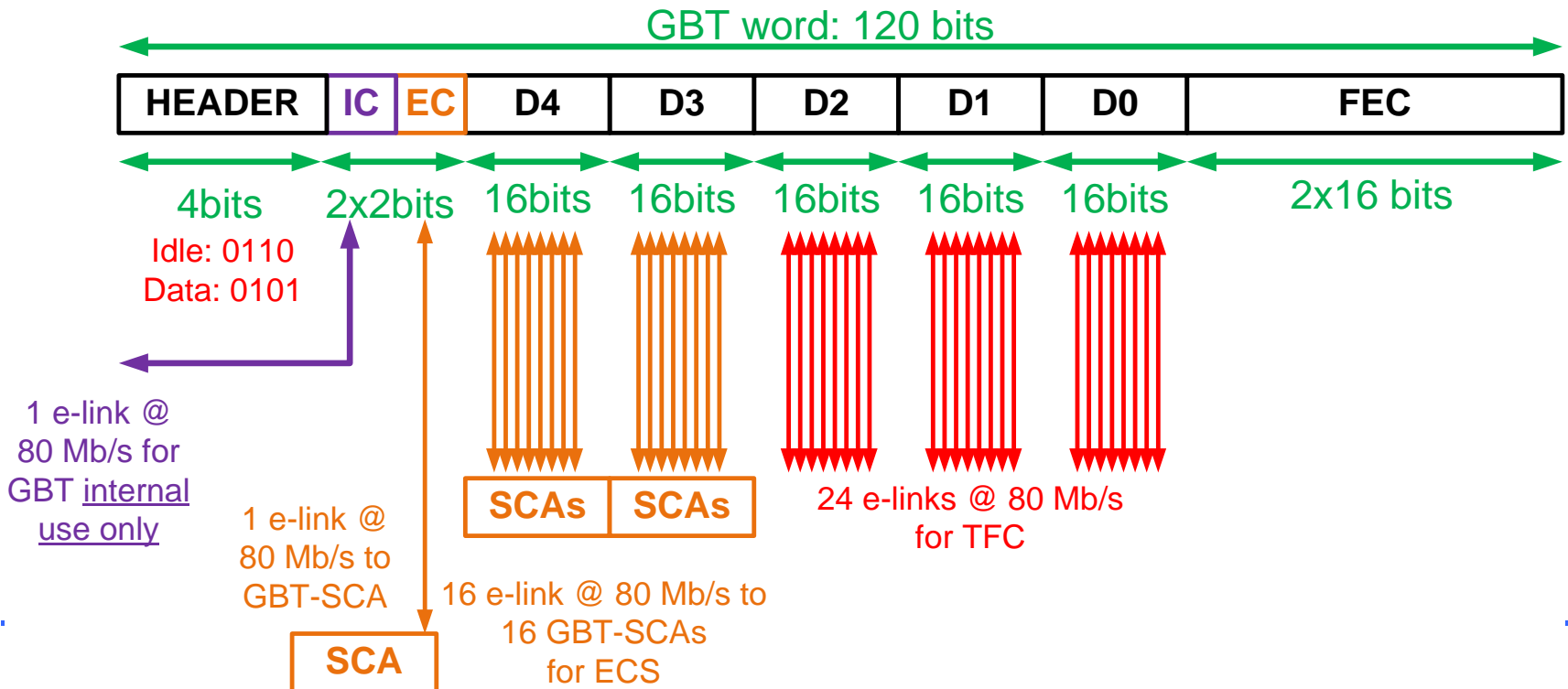
- ✓ would that help?
- ✓ FE could tie logical blocks directly on GBT pins...



# 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!







# Now, what about the ECS part?

Note that:

- Many FE chips can be connected to same GBT-SCA
  - ✓ Provided they use the same bus (and same protocol)
  - ✓ Provided we have a proper addressing scheme at the FE side (and its mapping)
- Or (viceversa) many GBT-SCAs can be connected to the same FE chip
  - ✓ If needed to have more than 80 Mb/s for each FE chip ECS... (really?!?)

To make this work, the only thing we need is to be able to:

*.. drive the right FE chip at the right address ..*

*.. with the right GBT-SCA ..*

*.. with the right protocol for the chosen bus ..*

*→ SOL40 will do that in the most flexible way possible*

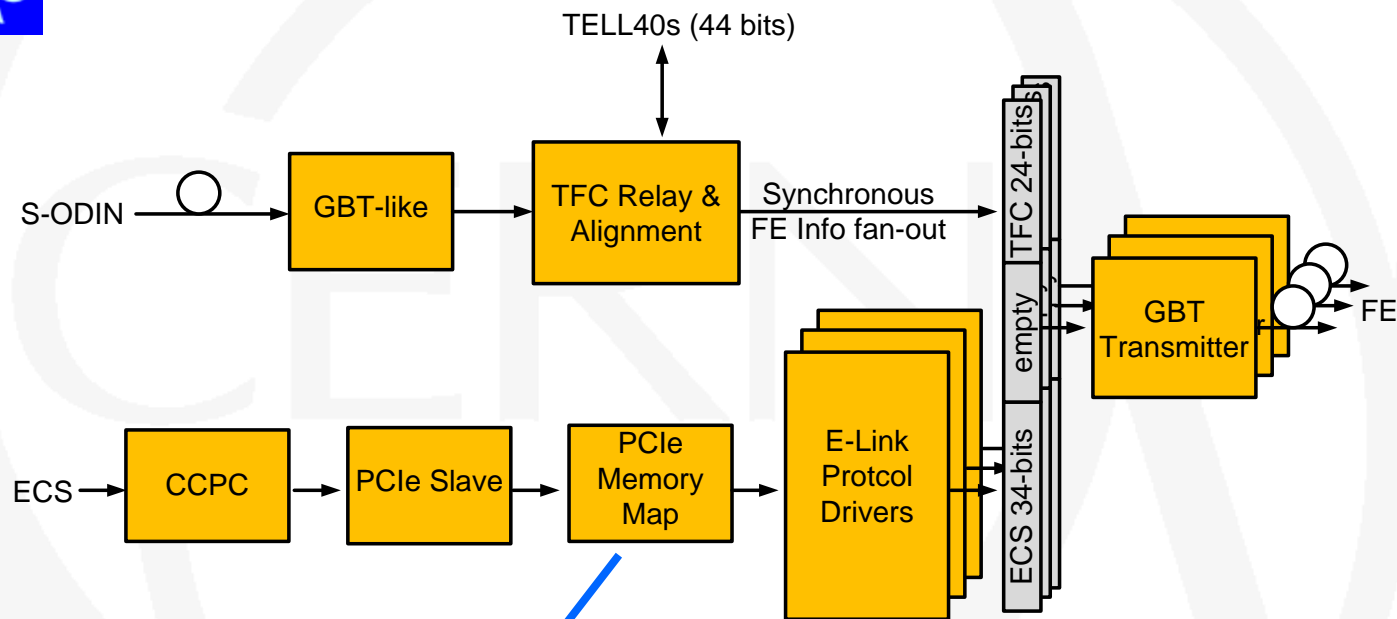
*✓ driving the GBT-SCA through the downlink*

*→ ECS (control pc) must provide:*

*✓ the mapping of the FE*

*✓ the mapping of GBT-SCA*

# 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*

# Usual considerations ...

*TFC+ECSInterface has the ECS load of an entire FE cluster for configuring and monitoring*

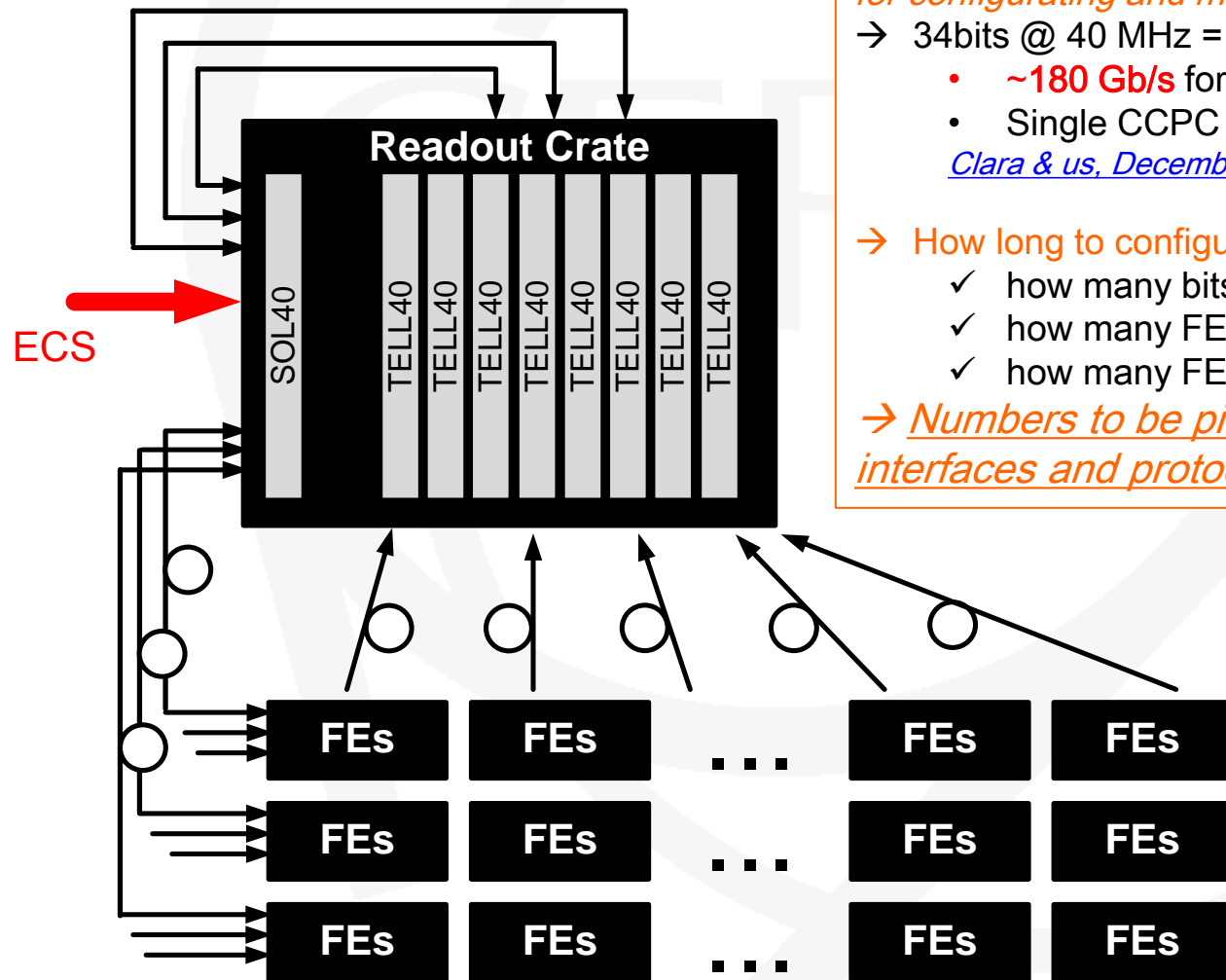
- 34bits @ 40 MHz = **1.36Gb/s** on single GBT link
  - **~180 Gb/s** for full TFC+ECSInterface (132 links)
  - Single CCPC might become bottleneck...

*[Clara & us, December 2011](#)*

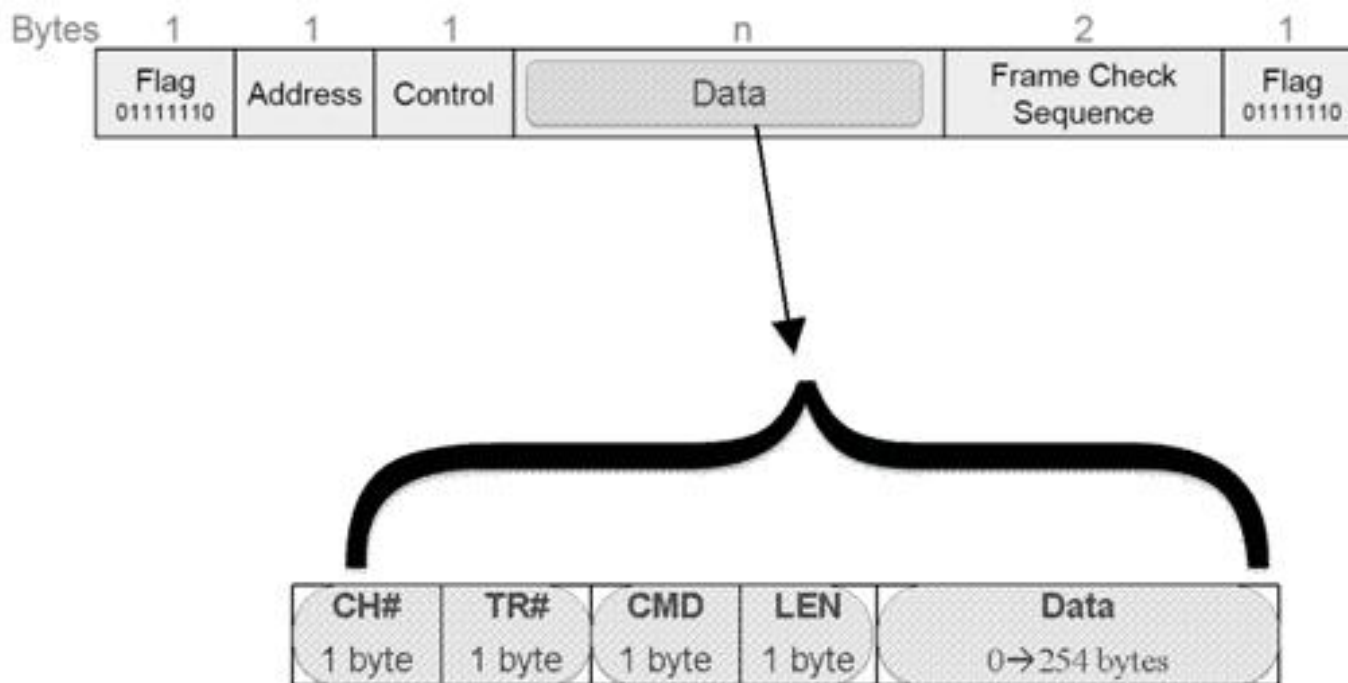
→ How long to configure FE cluster?

- ✓ how many bits / FE?
- ✓ how many FEs/ GBT link?
- ✓ how many FEs / TFC+ECSInterface?

→ *Numbers to be pinned down soon + GBT-SCA interfaces and protocols.*

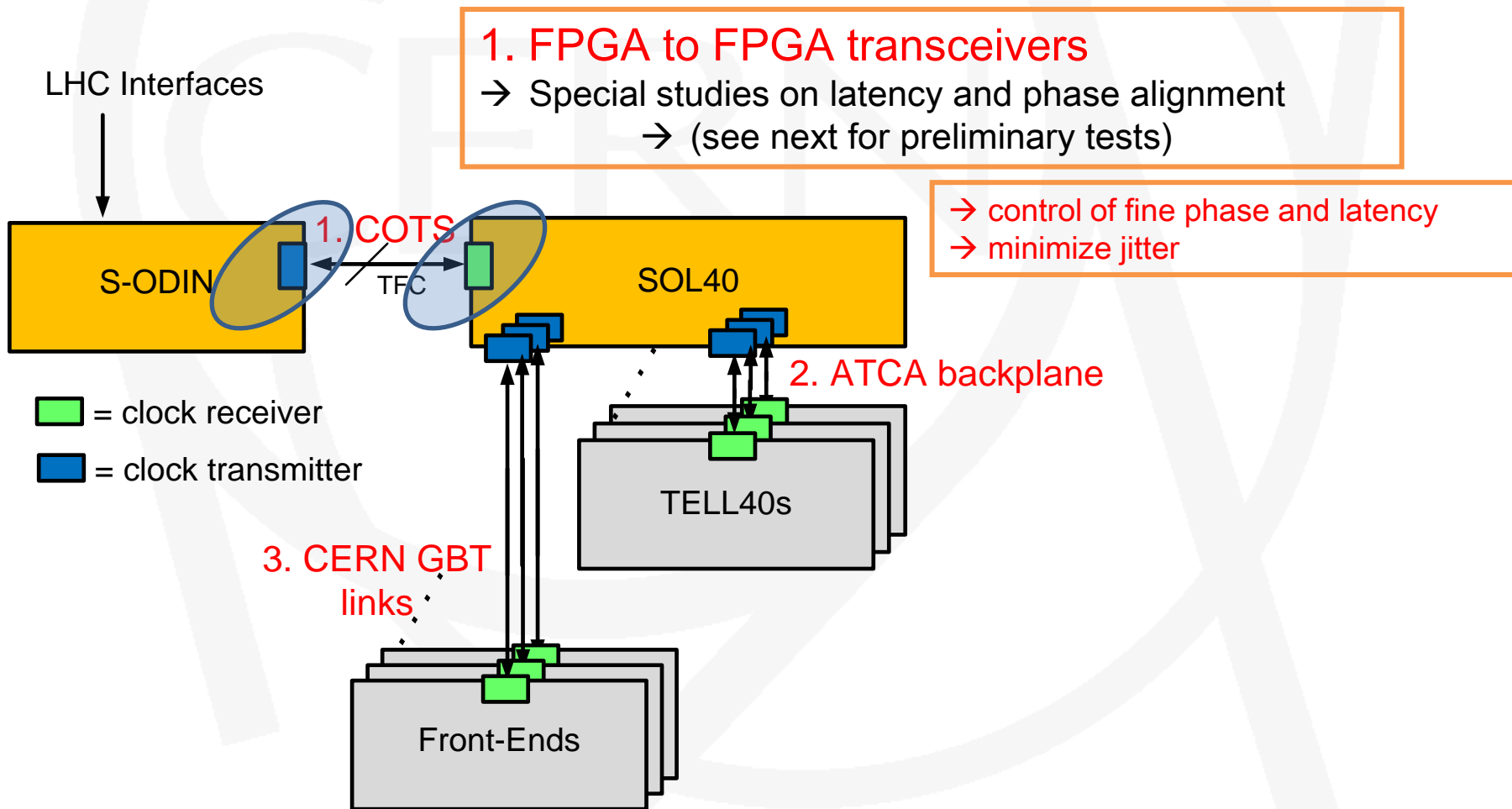


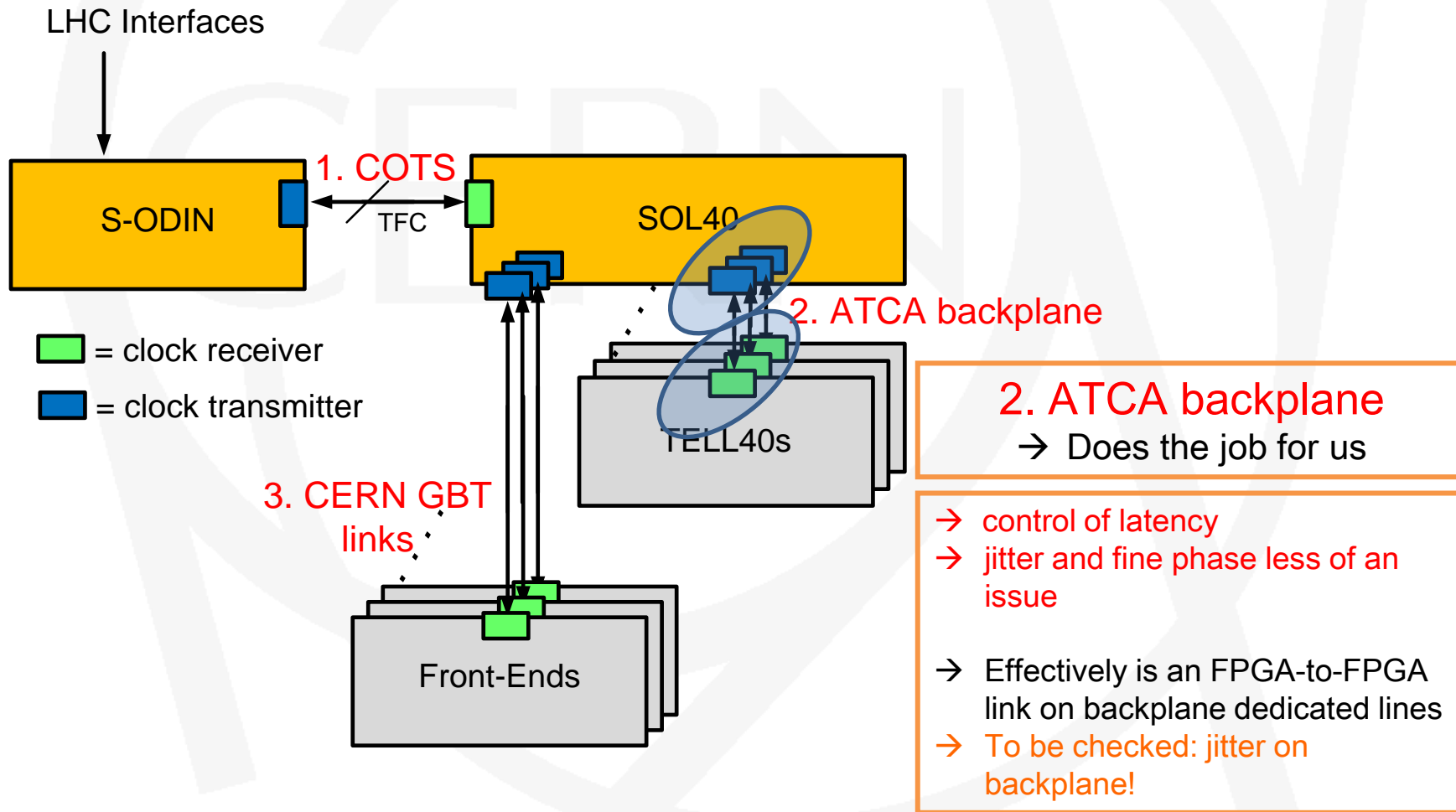
## Extra slide - GBT-SCA Packet

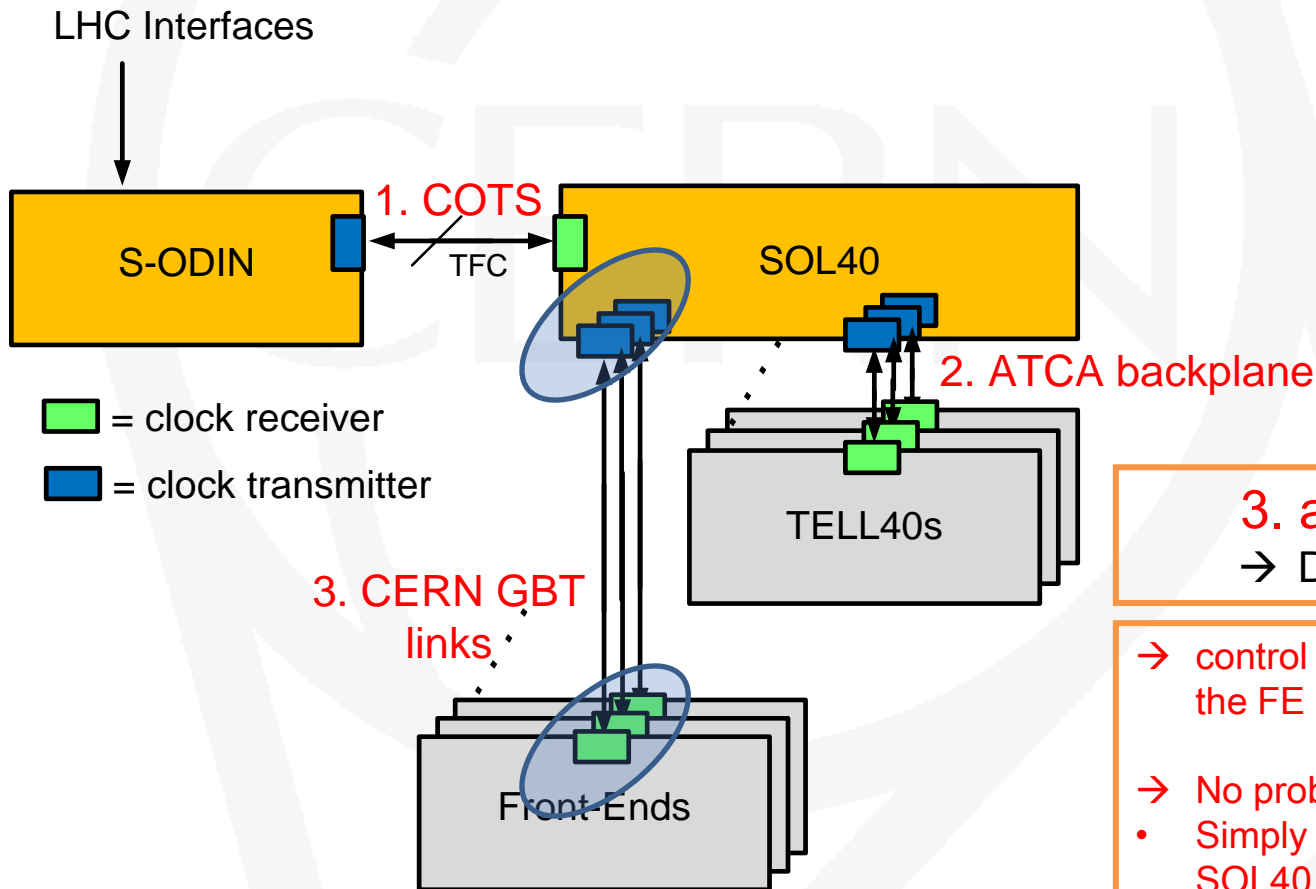


Easy: the GBT-SCA has everything set up (on paper...)

- Only thing is to build proper protocol
- FE designer selects the bus + addressing scheme, we built a generic entity in SOL40 to drive it.







### 3. at the FE: GBT

→ Does the job for us

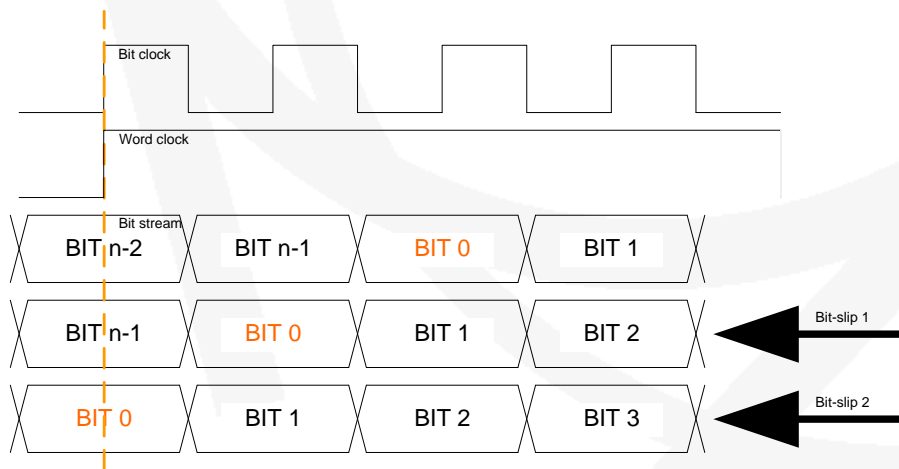
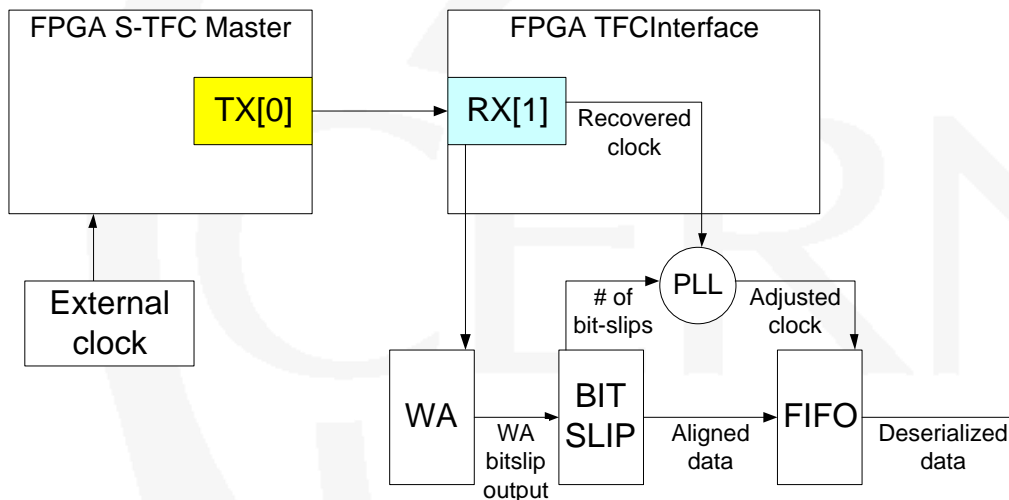
→ control of fine phase + latency at the FE + minimize jitter

→ No problem in ECS monitoring

- Simply decoding GBT protocol in SOL40 FPGA

→ No need of fine phase or latency control for ECS.

# Latency and phase control



- Investigate with ALTERA and Jean-Pierre **two years ago**
- Trick is to use **the bitslip management already in FPGA**
- ✓ **Issue: need 8b/10b encoding**
- Set a reference value
- Bitslip the word by the difference between the measured value and the reference value to re-align the words

- First implementation seems to work
- Stress tests various clock loss/recovering situations
- Estimation of maximum delay in recovering



The current system operates in a powerful mixture of *push and pull protocol* controlled by ODIN :

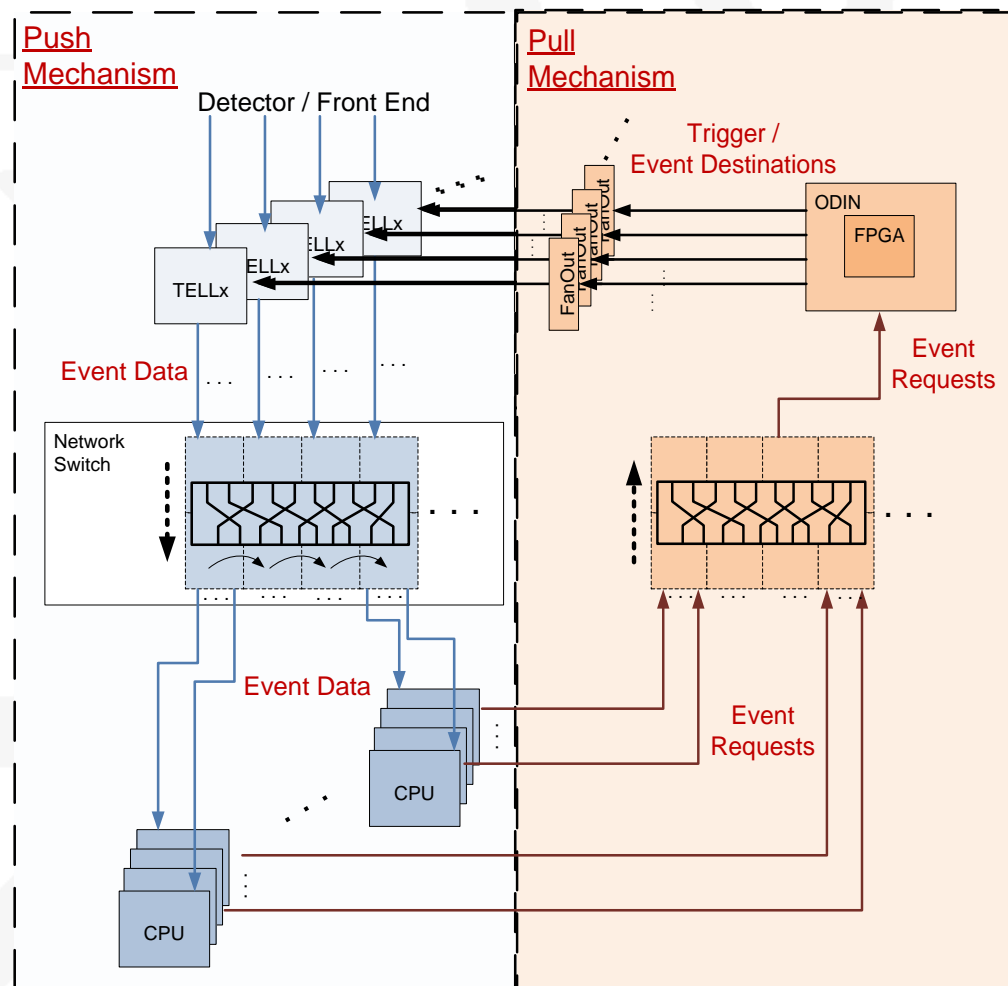
- Asynchronous pull mechanism
- “Push” driven by trigger type and destination command

→ [Note: LHCb-PUB-2011-023](#)

→ 4 years faultless operation

*Similar proposal for upgrade*

→ Comments in backup





# Event Destination and Farm Load Control

## Central FPGA based implementation

- Extreme reliability, flexibility, speed, controllable latencies

## → Central event packing control

- Different trigger types and destination types
- Variable MEP packing factor

## → Dynamic rate regulation as function of farm rate capacity

- Accounts for statistical variations in processing time

## → Dynamic handling of farm nodes in-flight

- Processing blockages, failures, interventions
- All impacts on rate capacity handled automatically
- As soon as nodes are recovered, included automatically in-flight by event request mechanism

## → Minimal event loss and accurate dead-time counting

*Contrary to conventional pull scheme, this is robust against event request packet losses*



# Event Destination and Farm Load Control

## Buffer requirement trivia

- Readout boards: ~1.5 MEPs per link
- Network: Some naïve assumptions
  - Rate: 30 MHz
  - MEP packing factor 10 → 3 MHz MEPs and 3 MHz MEP Requests
    - Current ODIN can handle 1.8 MHz of MEP Requests (ODIN <-> FARM is 1 GbE...)
  - Event size 100 kB → 1 MB / MEP
  - Farm nodes 5000 → 600 MEPs/node/s → 1.7ms / MEP
  - Switch subunit sharing resources: 50 links / subunit → 100 subunit
    - 30 kHz of MEPs per switch subunit
    - Every 1.7ms, 50 MEPs to juggle with → <buffer> = O("50 MB")
    - True need of memory depends on statistical variation of HLT processing time and "local farm derandomizer"
- Farm nodes: few MEPs in local derandomizing buffer

In our view, this looks like a straight-forward implementation...



# S-ODIN data bank

**S-ODIN transmits a data bank for each accepted event in a MEP**

→ Run number, event identifier, orbit number, bunch identifier, UTC time, event type, trigger mask, bunch crossing information

+

**S-ODIN data bank and LLT data bank is merged**

(reminder: LLT is in same board as new S-ODIN)

→ Info about timestamp, trigger type, bxid, trigger decision...

- Mostly like now

→ Will need at least 10GbE connection directly to FARM

- what about 40GbE...? ☺
- has to allow bandwidth partitioning as well
  - In fact «several» 10GbE ( $n \cdot 10\text{GbE} \dots$ ),

→ reduced bank size for local tests

- No LLT for instance



## 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



# TFC fits well on xTCA and Marseille hardware

TFC development work at CERN apart from hardware testing and high level control

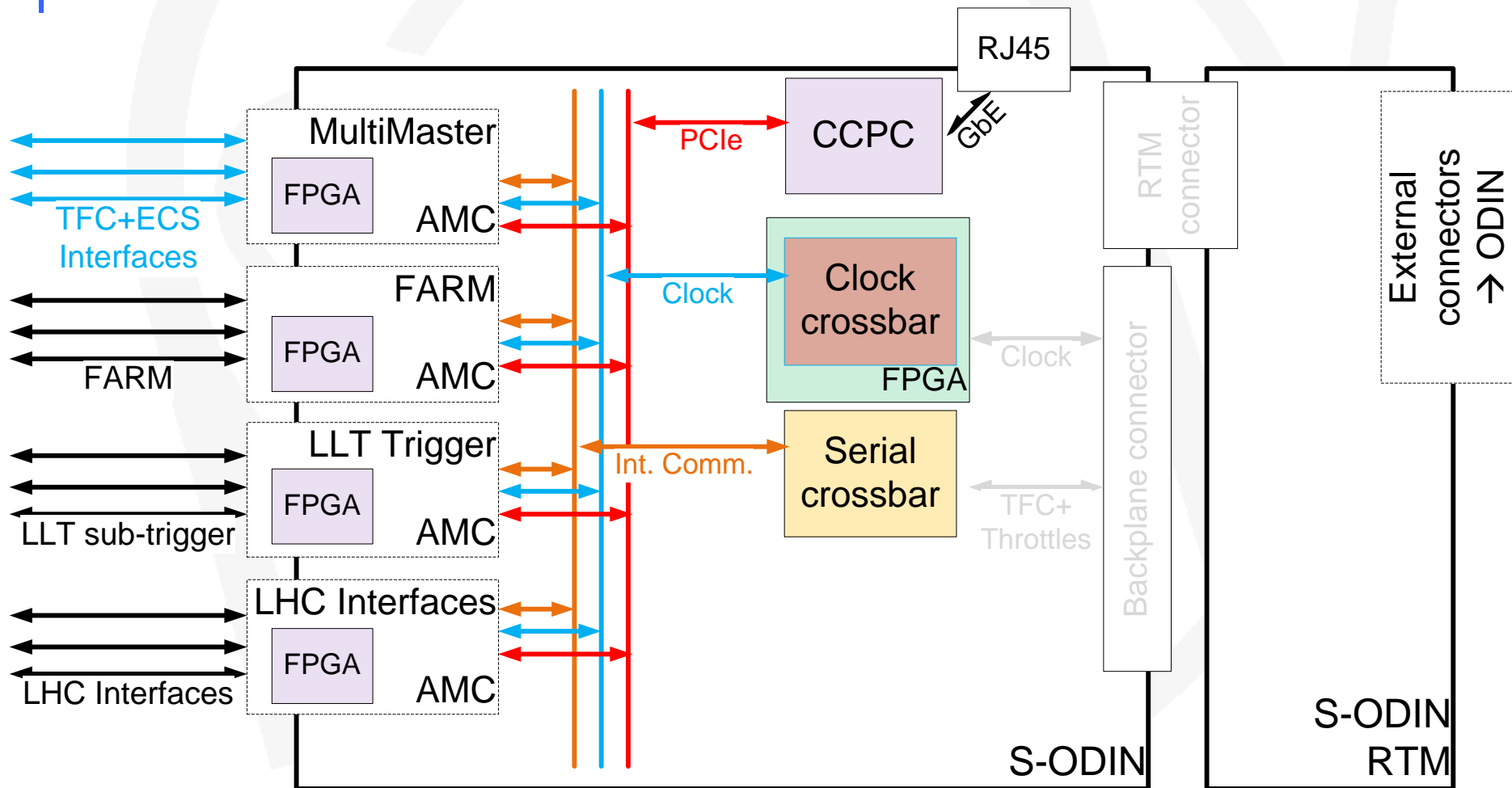
- ✓ All FPGA code (TFC logic including trigger, link protocols, board control)
- ✓ Special mezzanine(s) (Clock reception, LHC Interfaces, TTC, etc)
- ✓ Relay logic for TFC+FE Interface including optional ODIN logic for stand-alone activities
- ✓ Challenges:
  - TFC+ECSInterfaces with optional ODIN need to be ready and produced in quantities before any SD test setup
  - Validation of clock phase control and reproducibility of latency and jitter
  - Validation of transmission protocols S-ODIN+SOL40 and SOL40+FE
    - Go for 8b/10b decoder to start with
      - ❑ Simple, as part of the ALTERA decoding
    - Validate GBT with TFC+ECS to FE information

## Considerations

- ✓ In **ATCA Dual Star technology**, boards in Hub1 and Hub2 talks to all the boards in the crate
  - TFC+ECSInterface as Hub1 board
- ✓ S-ODIN located near the LHC interfaces (clock, machine timing and control information).
  - Bi-directional fibres connections to TFC Interface
- ✓ If no backplane, BE connection via TFC+ECSInterface must be done via fibres
  - How many receivers/transmitter per mezzanine/ how many mezzanines?

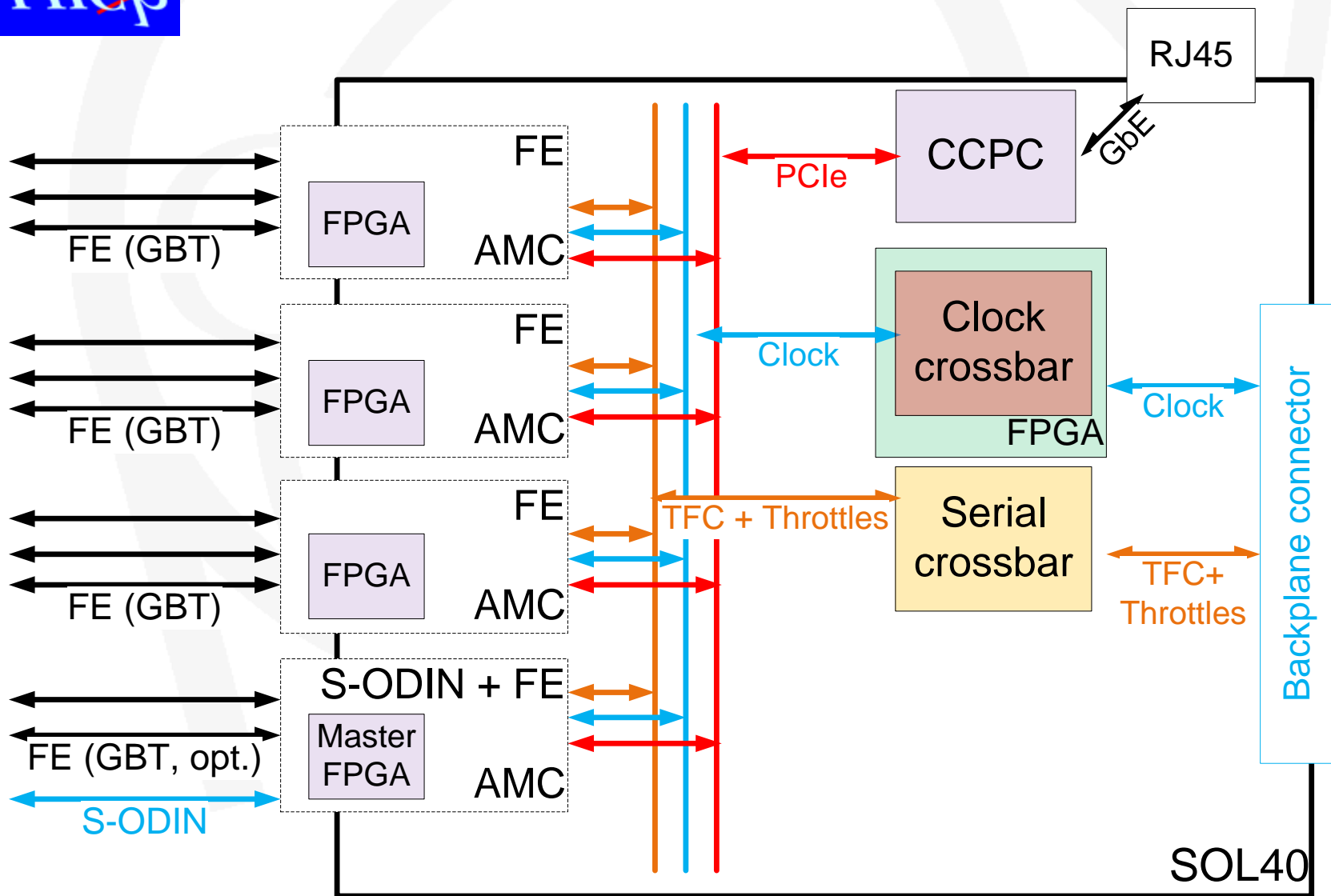


# S-ODIN on Marseille's ATCA board





# TFC+EC S Interface on Marseille's ATCA board

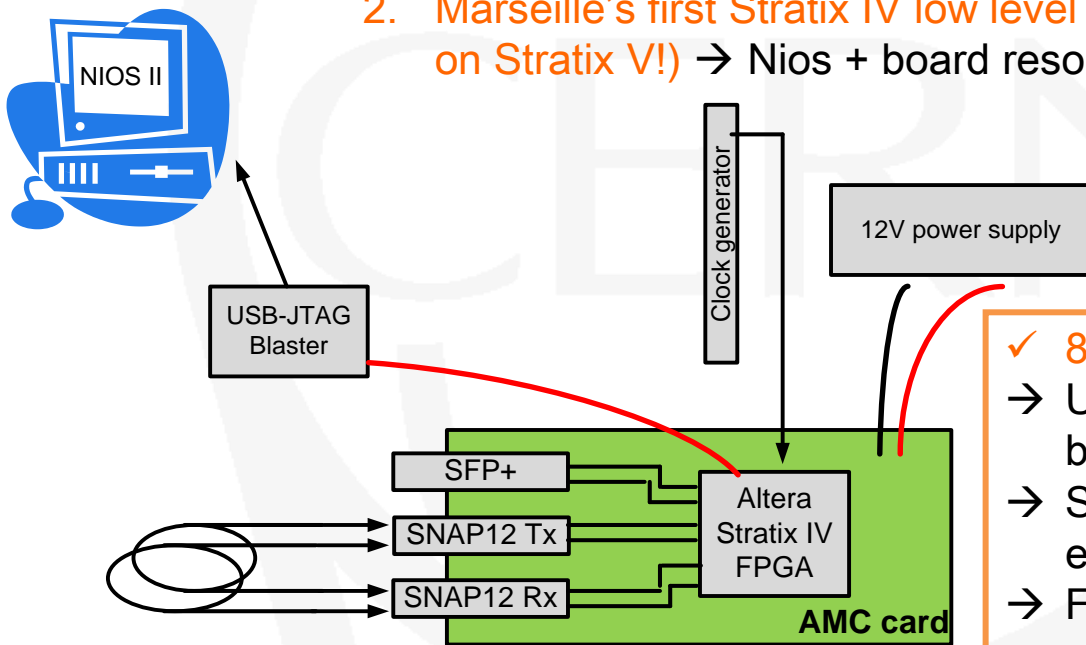




# Clock distribution and phase/latency control

First preliminary tests on phase/latency control using:

1. Marseille's first AMC prototype with ALTERA Stratix IV
2. Marseille's first Stratix IV low level interfaces (will need re-validation on Stratix V!) → Nios + board resources interfaces + thanks to Marseille team!



✓ **8b/10b protocol: no problem**  
 → Using «word alignment» from Altera GX buffer + «deterministic latency»  
 → Simply add Ctrl word for the 8b/10b encoder: 2bits more  
 → Full reproducibility upon power-up and resets and reconfiguration







✓ **FPGA-to-FPGA GBT protocol: ok, but needs special frame alignment**  
 → No deterministic latency if no special words are sent!  
 → Needs a special word (10 bits minimum) at power-up/after reset/after reconfiguration for the GX buffer to realign to the beginning of the frame + «deterministic latency»

- First preliminary tests were ok, *but needs more validation under stress tests!*



# Lab work and planned (first) tests

Some lab work has **already** started:

1. Set up a S-TFC test stand at the pit  **Done!**
2. Get Marseille's AMC board first prototype  **Done!**
  - First version of firmware with GBT protocol in it (developed in Marseille)
  - First version of control software in NIOS (developed in Marseille)
    - One word of acknowledgment for the work done by Jean-Pierre and Frederic Hachon in Marseille
3. Get acquainted with the hardware  **Done!**
4. Send first trigger words  **Done!**
  - 8b/10b protocol: using ALTERA features
  - GBT protocol: not possible to use ALTERA features
5. Stress tests to validate latency control  **Done (but will need revalidation on Stratix V + more tests!)**
6. Stress tests to validate phase control  **Done (but will need revalidation on Stratix V + more tests!)**
7. Implement first version of TFC+ECS firmware with functionalities presented here
  - Including first simple version of S-ODIN
8. Develop first single-AMC test stand board with firmware + software to control the hardware
  1. Need dedicated simulation framework
    - Already developed in 2009 (see <https://indico.cern.ch/conferenceDisplay.py?confId=65307>)
    - Needs mostly adaptations ...
  - Work to control (software) the single AMC card

