DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

SALSA ASIC DSP features and interface with EPIC readout chain

Damien Neyret (CEA Saclay IRFU) for Sao Paulo University and CEA IRFU teams

EPIC DAQ/electronics WG meeting

25/01/2024

SALSA specifications and architecture

DSP features

Input interfaces

Clock and sync command management

Implementation of EIC time structure

Output data

Summary of open questions

SALSA CHIP TARGET SPECIFICATIONS

Versatile front-end characteristics

- Dedicated to MPGD detectors and beyond
- 64 channels
- Large input capacitance range, optimized for 50-200 pF, reasonable gain up to 1nF
- Large range of peaking times: 50-500 ns
- Large gain ranges: 0-50 to 0-5000 fC
- Large range of input rates, up to 100 kHz/ch with fast CSA reset (limit assumed for EPIC: 25 kHz/ch)
- Reversible polarity
- Front-end elements can be by-passed

Digital stage

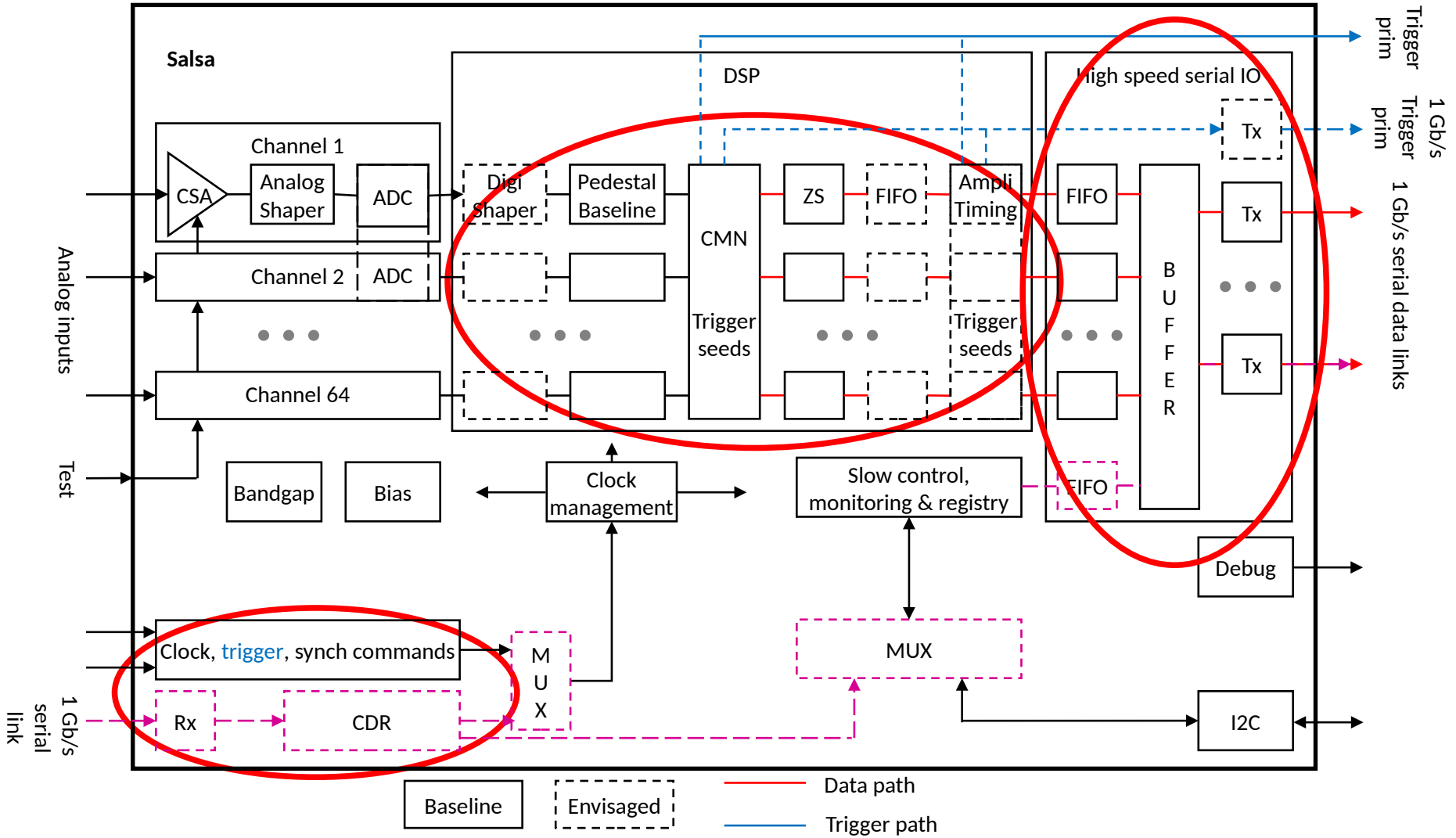
- Fast sampling ADC for each channel on 12 bits (> 10 effective bits) at up to 50 MS/s
- Possibility to double rates by coupling pairs of channels
- Integrated DSP for internal data processing and size reduction, treatment processes to be selected according to user needs
- Continuous readout compatible with streaming DAQ foreseen at EIC, triggered mode also available
- Several 1 Gb/s output data links

General characteristics

- ~1 cm^2 die size, implemented on modern TSMC 65nm technology
- Low power consumption ~ 15 mW/channel at 1.2V
- Radiation hardened (SEU, TID), working at 2T magnetic field

PRELIMINARY CHIP ARCHITECTURE

DSP, input and output interfaces discussed today



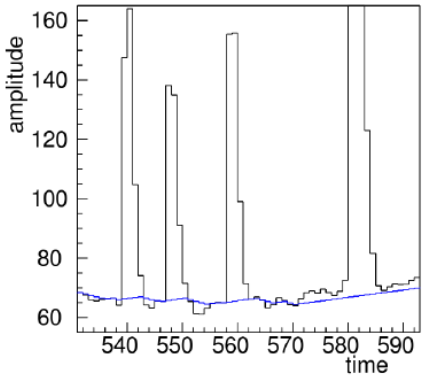
DSP DATA PROCESSING, PRELIMINARY VERSION

General remarks

- Data processing, reduction and formatting from ADC values to output links
- Each process can be deactivated individually by user
- Process parameters through ASIC registers
- Part of codes from SAMPA chip
- Considered processes still under study, suggestions welcome !

Baseline corrections

- Pedestal subtraction with fixed value per channel
- Common mode correction to reduce common noise impact, based on median value of samples of all channels for each sample time
- Baseline slope following algorithm



Plot from SAMPA doc

Digital shaping

- Cancellation of signal tail if necessary

Zero suppression

- Keeping samples above fixed thresholds, possibly neighbor ones in time and in channel number
- Possibility to drop too small set of samples above threshold

DSP DATA PROCESSING, PRELIMINARY VERSION

Feature reconstruction

- To further reduce data flux by extracting reconstructed data
- For instance peak finding algorithm, with extraction of amplitude + time + TOT
- Possibility to keep raw data from time to time (to be defined) to monitor reconstruction

Trigger management

- Samples selected when trigger signals received, with configurable latency
- Followed or not by zero suppression, feature reconstruction, etc...
- Not used in EPIC

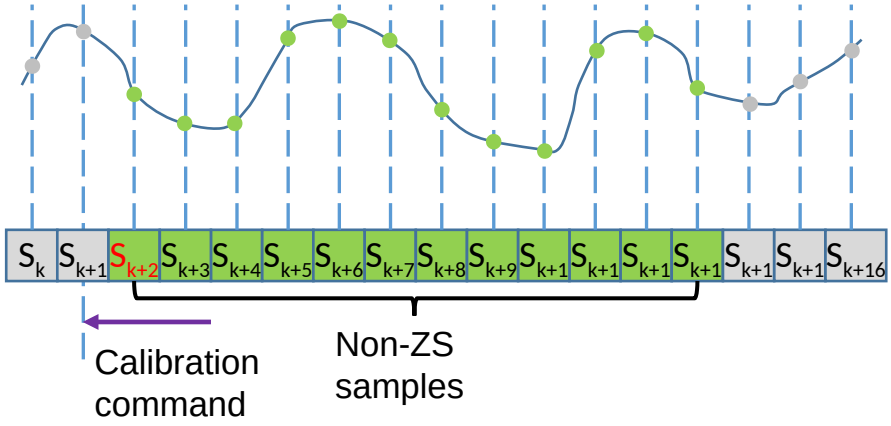
Trigger generation

- Trigger primitives generated when samples above threshold, with conditions on number of samples, multiplicity, etc...
- Possible to reduce latency by placing trigger generation early in the processing chain, for instance before common mode correction.
- Nature of trigger primitives to be defined (logic signal, data on specific fast link, etc…)
- Is this feature needed for EPIC ? What requirement on latency ?

DSP DATA PROCESSING, PRELIMINARY VERSION

Calibration data

- Specific fast commands to generate calibration data of several kinds
  - non-ZS data
  - test pulses
  - etc...



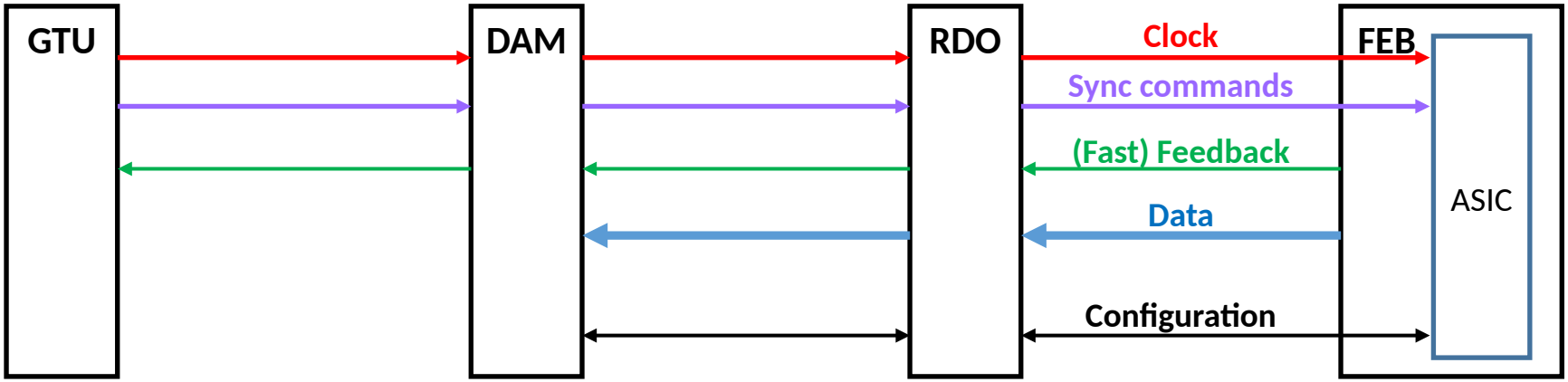
Information data

- Monitoring data like chip configuration, internal chip status (currents, voltages), environmental data (temperature, radiation, etc...)
- Software scaler histogram for occupancy per channel evaluation
- Also slow-control outputs

Data formatting and buffering

- Output data flux segmented in packets with different kinds of data: sample and reconstructed data, calibration data, monitoring data, error messages
- Packet identification by numbering and timestamps
- Each packet buffered and transmitted through one of the Gbit/s links
- Size of data words to be defined (32 bits ? 48 bits ?). Constraints from EPIC DAQ chain ?

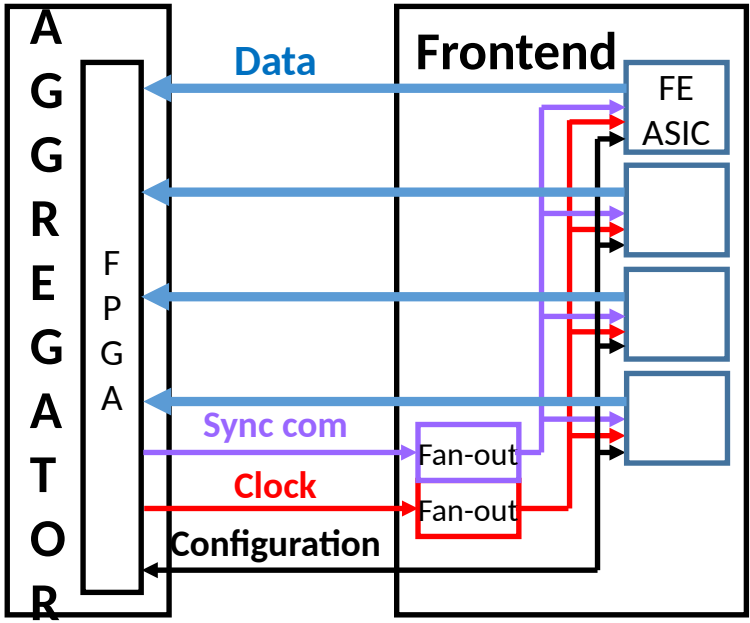
Communications interfaces : logical view



- Heterogeneous nature of communications
  - → Unidirectional / bidirectional
  - → Synchronous / asynchronous
  - → Low latency / reasonably slow
  - → High rate / low rate
  - → Continuous / on-demand
- May result to a number of heterogeneous physical interfaces
  - → Especially at the ASIC level

ASIC interfaces options : split clock-command model

- Important number of heterogeneous external interface signals
  - → Dedicated differential IO lines for clock and for commands
    - May need a high fidelity fanout ASIC for clock quality
  - → Serial data line used to deliver chip state information
    - Data & feedback
  - → Requires additional configuration interface : *e.g.* I2C
    - Bi-directional
  - → Optionally a differential trigger input
  - → Optionally a differential trigger primitive output

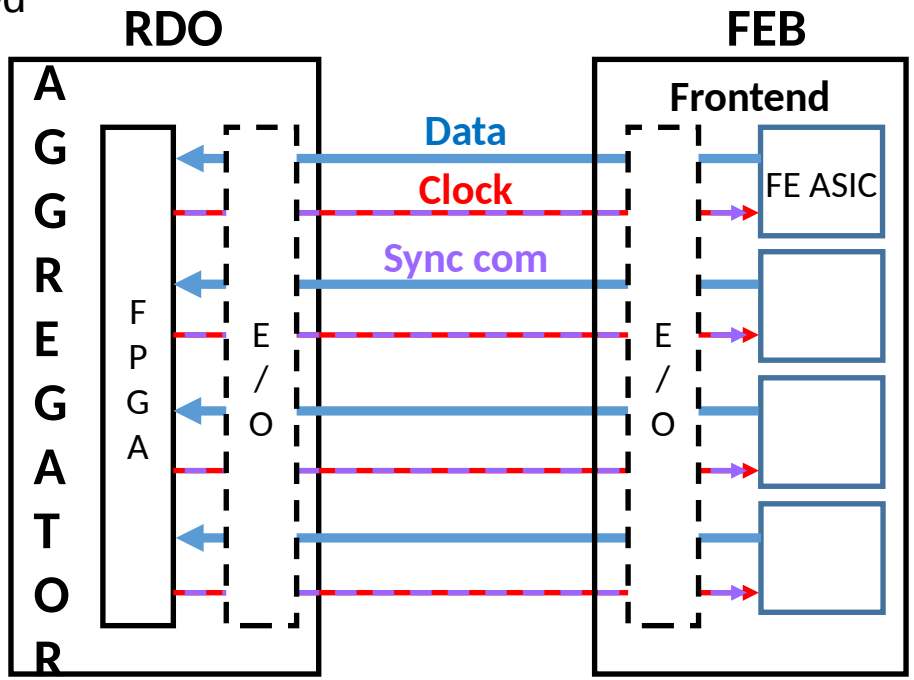


- Integration in ePIC environment is not easy
  - → > 1.5 T magnetic field : power distribution problematic
  - → Stringent space : RDO location not clear
  - → Low radiation : some immunity has to be guaranteed

- No on-FEB intelligence
  - → Electrical interface with RDO : signal integrity above few meters
  - → Optical interface with RDO : almost static bidirectional I2C is not well suited for conventional optical transceiver
- On-board intelligence with interfacing embedded clock-command-data world to ASIC
  - → Expect up to 50% power increase per channel : extra cooling
  - → Expect bulky power distribution cables or bulky magnetic field tolerant DC/DC convertors
  - → May need to take some care for avoid rare radiation effects

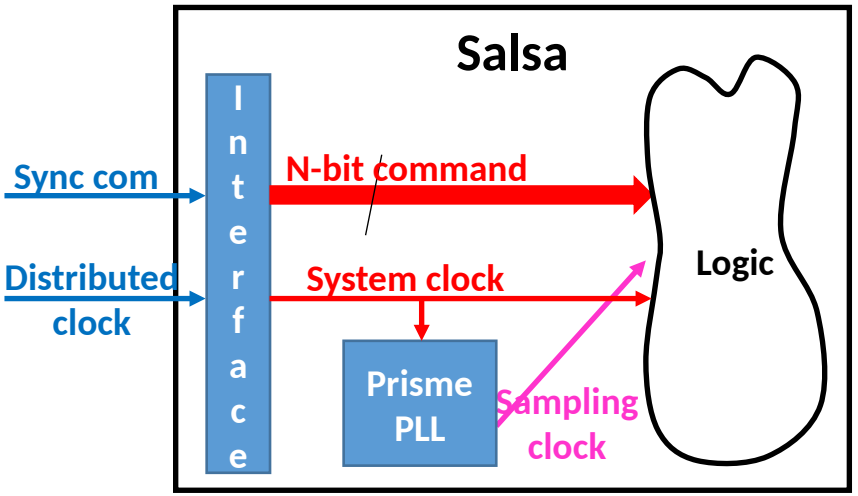
ASIC interfaces options : embedded clock model

- Minimal external interfaces
  - → Single differential Rx line for clock, commands, configuration and monitoring
  - → Differential Tx lines for data, fast feedback, but also for configuration, monitoring
    - 4 ASIC pins in a minimal configuration
- Integration in ePIC environment might be simplified
  - → On-FEB FPGA may be avoided
  - → Commercial optical transceiver can be used
    - Encouraging news on radiation tolerance of Samtec FireFly transceivers
- Counterpart
  - → Expect 15% increase of power / channel due to optical transceiver
  - → Requires cooperation from aggregator during initialization
    - Dedicated part of RDO firmware with possible hardware adaptation



Clock and command interface: some definitions

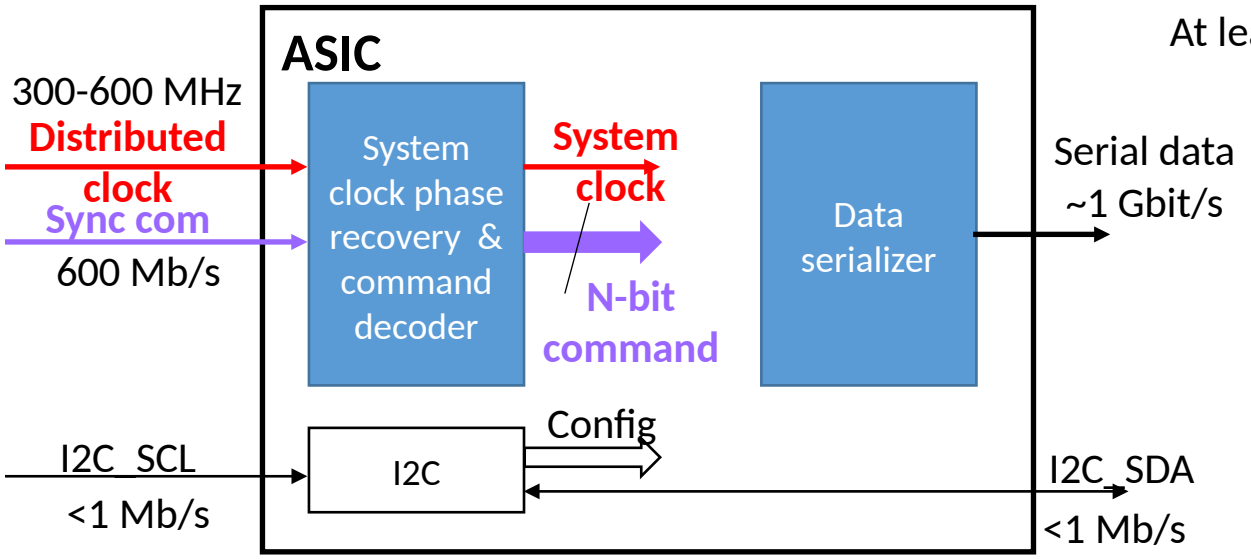
- System clock : common to all sub-detectors in an experiment → 100 MHz BX clock at EIC
  - → Data packets are coarsely tagged by a time stamp from this clock domain
    - Salsa maintains only N (*e.g.* 12) LSB-s of the time stamp
    - Fixed relationship between time stamps in Salsa and complete time stamps in aggregator
- Sampling clock : used by Salsa to acquire signal samples
  - → Derived from system clock
    - Known frequency and phase relationship exists between the sampling clock and system clock
  - → Used as time stamp reference for samples
- Distributed clock : delivered to Salsa
  - → System clock is recovered from distributed clock
    - Frequency and phase
  - → Sampling clock is produced from system clock
    - Frequency and phase



Clock and command interface: split model 1

- Can receive sync command at every system clock cycle
  - → Distributed clock to be a multiple of system clock to decode N-bit synchronous command
    - SDR mode: N * SysClock
    - DDR mode: N * SysClock / 2
  - → System clock phase needs to be recovered
    - Either a dedicated “IDLE” command with non repeated “comma” sequence
    - Or a unique bit pattern

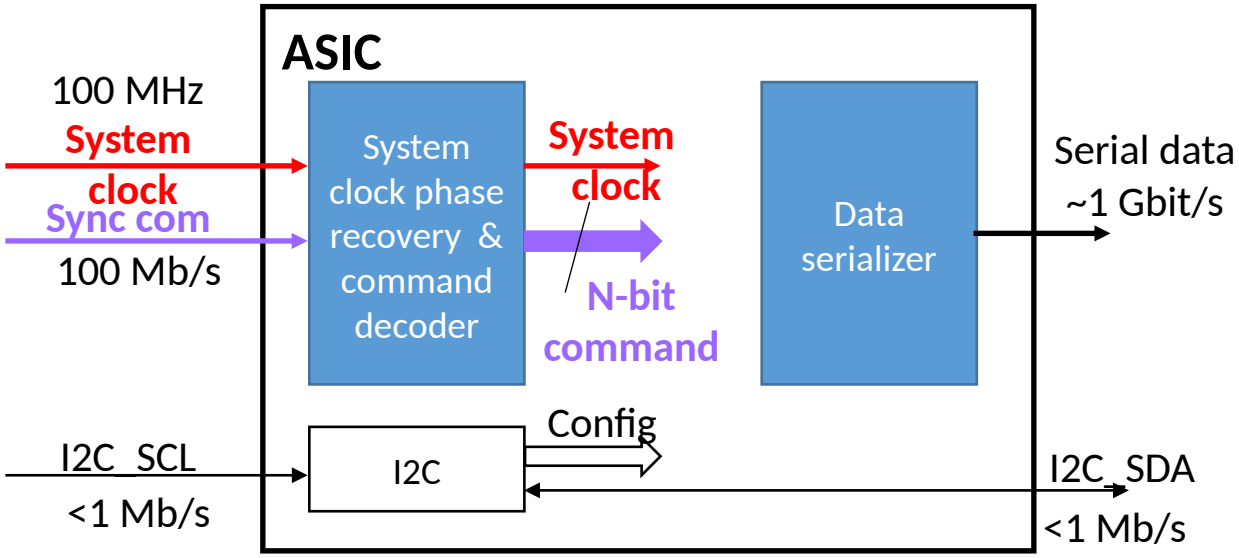
EIC example: system clock: 100 MHz Bx clock
600 Mb/s data with 300 MHz DDR interface
6-bits per bunch crossing clock
Some bits reserved for phase recovery
At least 16 synchronous commands



Clock and command interface: split model 2

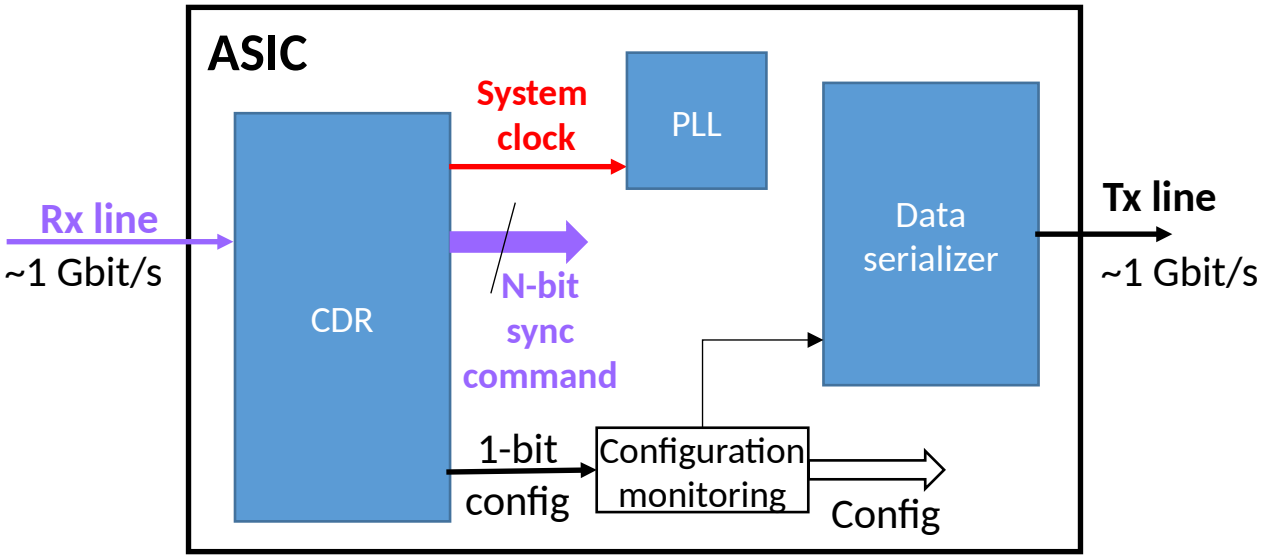
- Bit-by-bit aggregation of commands
  - → Distributed clock is the system clock
  - → Requires N clock cycles to construct N-bit synchronous command
    - Arrival time of the very first bit determines command timing
  - → May require dedicated trigger input to accept triggers at every system clock cycles

EIC example: system clock: 100 MHz Bx clock
8-bit sequence determines 6-bit command
8 bunch crossings needed
Synchronous commands can be received every 8th BX
150 commands per EIC revolution

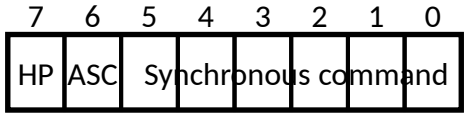


Clock and command interface: embedded clock model

- Single encoded line for Clock, SynCmd, Trigger inputs and I2C



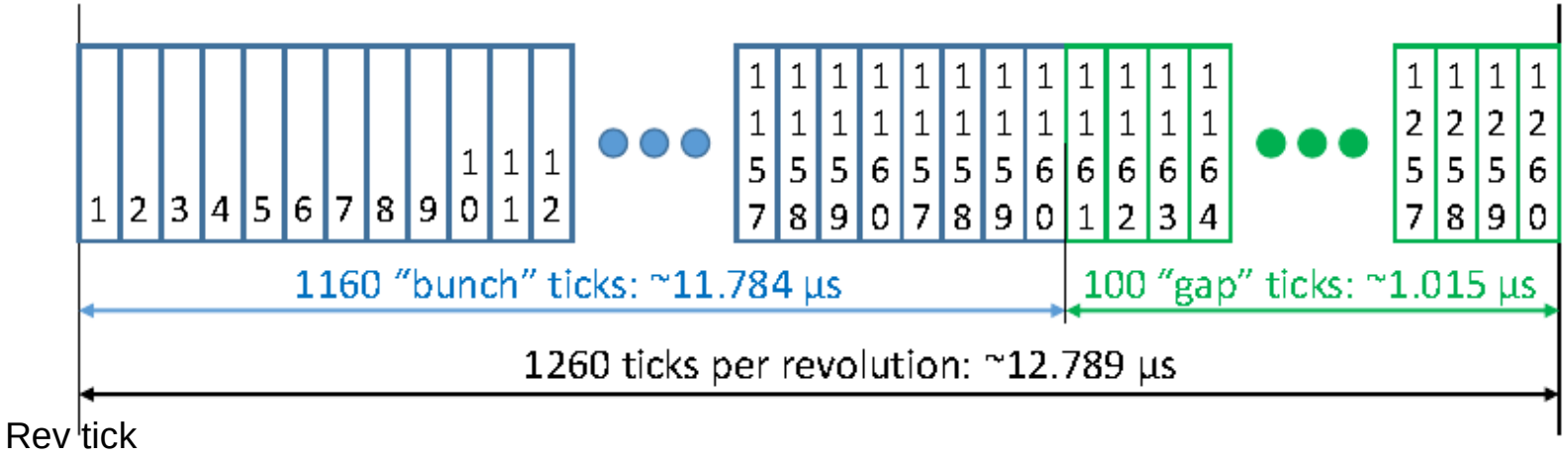
EIC example:
1 Gbit/s Rx line
100 MHz embedded clock
8-bits per bunch crossing including:
1 bit for horizontal parity
6 bits for sync commands
1 async slow control bit (20-bit async SC command at 5 MHz)



- Minimal external interface: a single diff Rx line + at least one diff Tx line
  - → Simplest case: only 4 pins (Rx_p / Rx_n + Tx_p / Tx_n) to communicate with the chip
  - → Parallel configuration of ASICs possible : fast startup and recovery time
- Relatively complex initialization phase requiring collaboration from the transmitter
  - → Clock recovery phase
  - → Data transmission phase

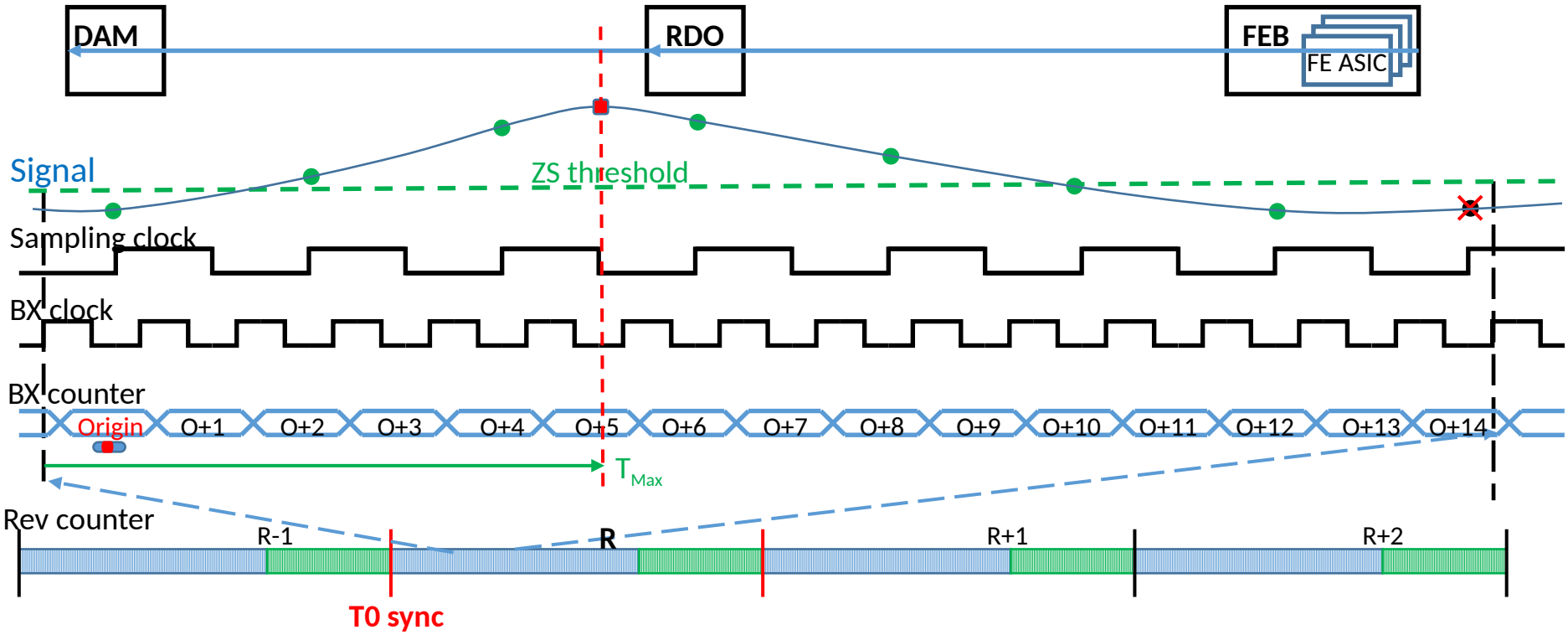
EIC beam structure and bunch crossing timing

- Beam structure repeats every ~12.7886 μs
  - → Revolution frequency: ~78.195 kHz
- 1260 clock ticks in each revolution
  - → Clock period: ~10.14968 ns, frequency 98.52525 MHz
  - → 1160 "bunch" ticks *approximately* marking *potentially* active intervals with particles
    - *Approximately*: electron beam structure is much more stable compared to hadron beam structure
    - *Potentially*: there can be 290 or 1160 active bunches with particles
  - → 100 "gap" ticks without particles



- Assume a stable clock can be derived from Machine with this frequency
  - → This clock or its (sub)multiples is distributed to frontend electronics as System clock
    - With bunch-phase recovery mechanism
    - Used for bunch level synchronization, coarse timestamp bookkeeping, serial communication and possibly timing measurements.

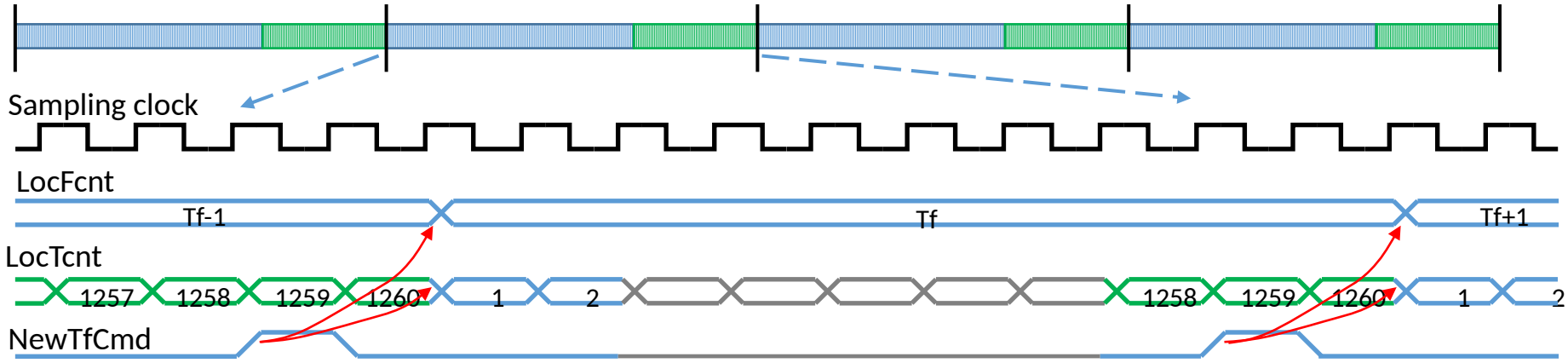
MPGD data: signal shape sampling with ADC



- ADC sampling clock is derived from bunch crossing clock
  - → Known frequency and phase relationship exists between the two clocks
  - → T0 time sync command: reset timestamp counters, align clock phases → fully known state
- Signal association with originating bunch crossing possible
  - → Max time or arrival time measured in sampling clock unit → give time distance from last T0 time synchronization
  - → Originating bunch crossing identified by time distance from last T0 time sync, converted in system clock

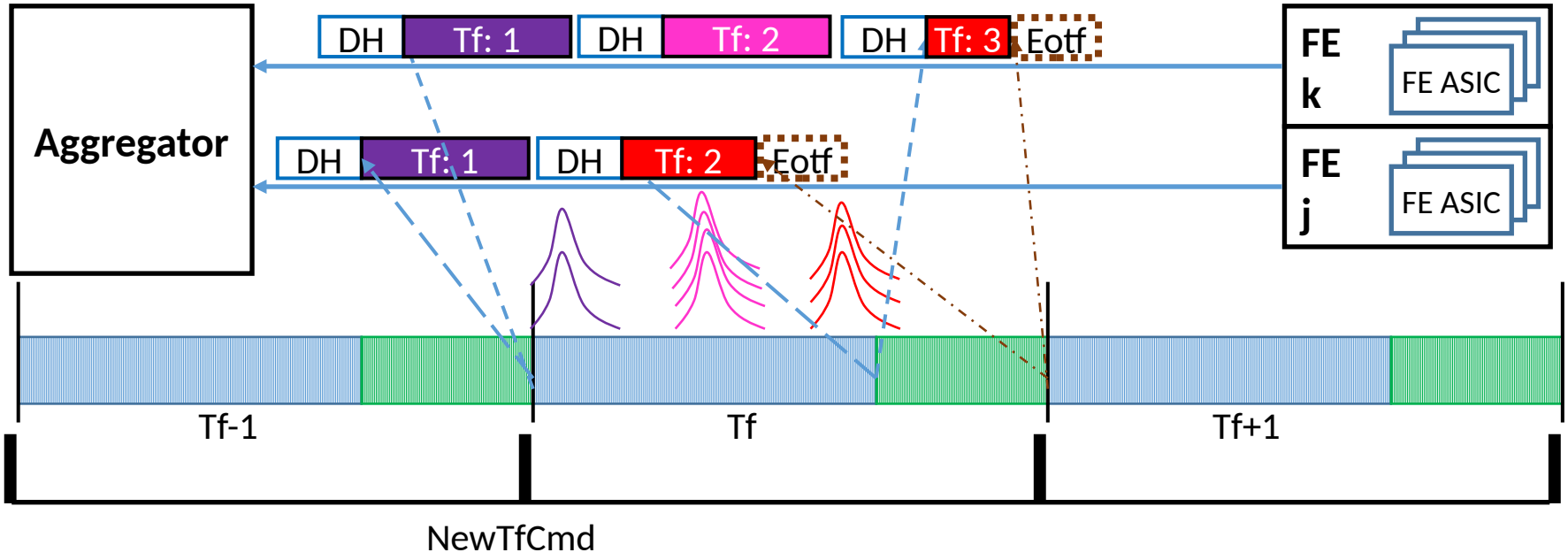
Synchronization with EIC and EPIC : time frames

- Assume there are periodically new time frames
  - → Always identified by an external signal, DAQ generates new time frame commands (NewTfCmd)
    - BC0 @ LHC
    - RevTic @ EIC ?
  - → ASIC T0 time synchronization when new time frame starts
- N-bit counter of system clock within a time frame in ASIC → local counter (LocTcnt)
  - → N is large – enough to avoid roll over within the time frame
    - 12 @ LHC; 11 @ EIC
- M-bit counter of time frames in ASIC → local time frame counter (LocFcnt)
  - → M small – enough to avoid roll over during the data collection in aggregator
    - 4 bits should be enough, but can be any reasonably small number of bits



Synchronization with EIC and EPIC : time frames

- DAQ (RDO or DAM ?) and ASIC perform actions on reception of NewTfCmd
  - → Verify that NewTfCmd at the expected LocTcnt - synchronization
  - → Reset time counters (LocTcnt and sampling clock one)
  - → Increment LocFcnt
- Optionally ASIC can send short end of time frame packet to DAQ (Eotf packet)
  - → Includes local time frame counter LocTcnt at end of last time frame and whatever additional handy information
  - → Eotf packet sent after all data packets belonging to last time frame has been sent
    - Assist aggregator in data collection : no data expected after Eotf packet for that frame



SALSA SYNCHRONOUS COMMANDS, PRELIMINARY VERSION

Context

- Commands received from DAQ in synchronization with system clock (98.5 MHz)

Resynchronization commands

- **COUNTERSYNC**: reset all counters, including event and time frame ones, empty FIFOs
- **LINKSYNC**: ASIC send know packets to the output serial links for synchronization

Data taking management commands (those useful for EPIC)

- **TOSYNC**: new time frame → reset clock and packet counters, realign clock phases, and send specific information packet like Eotf one
- **STARTREAD**: activate sample reading by DSP
- **ENDREAD**: deactivate sample reading by DSP, finish to process remaining samples, then send a specific packet when no more sample are remaining
- **CALIB0...N**: generate calibration data of type N
- **INFO0...N**: generate information packet of type N
- **TRIG**: generate trigger (not used at EPIC)

STRUCTURE OF DATA OUTPUT

- **Output data stream based on packets**
  - Different kinds of packets
  - Each identified by kind of packet, timestamp, packet number
  - Each sent to one of the output data links

- **Physics data packets**
  - Header + sample values + reconstructed values
  - Includes timestamps, chip address and channel numbers
  - Detail of format under discussion
  - Word size (number of bits) to be defined, constraints from EPIC DAQ ?

- **Calibration data packets**
  - Same format as physics packets + kind of calibration data
  - Triggered by sync commands

- **Information packets**
  - Carry any kinds of information and monitoring data: ASIC configuration, slow-control feedback, environmental informations, channel counting rates, etc...
  - Triggered by sync commands or slow-control

- **Error packets**
  - Information packet generated when error or warning encountered in ASIC

OPEN QUESTIONS TO EPIC DAQ

Input interface

- If split clock-command model: what max time gap between 2 sync commands ? 800MHz distributed clock if command at every system clock ? 400 MHz with DDR ?
- If embedded clock model: possibility to have specific parts in FPGA firmware RDO cards in order to implement communication protocols with SALSA ?
- Opinion of EPIC DAQ peoples about model to choose ?

Synchronization of ASIC with EIC and EPIC DAQ

- Plans of EPIC about time frames ? 1 beam revolution ? N revolutions ? How long ?
  - Should be anyway in phase with revolution ticks !
- Expected run control protocol, list of commands and actions ? Interface with ASICs ?
- Generation of calibration commands by DAQ ? Generic for all EPIC detectors or specific for each kind ?
- Generation of information request commands ?
- Trigger primitive generation: needed by EPIC ? Max latency required ? What output interface for trigger primitives ?

Output interface

- Constraints from EPIC DAQ about output format ? About packet word size ? Number of bits must be power of 2 or not ?