AGATA Global Trigger and Synchronization Hardware

1. Introduction

Data synchronization is an important aspect in the operation of the trigger and readout systems of the AGATA experiment. A high increase in efficiency with respect to current spectrometers is expected in AGATA by means of online gamma ray tracking and pulse shape analysis (PSA). Tracking and PSA require the concurrent digitization of preamplifier signals of the 36 fold segmented Ge crystals composing the array. Therefore, the design of the front-end readout and level-1 (L1) trigger in AGATA follows a synchronous pipeline model: the detector data are stored in pipeline buffers at the global AGATA frequency, waiting the global L1 decision. The L1 latency must be constant and shall match the pipeline buffer length. The whole system behaves synchronously and synchronization at different levels and in different contexts has to be achieved and monitored for proper operation of the system. In order to fix definitions, we list in Table 1 the various synchronization types that we refer to.

Type	*Description*
Sampling Synchronization	Synchronization of the detector signals with the clock phase
Serial Link Synchronization	Recovery of parallel data words from the serial bit stream.
Trigger Requests Alignment	Alignment of trigger data at the input of the trigger pipeline processor
L1 Validations Synchronization	Synchronization of L1A signal with data in the readout pipelines
Event Synchronization	Assignment of global clock and event number to data fragments in the DAQ path

Table 1: Synchronization types.

In AGATA each crystal is considered as a separate entity and from the point of view of the Data Acquisition System (DAQ), the whole detector may be considered as the aggregation of synchronized data supplied by individual crystals, possibly disciplined by a global trigger primitive.
Each crystal is composed of 36 segments and a central core contact, all individually readout.
The data from the core contact are processed for event detection and hence, a level 1 trigger request or local trigger generation. The choice between the two behaviors is done upon configuration, the former corresponding to an effective way to reduce front-end data rates in cases where anyone of the stages of the readout chain is unable to perform at the actual data throughput.

2. The Front-end Model

A columnar model of the data flow concerning each individual segment of a crystal can be represented as in fig. 1. This sort of digital pipe highlights the flow starting from the digitizer of a single channel down to the Pulse Shape Analysis farm where position estimation of gamma ray interactions is carried out.
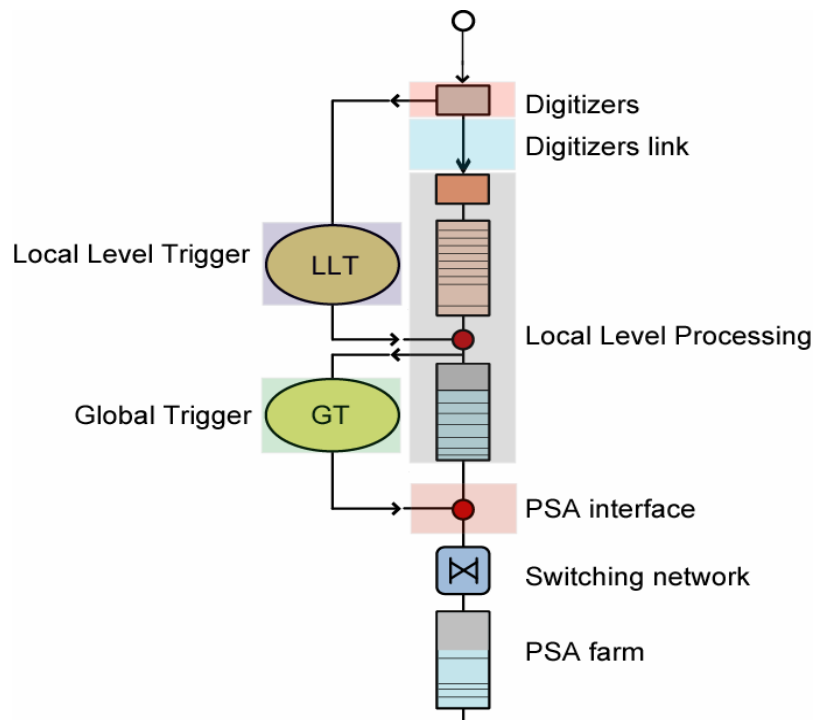
Fig. 1 – The AGATA readout column

The model shows two types of trigger interactions with the flow: a local trigger signal generated by central core processing and, possibly, a global trigger L1 accept signal generated externally from a central trigger processor. To ease the solution of the problems posed by the different synchronization levels specified before, AGATA shares a global time reference supplied by a global trigger and synchronization control system (GTS) and distributed by means of a network of optical fibres to the front-end electronics of each crystal. Fig. 2 shows the aggregate model of a front-end system and its interface to the GTS.

Fig. 2 – Model of a detector channel

The following functionalities will be described in terms of behaviour.

1: Extraction of the physical signal from the sub-detector and transformation to digital data.

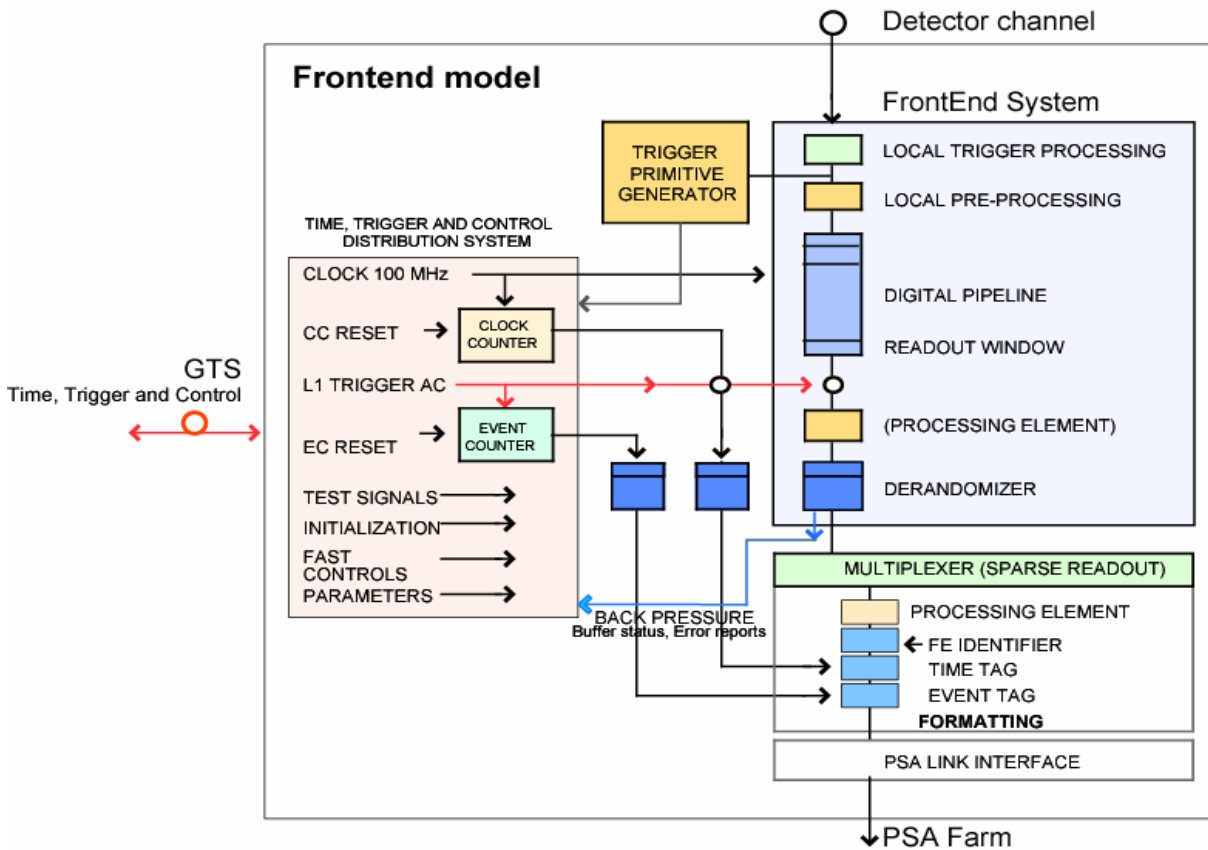The transformation includes any needed shaping or processing. If the detector is the central core contact of the crystal, the data are made available to the Trigger Primitive Generator (TPG).

2: Delay on the data during the trigger level-1 latency.

This delay can be implemented in various ways : storage with pointers management or a simple pipeline shifting synchronously with the global clock for example. The delay value will be the sum of the decision time of the trigger logic and the propagation time. Hence, the delay can vary depending on the location of the front-end.

3: Data selection and tagging.

On the result of level-1 trigger process, data corresponding to that event must be selected from the pipeline. A data frame (data recorded during several consecutive clock cycles) is to be considered and not a single data. Corresponding global clock and event numbers must be associated to those data to tag them.

4: Derandomization.

Once selected by a level-1 accept, the frame is saved in a multi-event buffer. This buffer stage is needed to allow a frequency derandomization between the level-1 rate and the readout process. The size of the multi-event buffer has to be defined in order to guarantee the smallest buffer-overflow probability possible. A FIFO-like behavior is needed for this buffer.

5: First level merge.

A given number of derandomizer buffers is attached to a merge engine that will readout data associated to a particular event. The engine produces a so called "Front End Event".

6: Front End Event formatting.

The data format of the Front End Event must include the global clock counter and event numbers to allow later data misalignment search [1]. The presence of a Front End identifier could also be required.

7: Transmission to FED

The FE Event is then sent out to the Front-end driver (FED) via a data link. The media and protocol used for this link should be a DAQ standard.

8: Monitoring of the derandomizer filling level

If the readout scheme is such that every derandomizer has the same status at the same time (after readout completion), this case is simple and the solution is straightforward (see below). If the readout scheme is such that every derandomizer can have a different status (e.g. variable event size), warning or worst status must be included to the "Front End Event" for later processing at the FED level.

9: Test facilities

To have maximum efficiency in the problem detection, they have to be implemented as high as possible in the readout chain.

10: Processing.

Data processing (e.g. lossless compression) can also be needed in the front end. Location and exact functionality of the processing is to be defined.

3. GTS Functionalities

From the logical description of the front-end operation given above it turns out that a certain number of global time referenced signals are needed. Among them:

1. common clock
2. global clock counter
3. global event counter
4. trigger controls:
   - i. Throttling of the L1 validation signal
   - ii. Fast commands (fast reset, initialization, etc.)
   - iii. Fast monitoring feedback from the crystals
   - iv. Calibration and test trigger sequence commands
   - v. Monitor of dead time
5. Trigger requests
6. Error reports

In AGATA, the transport medium of all these signals is shared by use of serial optical bidirectional links connecting the front-end electronics of each crystal with a central global trigger and synchronization control unit in a tree-like structure, thus actually merging together the three basic functionalities of synchronization distribution, global control and trigger processing.

More in detail:

1: Common clock

This is a 100 MHz digital clock supplied by a central timing unit (possibly GPS disciplined) and used to clock the high speed optical transceivers reaching the front-end electronics of every crystal. At the crystal receiving side the clock is reconstructed and filtered for jitter. The clock signals of each crystal may be equalized for delay and phase, thus accounting for different fibre lengths and different crystal locations in the array.

2: Global clock counter

A 48 bit digital pattern used to tag event fragments before Front-end buffer formatting. The pattern is the actual count of the global clock. It will be used by PSA and global event builders to merge the event fragments in one single event.

3: Global event counter

A 16 bit digital pattern used to tag event fragments before Front-end buffer formatting. The pattern is the actual count of the L1 validations.

4: Trigger controls

The Trigger Control must guarantee that sub-systems are ready to receive every L1 Accept delivered. This is essential to prevent buffers overflows and/or trigger signals missed when the crystals are not ready to receive them. In either case, the consequence would be a loss of synchronization between event fragments.

Warning signals sent from the crystal through the GTS network, indicating that some of its buffers are almost full, may be received centrally. However this feedback signal can take few microseconds to reach the Trigger Control, which meanwhile could have delivered a number of L1A signals that originate a buffer overflow. This problem is particularly acute in the front-end derandomizers which have a small storage capacity.

According to the front-end electronic logical model, the front-end derandomizers after the L1 latency pipelines are the first devices to overflow when the L1A rate is too high. Space and power constraints in the front-ends imply small derandomizer depth and hence these queues are very sensitive to bursty L1A. In general, the derandomizers behave like a first-in-first-out queue: the input/output frequency is directly the L1A rate. The overflow probability is strongly dependent on the ratio between the service time and the buffer depth. The consequence would be of resetting the whole front-end electronics which would cause a severe loss of efficiency in the DAQ. All front-end derandomizers behave identically. Therefore, their occupancy depend only on the L1A rate and on the service time. A state machine receiving the L1A signals can emulate the de-randomizer behavior and determine its occupancy at each new L1A. If a new L1A is estimated to cause a de-randomizer overflow, this L1A is throttled. In general, it would be very difficult to guarantee that the state machine reproduces exactly the buffer status at every time. However in the present case the L1A accept signals are synchronous with the clock and the write and read latencies are measured in multiples of the clock period. It is this time quantization that makes the de-randomizer emulation really possible. A complementary solution to the same problem is to oblige the delivery of L1A signals to comply with a set of trigger rules. These rules take the general form 'no more than n L1A signals in a given time interval'. Suitable rules, inducing a negligible dead time, would minimize the buffer overflow probability.

5: Trigger request

The central core contact signal might be considered as the overlap of all the signals in the segments of a crystal; in fact a deposit of energy in any of the segment will induce a signal in the central core, thus acting as a sort of analog sum of single segment signals. Therefore, the central core can be processed for event localization in a crystal. Suitable algorithms for this task have been identified [] and tested. The outcome of the algorithm may issue a trigger request to the central trigger processor by asserting this signal which is transmitted via the high speed serial links of the GTS network upwards to the central trigger unit.

All the trigger requests collected from the crystals at each global clock cycle form a pattern that can be processed centrally for multiplicity or coincidence with ancillary detectors. The result of this processing stage constitutes the L1 validation.

6: Error reports

Abnormal conditions as buffer overflows, local faults, built-in self tests, etc. can be reported centrally for proper corrective actions.

4. Front-end Simulation

The Front End System presents intrinsic inefficiencies by design, in the sense that all level-1 triggers might not be processed. The two major causes of these inefficiencies are the Front End System dead time following a trigger, and the limited size of the Front End Channel derandomizing buffers.
For the complete readout chain of AGATA, a maximum acceptable inefficiency has to be defined: a value in the range of a few percents at 1 MHz trigger rate has to be achieved. Unfortunately, this trigger rate is not known as an absolute worst case, hence, to have a good safety margin for system design, we should regard it as the variance of a Gaussian process and take the 6σ value as our absolute worst case.

4.1 Front-end system dead time

A dead time is induced after level-1 trigger reception by the hardware architecture of the Local Level Processing electronics. This dead time is due mainly to the time needed to read a frame out of the front-end fifo's and to the time needed to sink the synchronization tags from the GTS interface mezzanine. As a consequence, triggers occurring during this period cannot be processed, and corresponding event data are lost.

An estimation of the inefficiency induced by the Local Level Processing hardware can be calculated with the following assumptions. Let *d* be the dead time, *r* the trigger rate. If we consider a Poisson law for the trigger distribution, the probability of one event (at least) occurring during the dead time is:

$$P_l = 1 - \left(e^{-rd} \, x \, \frac{\left(rd^0 \right)}{0!} \right) = 1 - e^{-rd} \approx rd$$

assuming that *rd* is small relative to 1.

Hence, at the level of each crystal, 1 microsecond of dead time after local trigger generation will account for 5% of local efficiency loss at 50 KHz trigger rate. Global system inefficiency due to hardware dead time is computed in the following way. Let's remember how a global trigger is usually generated in AGATA: local triggers generated at the crystal level get routed to a central trigger processor which is configured for asserting a trigger validation whenever a programmable condition is met. The simplest of these conditions is a multiplicity, e.g. pretending that more than one (usually M = [2..30]) trigger requests are asserted in the same time window.
It's easy to realize that, at multiplicity M>1, dead time of one crystal electronics induces a dead time for the whole system. For sake of simplicity, let's suppose M=2. An event firing crystals, say, no. 1 and 2 gets validated and those crystals enter a 1 microsecond dead time. Any subsequent event firing any two crystals will be validated only if crystals no.1 and 2 are not interested by the new event. The probability of the new event firing crystals no. 1 or 2 is computed by knowing the number of times that crystals no. 1 or 2 are present in all possible combinations without repetition of any two crystals out of 180 (this is the number of HPGe detectors foreseen for AGATA).
Let's take N = the total number of crystals, M = multiplicity: the total number T of combinations without repetition of any M crystals out of N is the well know formula:

$$T = \left(\frac{N!}{M! \times (N - M)!} \right)$$

The number of times K that one crystal is present in all possible combinations without repetition of any M crystals out of N is given by :

$$K = \left(\dfrac{T}{\dfrac{N}{M}} \right) = \dfrac{(N-1)!}{(M-1)! \times (N-M)!}$$

So, the probability of the new event firing a crystal that entered a dead time is given by:

$$P_d = M \times K / T = \dfrac{M^2}{N}$$

It's nice to note that, already at multiplicity M=13, with N=180, it's almost certain that a new event occurring less than one microsecond after the preceding will hit a crystal whose electronics will not catch it.
Hence, the global system inefficiency due to crystal dead time is obtained as:

$$P = P_l \times P_d \approx \dfrac{M^2 rd}{N}$$

for $M \leq \sqrt{N}$ or

$$P = P_l \times P_d \approx rd$$

for $M \, \phi \, \sqrt{N}$

4.2 The Derandomizer Size

The size allocated to the front-end derandomizing buffers is a crucial issue. On one hand, available space on silicon and power budget lead to minimize the size. On the other hand, data loss probability due to buffer overflow has to be minimized to avoid misalignment in the DAQ, thus inducing large inefficiencies caused by long recovery times. This constraint leads to maximize the derandomizer size.
The model used for this study is mainly defined by 3 parameters:

- the mean time between two triggers "*t*", i.e. the inverse of the trigger rate
- the fifo's service time "*s*", i.e. the minimum time between two consecutive read access to the fifo's.
- The fifo depth "*d*". This parameter is expressed in terms of events (a constant event size is assumed).

With this model, *P(n)* , the probability of having *n* events waiting in the fifo is computed. *P(n=d)*, is the probability for the fifo to be full and also the derandomizer inefficiency.

To analyze the contribution of these parameters to the global system inefficiency a simulation environment has been setup. We made a faithful description of the Local Level Processing hardware and the GTS interface mezzanine operations by means of the SystemC hardware description language [ ]. Being both synchronous with the GTS distributed global clock, a cycle accurate description of the local trigger generation, GTS handshake mechanism, fifo's storage and readout and trigger matching procedure has been easy to achieve and deploy.

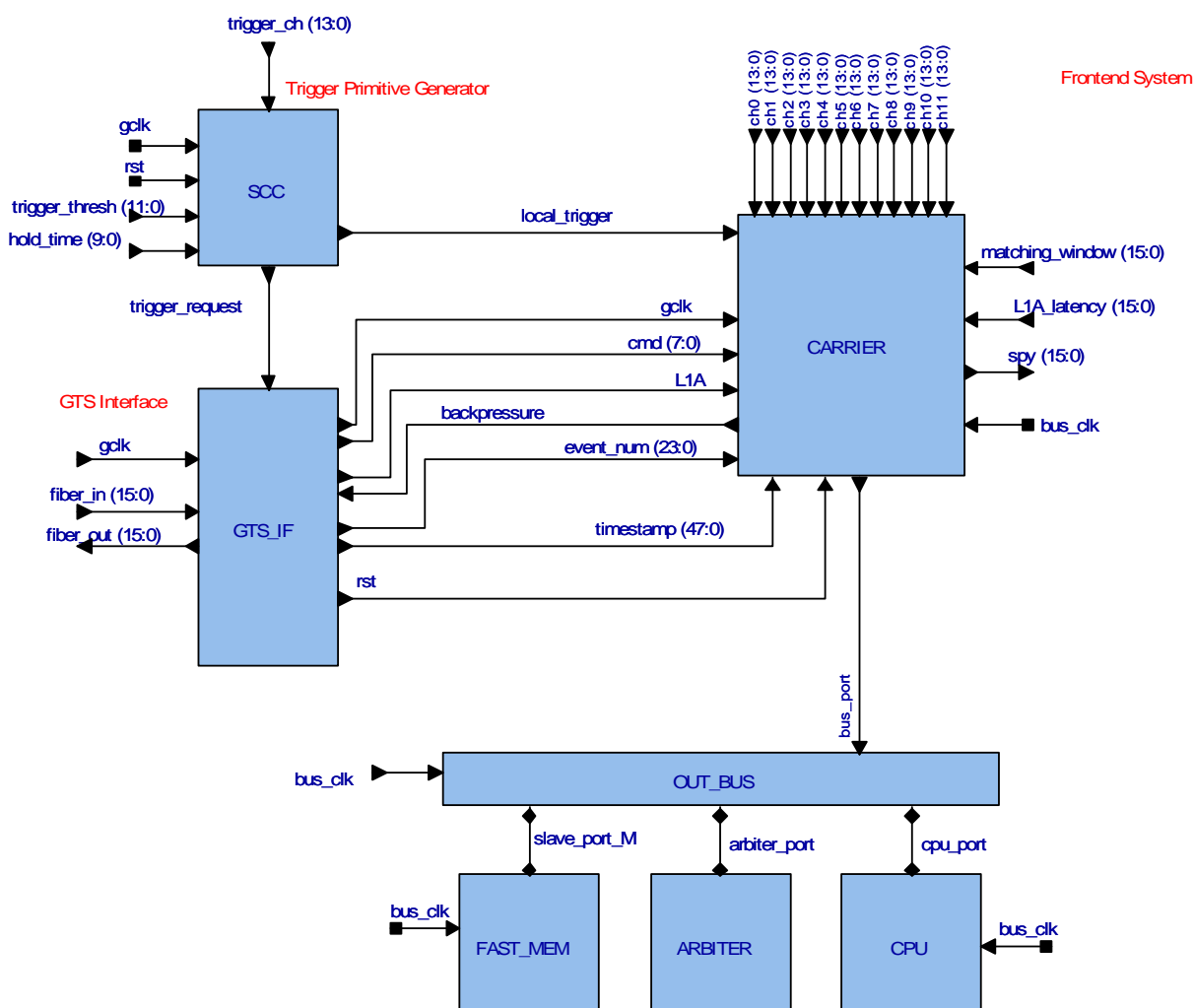## *4.3 The Simulation Environment*

Fig. 3 – Top level view of the simulation environment

Figure 3 shows the top level block diagram of the simulation environment. The front-end system is comprised of a GTS mezzanine interface, a primitive trigger generator which analyzes sampled data from central contact of each crystal for local trigger generation, a carrier box collecting data from twelve crystal segments and a readout bus to which a readout cpu and a global memory are connected. The model is sourced with real data taken from 24-fold and 36-fold segmented prototype crystals, illuminated with different radioactive sources.

It is worth noting that the simulation environment is a coded implementation of the front-end model depicted in fig. 2.

4.3.1 The carrier

The carrier box has the complex structure shown in fig. 4: two mezzanines (M1 and M2) take care of computing the energy and storing the rising edge samples of each event from six crystal segments each. Upon request of the readout engine box, on a per event basis these data are stored in a dual port ram after being tagged with a timestamp and event number taken from the 'evcount_fifo' and 'tstamp_fifo'. A Direct Memory Access (DMA) controller box synchronizes with the Readout Engine box by means of two fifo's (ro2dma and dma2ro fifo's) and sinks event data from the dual port ram into its bus port.
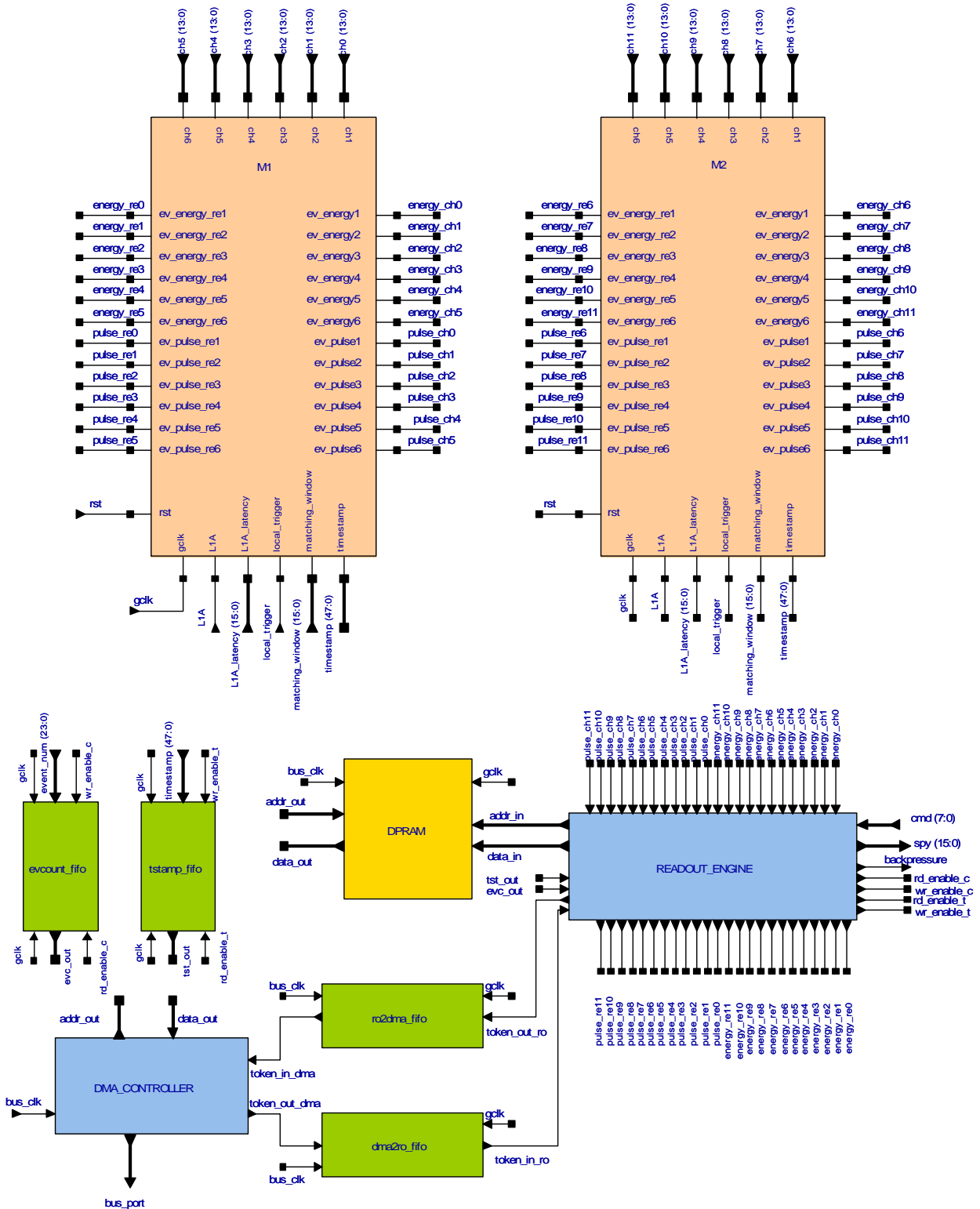
Fig. 4 – Block diagram of the "Carrier" box of fig. 3.

The mezzanines M1 and M2 are a six-fold instance of the block diagram of a single channel, shown in fig. 5.
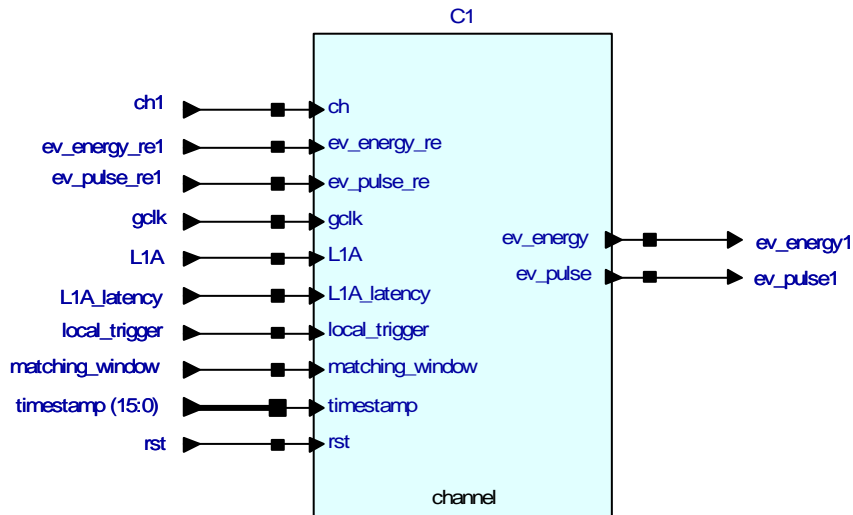
Fig. 5 – Block diagram of the "Mezzanine" box of fig. 4.

4.3.2 The Channel Structure

At the heart of the simulation code is the channel structure, as shown in fig. 6. Samples from the segment contact enter both a delay line and a Moving Window Deconvolution (MWD) box that computes the energy of the pulse and stores the value in a fifo (en_fifo) for later readout. Upon arrival of a local trigger, a pulse controller box (pulse_cntr) moves a predefined number of samples from the delay line into an event fifo (ev_fifo), thus isolating the rising edge of the pulse.
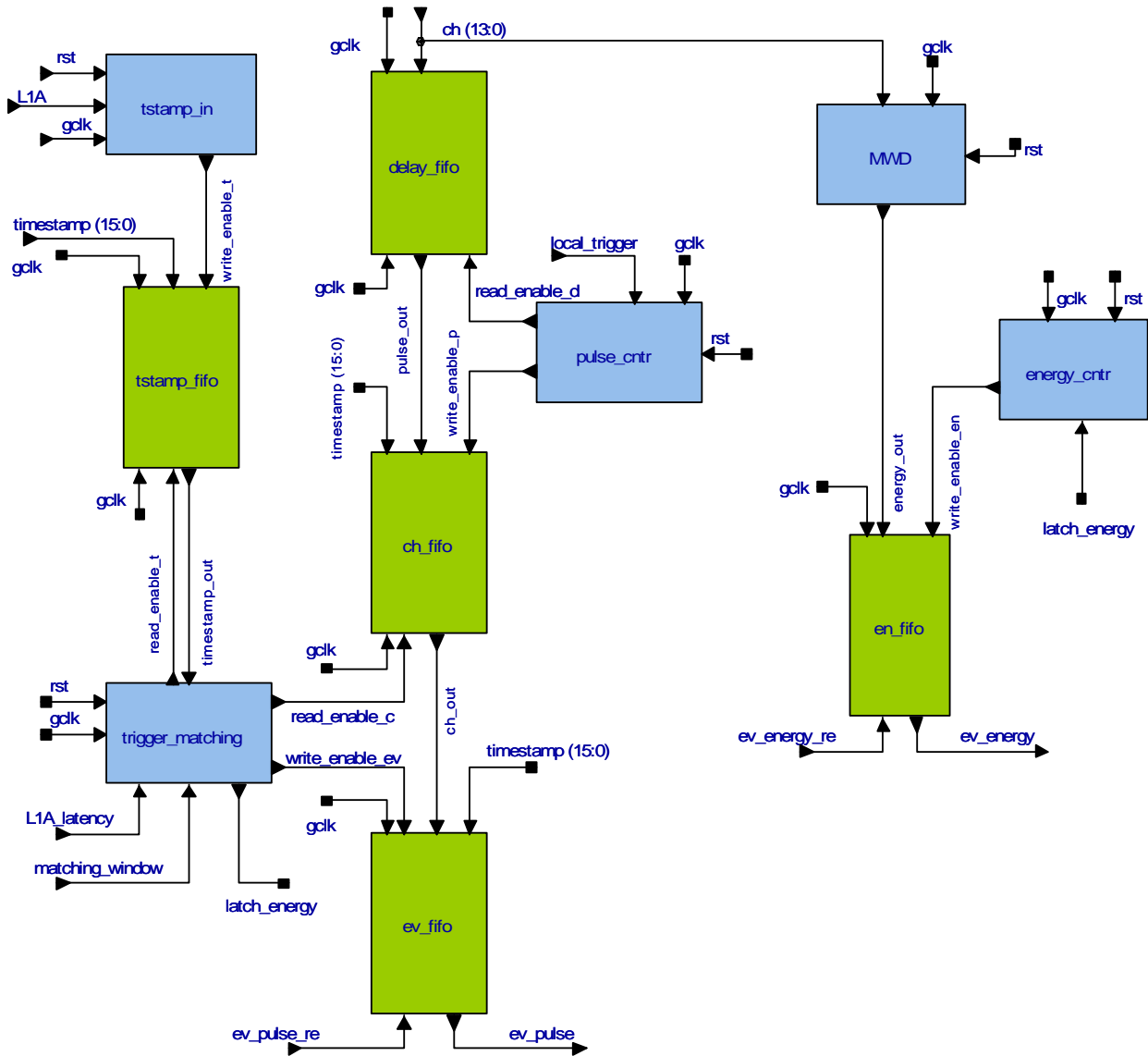
Fig. 6 – The channel structure.

4.3.3 Trigger Matching

The arrival of a trigger validation instead enables the storage of the current GTS time into a timestamp fifo (tstamp_fifo) to be used by the trigger matching engine.

The trigger matching engine is in charge of correlating the trigger validations stored in the tstamp_fifo with candidate local triggers that have triggered the storage of samples into the ev_fifo. In other words, trigger matching is a time match between a trigger validation time tag and the local trigger time tags themselves. The idea is shown in figure 7.
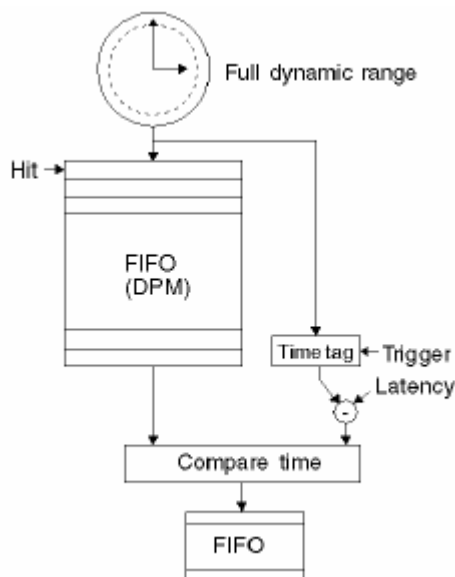
Fig. 7 – Trigger matching technique.

Local triggers and subsequent trigger validations differ in time for a number of reasons. The most obvious is that the latters are the result of logical operations performed on the formers, so they are intrinsically time ordered. There is a constant time difference between the two due to hardware infrascture that transports local trigger signals to a central trigger processor for validation generation. There is also a variable time difference between the two due to the mechanism of local trigger generation which depends somehow on the amount of charge induced by gamma interaction in each single crystal. As a consequence, the trigger matching procedure must be done inside a window, the match window, as show in the following figure.
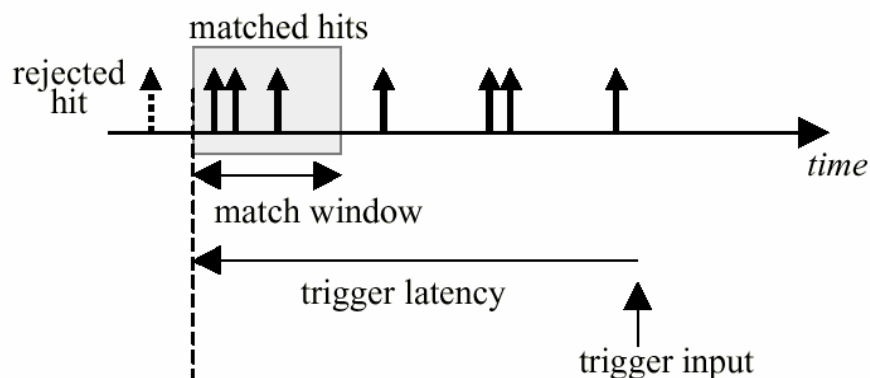

Fig. 8 – Window based Trigger matching.

At the time of a validation arrival ('trigger input' in fig. 8), from the current value of GTS clock a trigger latency value is subtracted. The result is an estimate of the time at which the candidate local triggers have happened. From that time on, for a period that equals the match window, all events whose time tags fall inside the window get validated and will be moved in the readout fifo. The events whose time tags fall before the time window can be discarded from the input fifo because they have no chance to be validated in future. Those events instead, with time tags that are younger than the match window will remain in place due to the possibility of being validated at a later time.

4.3.4 Simulation Results

5. Global Timestamp Protocol

To be filled …

6. Global Trigger Algorithms

To be filled …

7. Implementation

AGATA GTS will have a tree topology, originating from the root node that will therefore act at the same time as the source of all global information (clock, timestamps, commands, L1 validations) and the sink of all trigger requests, fast monitoring signals and service requests coming from the crystals.
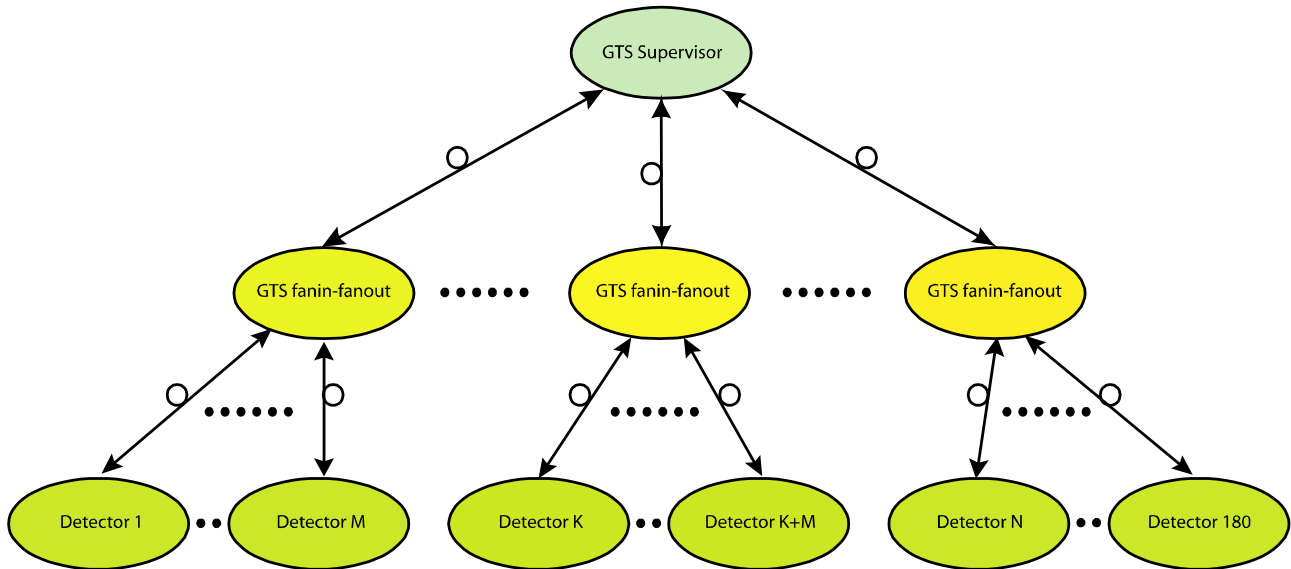

Fig. 9 – GTS topology

To solve the problems of building a bi-directional, high capacity and high speed tree network that drives hundreds of nodes displaced several tenths of meters apart, a certain number of technological issues must be addressed. Among them, the fan-out of a source synchronous transmission, noise immunity, low error rate and throughput. Modern serial transceivers, as used in commercial high speed telecom networks, can solve part of them. Simply stated, these devices basically transfer a digital pattern from one side of a transmission channel to the other (and viceversa) by serializing the pattern at a speed that equals the input clock frequency times the pattern width; the serialized pattern is then reconstructed identical at the receiving side of the transmission line by means of a serial to parallel conversion. AGATA GTS may greatly benefit of the technological solutions devoted to high speed serial transmissions, because, by exploiting the use of these components, the design can be kept simple yet adequate. The solution proposed is sketched in the following figure.
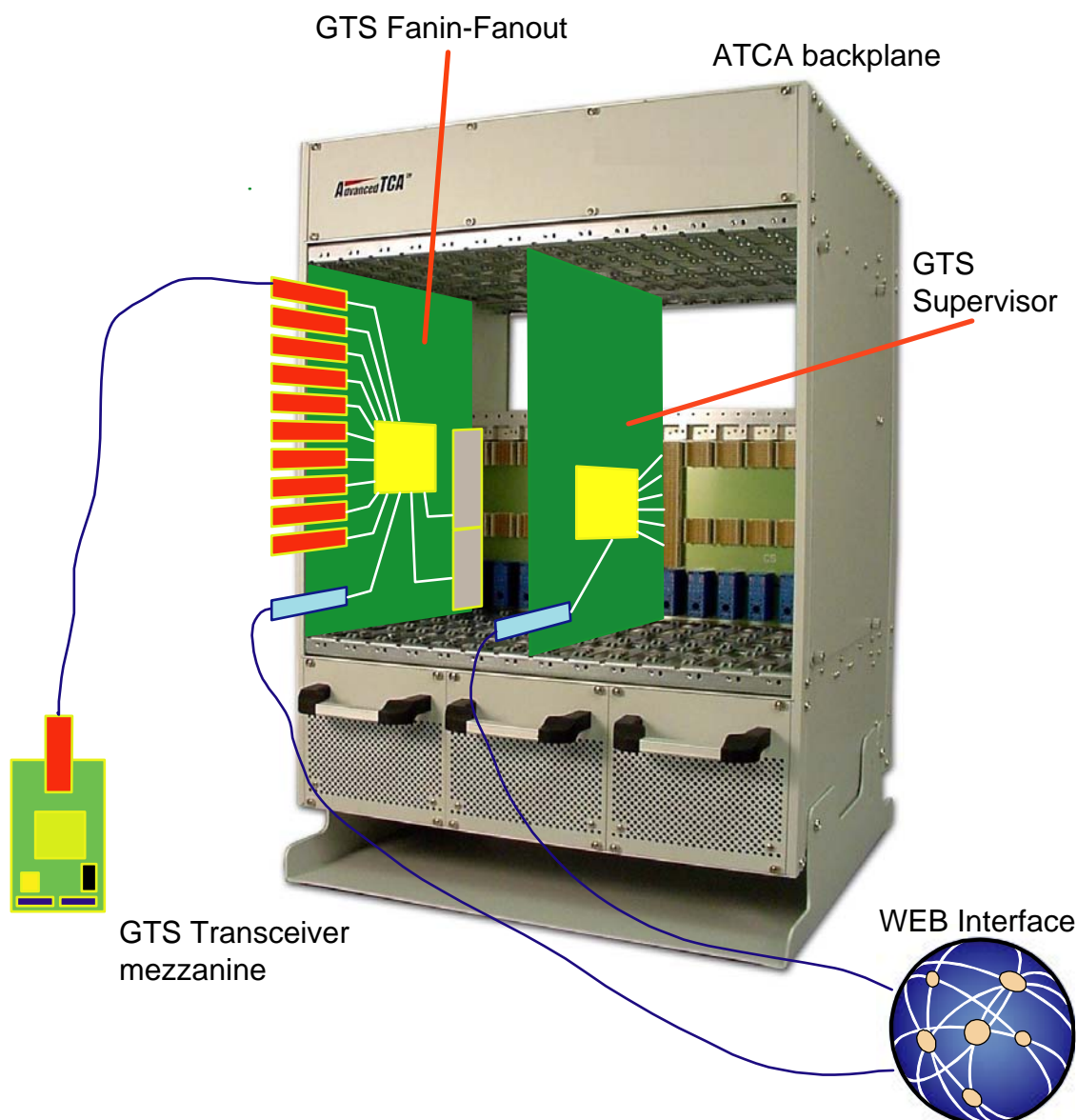
Fig. 10 – GTS technology and implementation

The hierarchy is composed of five different parts:

1. the root node
2. the backplane
3. the fanin-fanout nodes
4. the fibre connections
5. the mezzanine interface

7.1 The root node

This is the global time reference source and trigger processor. It might actually split in two boards, one for each direction of transmission. It sits in the central slot(s) of a (dual) star backplane and sources:

         a)   global clock

b) global timestamps (global clock counter, global event counter)
c) fast commands

and sinks:

a) trigger requests
b) fast monitoring feedback signals
c) error notifications

A block diagram is shown in the next figure:

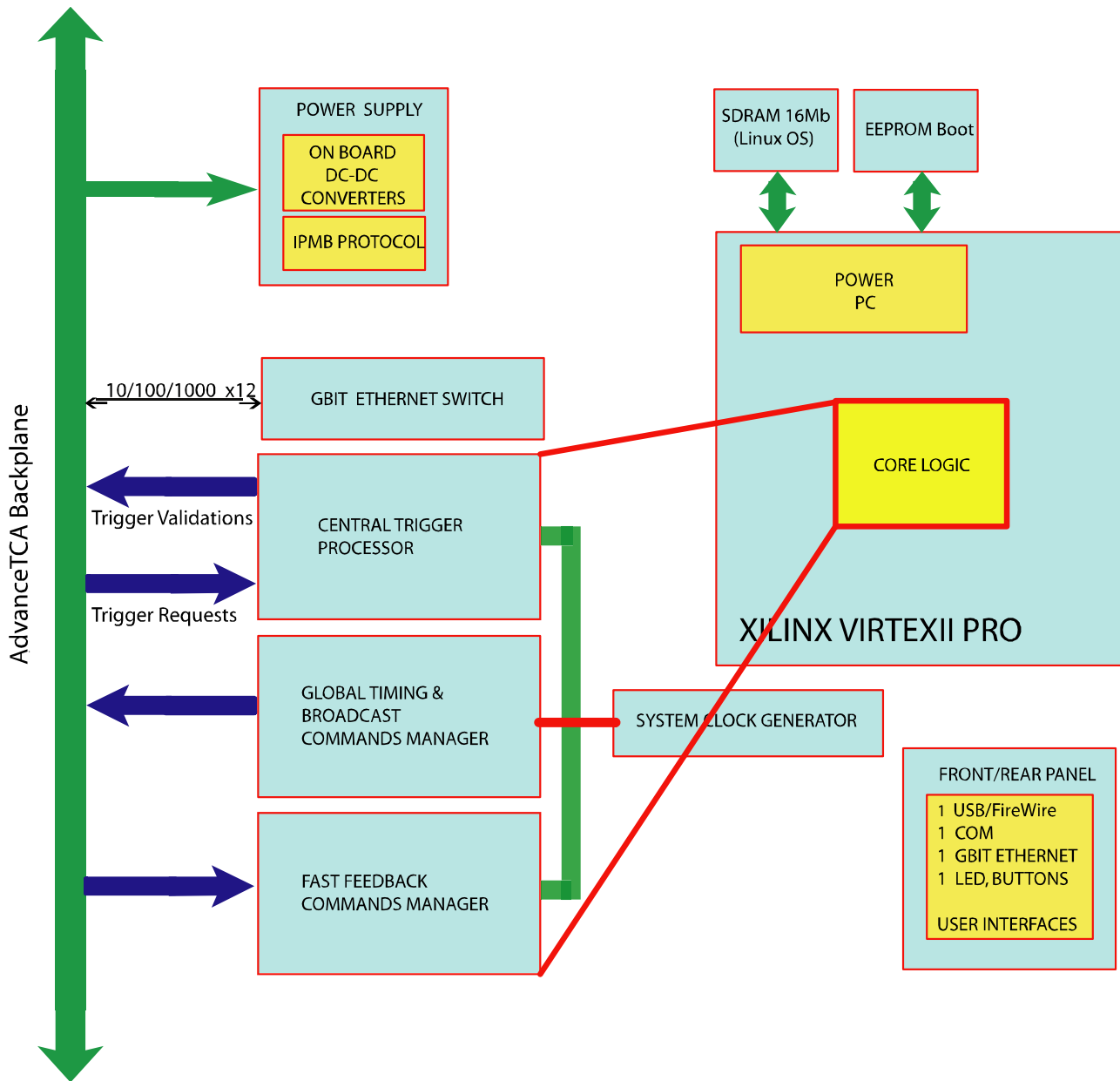**Fig. 11 – Block diagram of GTS root node**

7.2 The backplane

Advanced Telecommunication Architecture (ATCA) is a telecom specification (PICMG3.0) [2] aimed at standardizing the connectivity backplane for high speed, high throughput computing and switching devices. The specification is geared towards serial switched technology and is of interest for AGATA GTS. Backplanes following ATCA specification have dual star (or full mesh) topology

with the two central slots as the centres of the two stars and 6+6 slots on the periphery (14 peripheral slots is also an option). Each slot is connected to the centre by means of a certain number of matched impedance PCB traces (whose differential skew is less than 10ps) and routed to sustain a bit rate in excess of 3 gigabits/s per pair.

7.3 The Fanin-fanout nodes

These nodes act as splitting-combining nodes. When splitting, they replicate the information originated in the root node in each of the channels that they address; when combining, they merge the information originated in the crystals and send it to the root node. Each node is actually a board sitting in one of the 12 peripheral slots of the ATCA backplane and addresses 16 crystals. A block diagram of the board is show in the following figure:
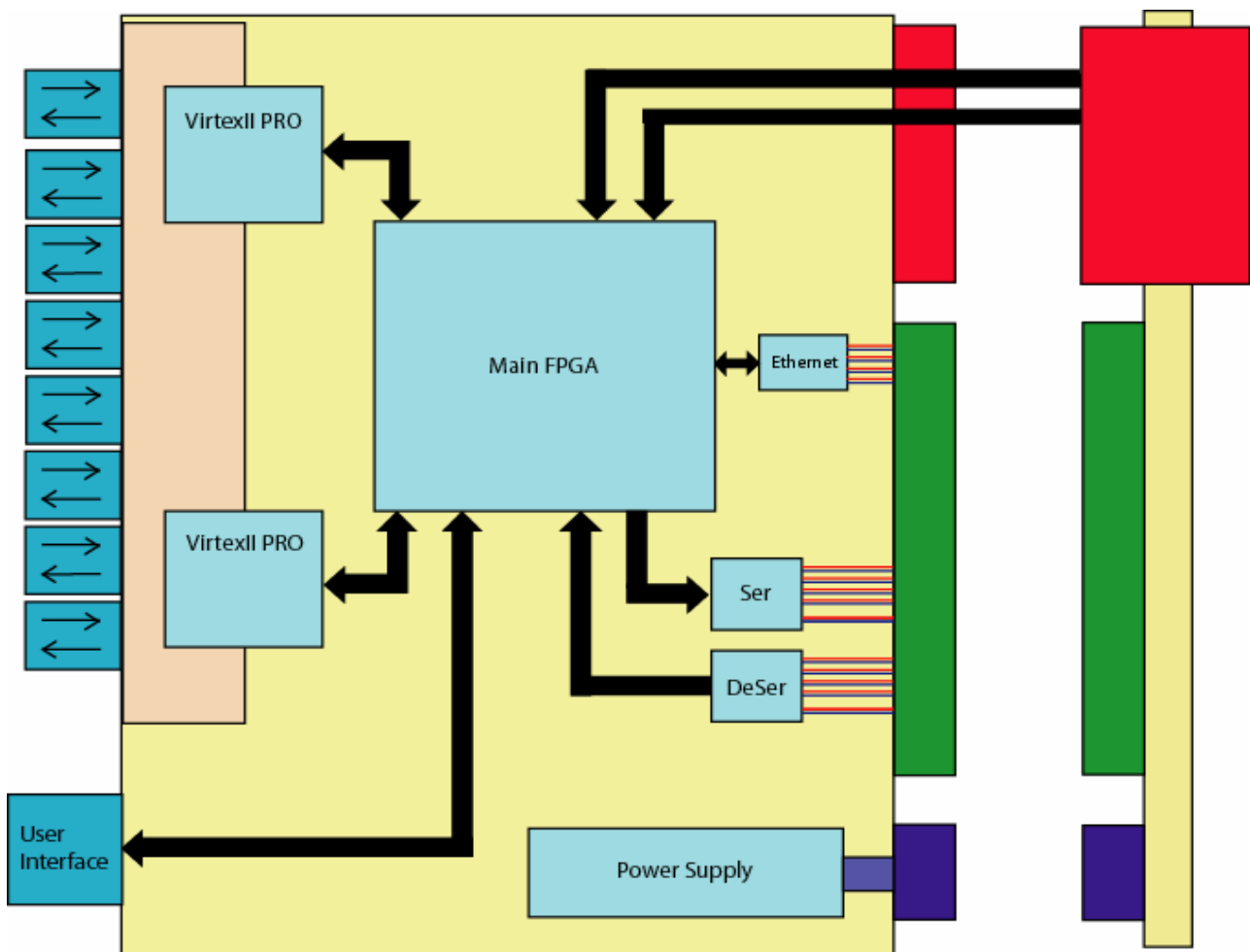

Fig. 12 – Block diagram of GTS fanin-fanout module

7.4 The fibre connections

Long distance transmission, noise immunity, galvanic isolation and low bit error rate (BER) will benefit from the use of serial optical transceivers and fibre connections between each channel of

the fanin-fanout nodes and the mezzanine interfaces that are hosted in the front-end electronics of each crystal.

7.5 The mezzanine interface

This is the block diagram of the GTS interface at the crystal level (or any ancillary detector). It decodes the information originated from the root node in one direction and encodes the information originated in the crystal in the opposite direction. The board is centered around a Xilinx Virtex2pro-XC2VP7-FF672 FPGA which hosts the custom logic for GTS operation. Figure 13 shows the main functional blocks of the card.
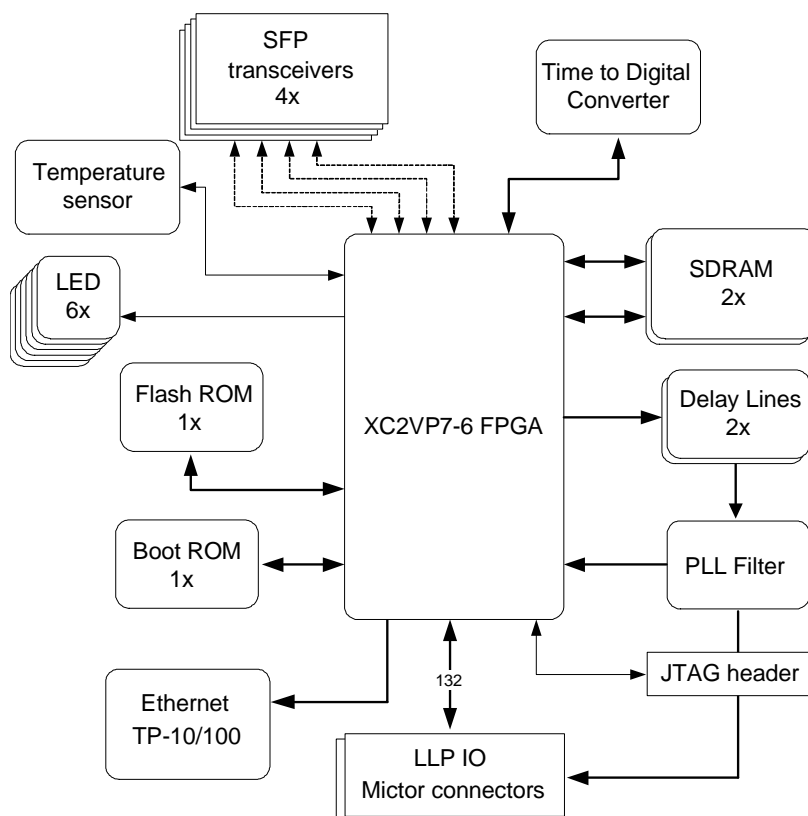

Fig. 13 – Block diagram of GTS mezzanine interface

The board features:
- A Xilinx VirtexII-pro XC2VP7-FF672-6 field programmable gate array with 8 serializers/deserializers (rocketIO's) capable of 3.125Gb/s operation and an embedded PowerPC cpu.
- Up to four pluggable optical transceivers interfacing the rocketIO channels of the FPGA to the fiber tree of the GTS system
- 128Mbytes of RAM and 16Mbytes of flash ROM for the embedded PowerPC microprocessor and operating system support
- A flash ROM for FPGA configuration
- Two programmable digital delay lines for phase adjustment of the GTS global clock
- A high performance PLL with very low loop bandwith for jitter removal of the GTS clock
- A twisted-pair Ethernet channel for slow control operation
- Two mictor-type connectors for Local Level Processing and Ancillary Detector interfacing
- A time-to-digital converter for time-of-flight pulse measurements during GTS global clock phase equalization procedure.

A picture of the board is show in figure 14. Actually two implementations are shown, with one and four optical transceivers.
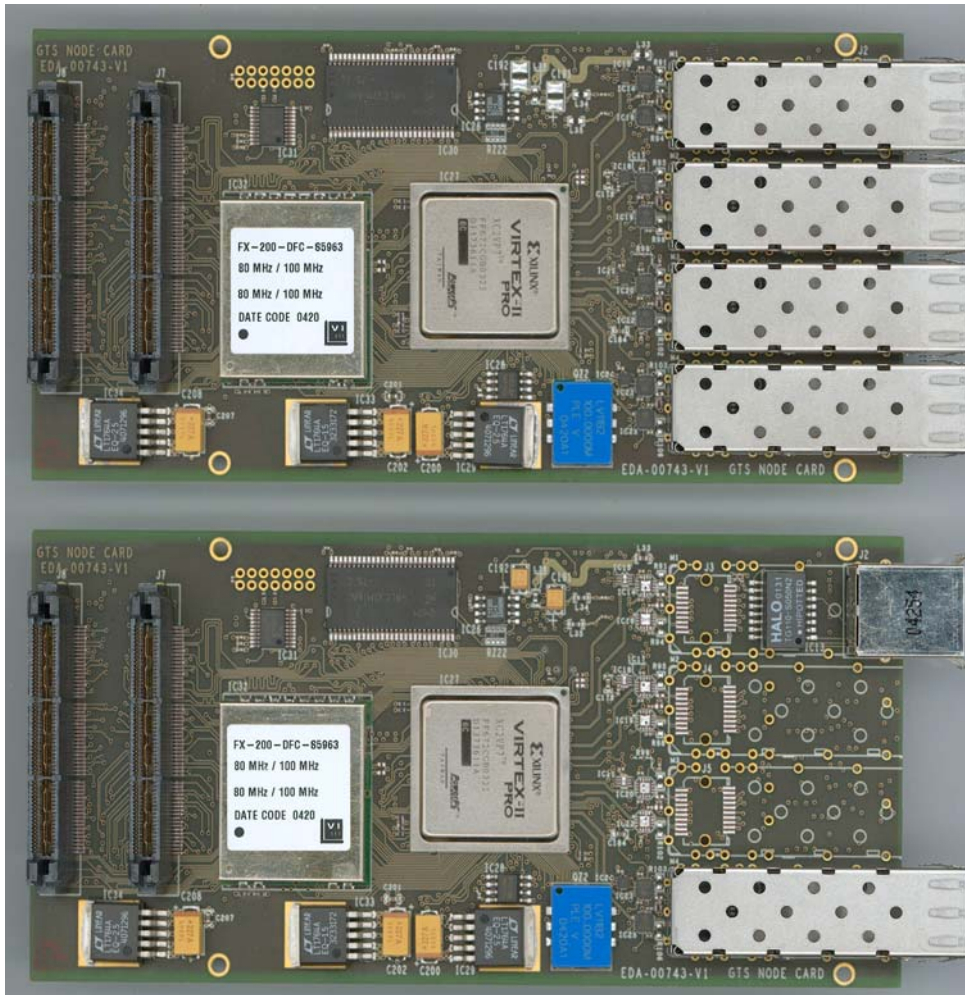

Fig. 14 – Picture of the GTS mezzanine interface

The form factor adheres to the Common Mezzanine Card (CMC) standard.

7.5.1 GTS Mezzanine operation

The basic tasks that the mezzanine fulfils are listed below:

- It must reconstruct the global GTS clock, 48bit timestamp, level 1 trigger validations and global commands coming from the GTS root node through the fiber and a rocketIO channel into the FPGA.
- It must source the reconstructed global clock, 48bit timestamp and global commands to the LLP or ancillary node through the mictor connectors.
- It must feed the root node with a 1 microsecond paced heartbeat messages ("idle" commands) to be used in the central trigger processor.
- It must provide an Ethernet based slow control connection.

- It must provide a mechanism for setting the reconstructed global clock to an arbitrary phase in a 10 ns interval.
- It must source the reconstructed global clock with a sufficiently low phase noise so as to be readily used for multi-gigabit transceivers operation by the LLP or ancillary node.
- It must provide a mechanism for accepting, tagging and storing local trigger requests coming from the LLP or ancillary node according to a well known, fixed latency handshake protocol.
- It must route tagged trigger requests, status informations and trigger throttle control commands originating from the LLP or ancillary node to the GTS root node via a multi-gigabit transceiver channel and optical transceiver.
- It must implement the trigger matching algorithm, that is correlating trigger validation tags coming from the root node with stored trigger requests and in case of matching or rejection, it must inform the LLP or ancillary node according to a well known handshake protocol.
- It must provide a mechanism for measuring the round trip times of pulses originating in the GTS root node and reaching the Digitizers clock distribution boards for the purpose of phase equalization at the Digitizers nodes.
- It must provide all the necessary resources for the internal CPU to run a high level operating system with a full TCP/IP network connection.

Hence, the role of the GTS mezzanine as global synchronization and control agent becomes evident.

7.5.2 A VHDL model of the GTS Mezzanine

As stated before, the role of the Xilinx FPGA in the mezzanine is to host both the glue logic for interfacing the board hardware resources and the custom logic for the value added operations listed above. The following figure shows a block diagram of this logic. Actually the diagram hides a hierarchy of blocks, each taking care of a basic subtask, and deeply relating each other by means of signals. The hierarchy is a graphical representation of an underlying vhdl model that is synthezisable for the VirtexII-pro technology. The aim of the model is also to provide a tool for checking the correctness of mezzanine operation by means of simulation. In fact, the model is part of a broader simulation strategy whose target is to develop and deploy a software framework for the validation of the whole GTS tree.

Appendix D reports an expanded view of the hierarchical block diagram, together with a detailed explanation of the interplay among the different subblocks.

Fig.15 shows the top level interface of the mezzanine vhdl model ("GTS_TOP") interfaced, by means of two complementary pairs ("fiber_in" and "fiber_out"), to the root node model ("ROOT") which acts as a testbench exerciser. The task of the root node model is to provide the necessary setup informations (reset commands) and 8-bit coded timestamp pattern to the mezzanine and subsequently to validate all trigger requests coming from it, thus emulating a multiplicity 1 trigger condition. The red-coloured signals put in evidence the LLP-side interface, whose name, width, direction and electrical standard has been completed specified with a LLP document [ ].

The full model is browsable at the web site : http://agata.pd.infn.it in the GTS reserved section.
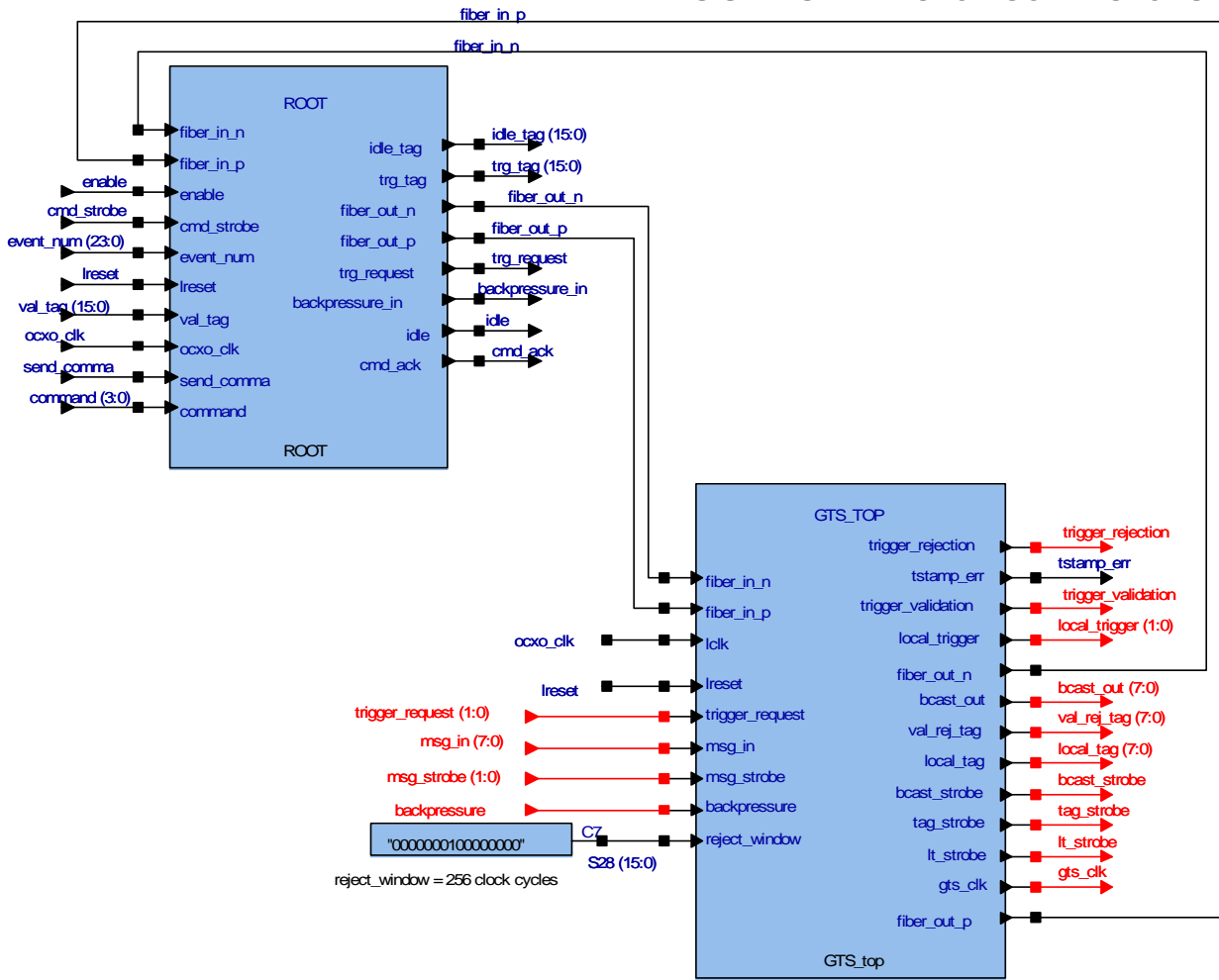
RED SIGNALS <=> MICTOR CONNECTORS

Fig. 15 – Top level block diagram of the FPGA operation, together with its testbench

8. Phase equalization of GTS clocks

The GTS clock, together with its associated timestamp, constitute the time reference for the whole AGATA system. In each crystal, a replica of the main GTS clock, sourced through a dedicated GTS interface mezzanine, feeds the analog-to-digital converters for digitization of segments signals from the charge preamplifiers. The same clock feeds also the communication devices that transfer the relevant information from the crystals up to the local level processing hardware. Moreover, as the whole data acquisition process relies totally on timestamps for the event building process, it is easy to understand that the GTS system must source replicas of its main clock and timestamp that are time aligned at each crystal level. Failing to do so would imply a number of consequences, the most obvious of which is the incorrect time tagging of event fragments and hence incorrect event building, but also potentially incorrect trigger processing. Hence, the problem of clock phase equalization is a crucial issue in AGATA.

The reasons behind the misalignment of clock phases among different crystals are multiple. To cite a few there are: different PCB trace lengths, different fiber lengths, different propagation delays of active devices, different routes inside programmable logic devices, different process-voltage-temperature corners of active devices, different equalization fifo's depths of serializer-deserializer (serdes) devices used throughout the GTS system and many others. While a detailed model of clock phase mismatches is beyond the scope of this document, from what stated above it is obvious that we need to devise a method for diagnose and take care of phase misalignement. An analysis of the problem shows that the contributions to the phase mismatch can be divided in four categories:

	Type of Contribution	Reason
1	Fixed	PCB trace lengths, propagation delays, …
2	Static	Equalization fifo's of serdes devices
3	Slowly changing	Temperature, optical phase dispersion, …
4	Rapidly changing	Power supply ripple, …

Table 2: Contributions to phase mismatch.

While the contributions that fall in the first category can be measured once, those falling in the last three categories must be diagnosed at run-time and fixed with a fast and automated procedure, so that the contribution to the global system inefficiency is negligible.

In AGATA, two different methods of phase equalization are currently under investigation:

8.1 Continuous resets method

The situation is the following: in the root node of the GTS tree the global clock and associated timestamp are generated and broadcasted to end nodes. In the end nodes the global clock is reconstructed and phase locked with the clock of the root node but with an unpredictable phase.
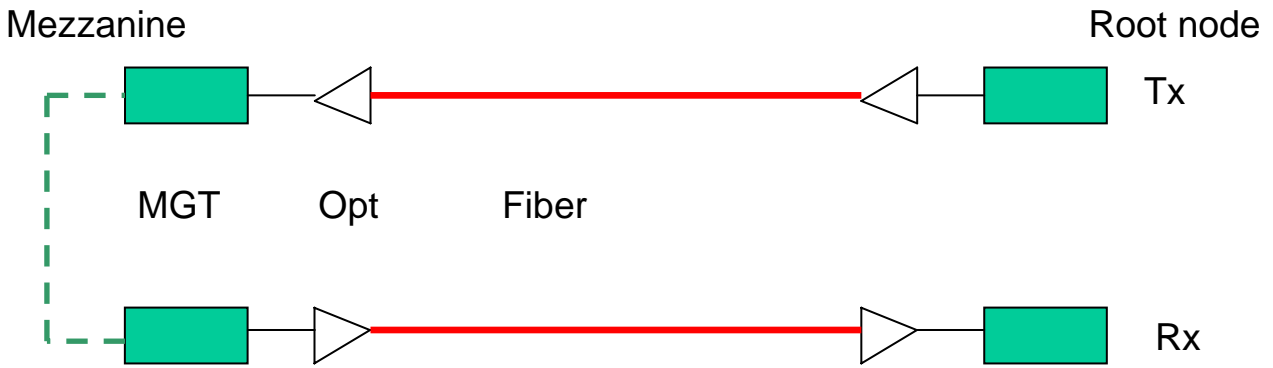
Mezzanine   Root node

Fig. 15 – Root node – End node pair with loopback

The main reason of phase uncertainty is contributions of type 2, e.g. due to the nature of clock data recovery circuit (CDR) of serdes devices. By putting a loopback in one end node, from the root node we measure the round trip time of a pulse and try to devise a method from which we can safely assume that the downlink and uplink channels are symmetric and hence the measurement (divided by two) equals the time of flight of a pulse from the root node to that end node. By repeating the measurement for each end node we will know how to setup programmable delay lines at the end nodes in order to reach phase equalization for all crystals.

From laboratory test, we can basically assume that the phase of recovered clock at one end node is almost uniformly distributed in one clock period. If we take the phase of recovered clock as a random variable, its probability density function looks like in figure 16
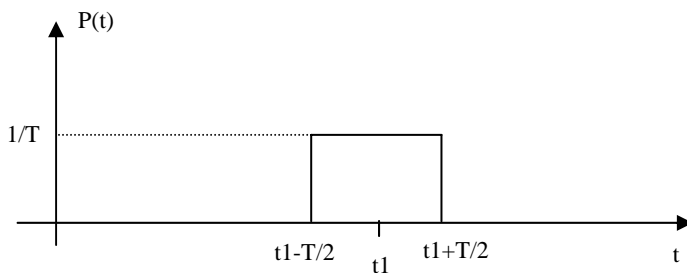

Fig. 16 – Ideal probability density function of recovered clock phase at the end node

Now if we consider the transceiver pair with loopback, as in figure 15 and we consider the phase of the recovered clock at the root node as a random variable, we might assume that it is the sum of two identical variables (as long as contributions of type 1 to phase mismatch have been minimized by design).
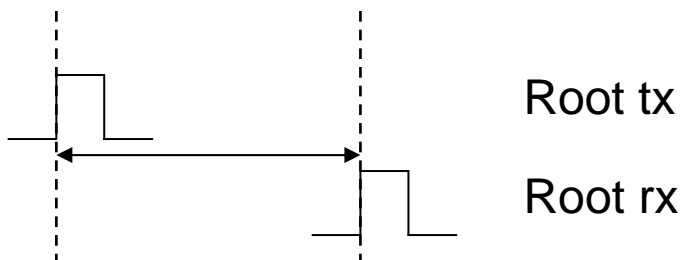

Fig. 17 – Global clock and recovered clock at the root node

The channels are in fact mutually independent because the transmitter clock is independent from the recovered clock and viceversa, so the probability density function should look like:
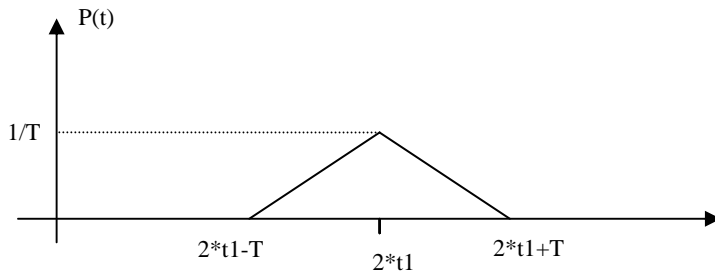
Fig. 18 – Ideal probability density function of recovered clock phase at the root node

In order to obtain the single channel latency dividing the total round trip time (RTT) by two, we have to look for a situation of maximum symmetry between the channels. The idea is to perform several reset-and-measure cycles until the RTT measure is close to the expected maximum or minimum. When this happens one can safely assume that the channels times of flight are respectively both close to their maximum or minimum and we can divide by two the RTT value to obtain the individual channel latency.

8.1.1 Test results

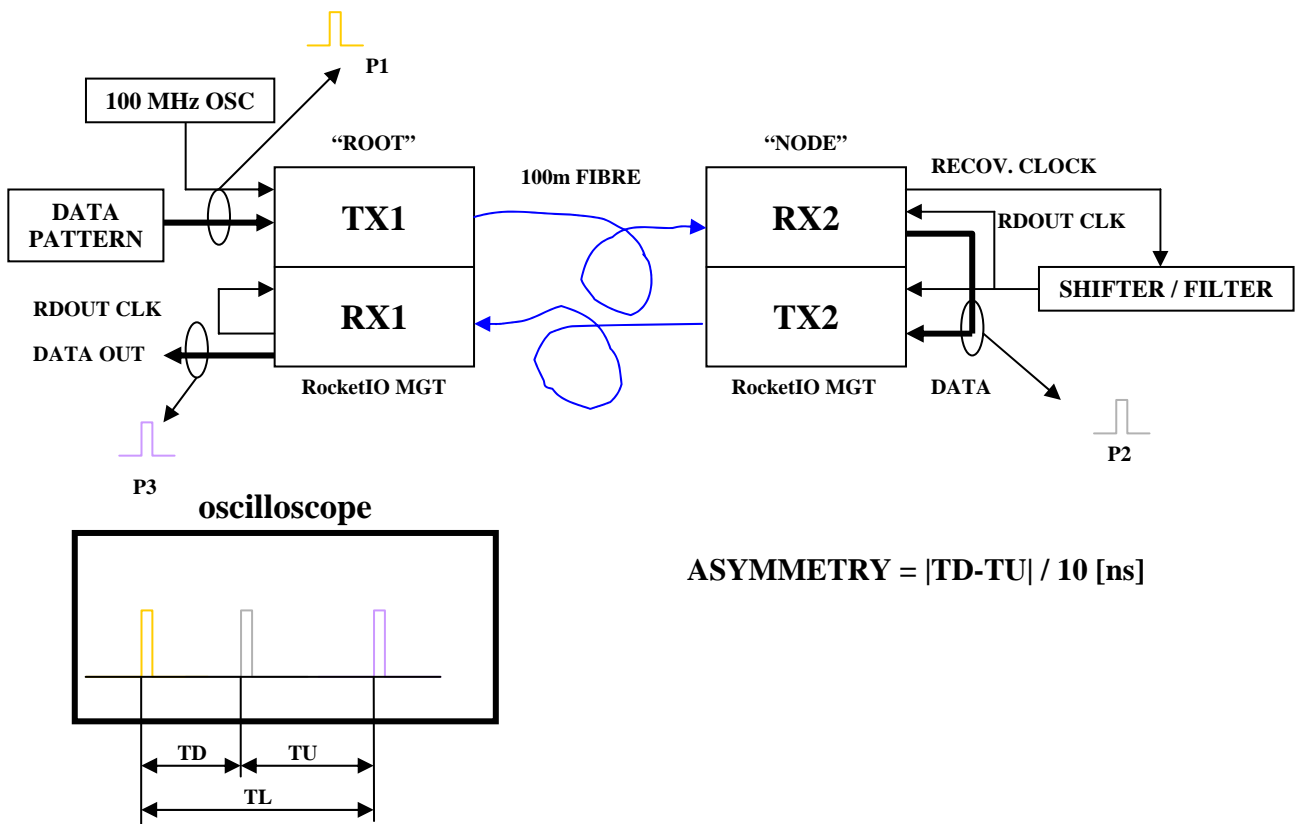The test setup is illustrated in the following figure.

Fig. 19 – The test stand for the continous resets method.

A data pattern P1 is injected at the root node in the downlink. At the end node the same pattern (P2) is recovered from the serial stream and its time of flight is measured by means of an oscilloscope (triggered by the time of occurrence of P1). The pattern is loopbacked towards the root node, recovered (P3) and its time of flight measured in the same way. In figure 20 the latency distributions of the downlink (TD) and uplink (TU) in a clock cycle are reported. In figure 21 the latency distribution of the loop (TL) in a clock cycle is reported.
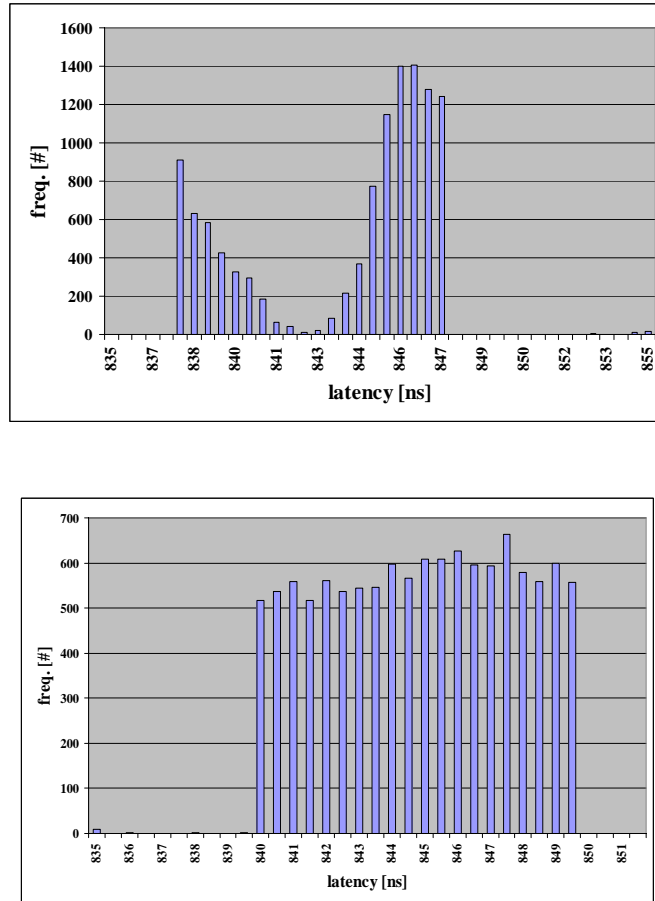

Fig. 20 – Latency distribution of downlink (TD – upper) and uplink (TU – lower).
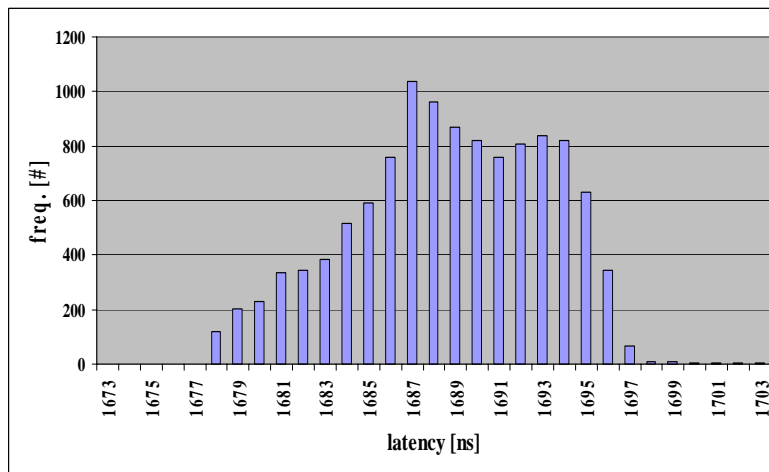

Fig. 21 – Latency distribution of the loop (TL).

By defining the degree of asymmetry as:

$$Asymmetry = \frac{\left|TD - TU\right|}{10(ns)}$$

one can plot the degree of asymmetry of the loop versus the global latency as in figure 22:
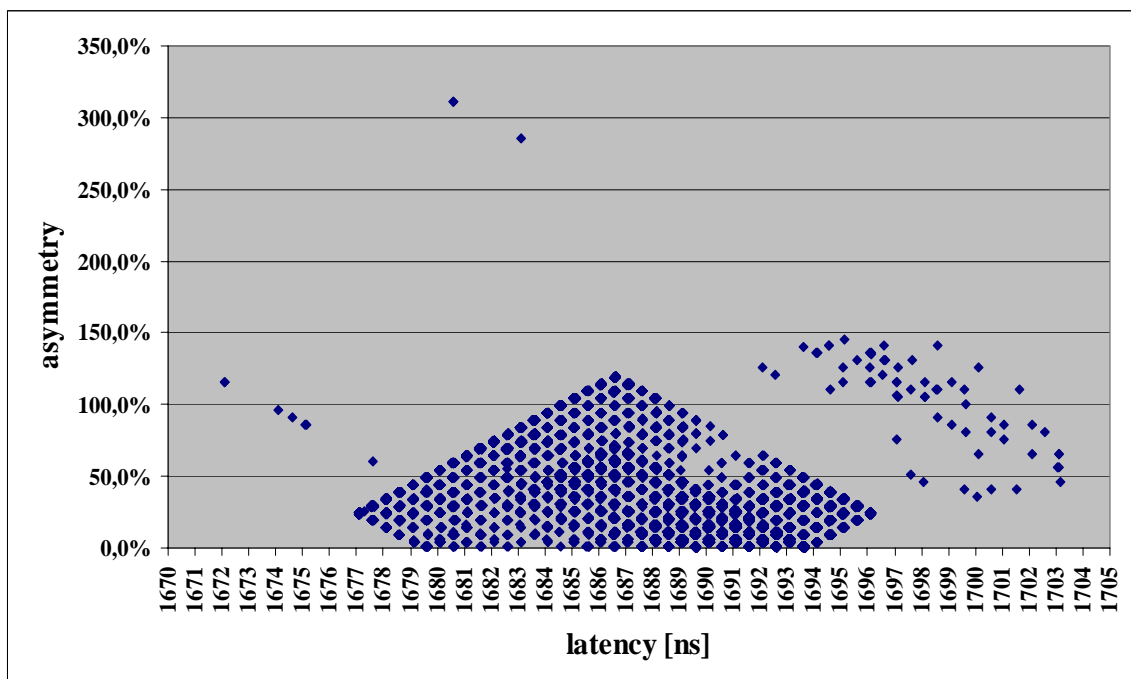

Fig. 22 – Loop asymmetry versus global latency.

The measurements shown in figure 22 have been obtained by performing more than 11.000 reset-and-measure cycles on the test stand of figure 19 with an automated procedure. This picture deserves some useful comments:

  ✓ Two minima of asymmetry are evident at the extremes of the global latency interval [1677..1696] ns. This means that, at those minima (which correspond respectively to the global minimum and maximum of the loop latency) the downlink and uplink latency are almost equal and the two channels (uplink and downlink) are symmetric. Hence, we can estimate the end node clock phase with a measurement procedure performed at the root node.
  ✓ The resolution of the measurements is 500 ps, which corresponds to the clock cycle of the serial stream.
  ✓ There is a certain number of points outside the [1677..1696] ns interval. These points are due to computation errors in the automation procedure (less than one hundred out of more than eleven thousands).
  ✓ There is a pedestal in the plot greater than zero, e.g. the minima of asymmetry at the extremes of the interval are not zero. In other words there is a fixed contribution to asymmetry, e.g. a systematic error due to contribution of type 1 (PCB traces and cable mismatches, propagation delays impairments, impedance mismatches, etc.). This type of contribution can easily be measured from a plot like the one in fig. 22. Once subtracted, the plot in fig. 22 becomes a perfect triangle and the minima at the extremes of the interval become global minima.

Due to the nature of our test stand, the measurements shown in fig. 22 took more than two hours to be performed. While an investigation is currently under way to speed up the measurement

process, it is evident that it is in the nature of the method itself to perform repetitive resets of serdes devices until a preferred latency is measured. That is, the method can be time consuming because we cannot foresee the time it takes to converge. For this reason it is not obvious that it might be used online, for periodic calibrations of the whole GTS distribution tree at beam time.

8.2 Direct Measurement

8.2.2 Test Results

9. References

[1] – I. Lazarus editor - AGATA Pre-Processing Hardware – Draft. Document – 17/12/2003
[2] – PICMG 3.0 – Advanced Telecommunication Computing Architecture – http://www.picmg.org

10. Revision History

12 Jan 2004 -- Draft version 1.0.
10 Aug 2004 -- Draft version 1.1.
24 Oct 2004 -- Draft version 1.2.
11 Nov 2005 -- Draft version 1.3.

Appendix A. GTS Mezzanine Interface Pin-out

The GTS mezzanine interface pin-out is specified as:

Name	Direction	No. of bits	Description
Clock100	Out	1	Global clock
Clock100des	Out	1	Global clock phase adjustable
Timestamp	Out	48	Global clock counter
Ev_counter	Out	16	Event counter
Scl	In	1	I2C interface
Sda	In/Out	1	I2C interface
Trigger_valid	Out	1	Level 1 trigger validation
Trigger_req	In	1	Trigger Request
EvCnt_str	Out	1	Event Counter strobe
BrdCst_in	Out	8	Broadcast Command bus from root node
BrdCst_in_str	Out	1	Broadcast bus strobe
BrdCst_out	In	8	Broadcast Command bus towards root node
BrdCst_out_str	In	1	Broadcast bus strobe
Error	Out	1	Transmission error
Master_reset	In	1	Mezzanine reset
Ethernet[0-3]	In/Out	4	Ethernet connection

Appendix B. Costs

B.1 Single Crystal prototype costs.

For single crystal prototype operation only 1 GTS Mezzanine card (working in emulation mode) is needed. GTS Mezzanine costs are detailed in the Pre-Processing Hardware Specs. [ ]

B.2 Demonstrator costs.

For demonstrator operation (15 crystals) the following items are needed:

- 15 Mezzanine interfaces (costs included in the Pre-processing Hardware Specs [ ])
- 1 Fanin-fanout board (16 channels)
- 1 Root Node (possibly splitted in two – see before)
- 1 ATCA crate

Root node(s)	Qty	Unit price	Cost
Pcb Layout	2	€5000	€10000
Pcb	2	€1000	€2000
VirtexII pro	2	€2000	€4000
Memory	2	€10	€20
Flash mem	2	€100	€200
LVDS serdes	24	€20	€480
Power supply	2	€100	€200
Connectors		€300	€600
PLL	2	€120	€240
Misc	2	€300	€600
Total			**€18400**

Fanin-Fanout	Qty	Unit price	Cost
Pcb Layout	1	€5000	€5000
Pcb	1	€1000	€1000
VirtexII pro	1	€2000	€2000
VirtexII pro V70	2	€500	€1000
Memory	1	€10	€10
Flash mem	1	€100	€100
Optical Transceivers	16	€250	€4000
Power supply	1	€100	€200
Connectors		€300	€300
PLL	2	€120	€240
Misc	1	€300	€300
Total			**€14150**

Assuming that, at the prototyping stage, two runs are needed to obtain a working system, then the costs for hardware for the demonstrator are (€18400 + €14150) X 2 = **€65100** for the boards and **€20000** for one ATCA crate. In total **€85100.**

Instrumentation & Software	Qty	Price
TDR test Jig	1	€4000
Differential Probe	1	€4000
DSO 4GHZ	1	€45000
2nd ATCA crate	1	€20000
Optical Power Meter	1	€2000
ATCA extender	1	€2000
Xilinx Dev Kit	1	€5000
Xilinx ISE	1	€500/year
Synplify Pro	1	€1000/year
VisualHDL	1	€2000/year
Synopsys	1	€2000/year
Total		**€87500**

NRE costs for Xilinx IP Cores are detailed in the Pre-Processing Hardware Specs.

Appendix C. Timescale and manpower

Timescale and manpower for the GTS Mezzanine are detailed in the Pre-Processing Hardware Specs [1]. It is reported here for sake of completeness:

C.1 GTS interface mezzanine

Task	Time (months)	Manpower (1 man)
PCB layout & Assembly	6	1
Testing		
FPGA development	9	1
Software		
Total	**15**	**1**

For the root node and fanin-fanout boards the following figures hold:

C.2 Root Node

Task	Time (months)	Manpower (1 man)
PCB layout & Assembly	4	1
Testing	1	1
FPGA development	12	1
Software	7	1
Total	**24**	**1**

C.3 Fanin-fanout Node

Task	Time (months)	Manpower (1 man)
PCB layout & Assembly	4	1
Testing	1	1
FPGA development	12	1
Software	7	1
Total	**24**	**1**

Appendix D. The GTS Mezzanine VHDL model

The GTS mezzanine is the leaf node of the Global Trigger and Synchronization tree and plays the role of an intelligent interface between the front-end electronics and the trigger complex.

The following is a description of the vhdl circuit structure that resides in the field programmable gate array in the mezzanine.
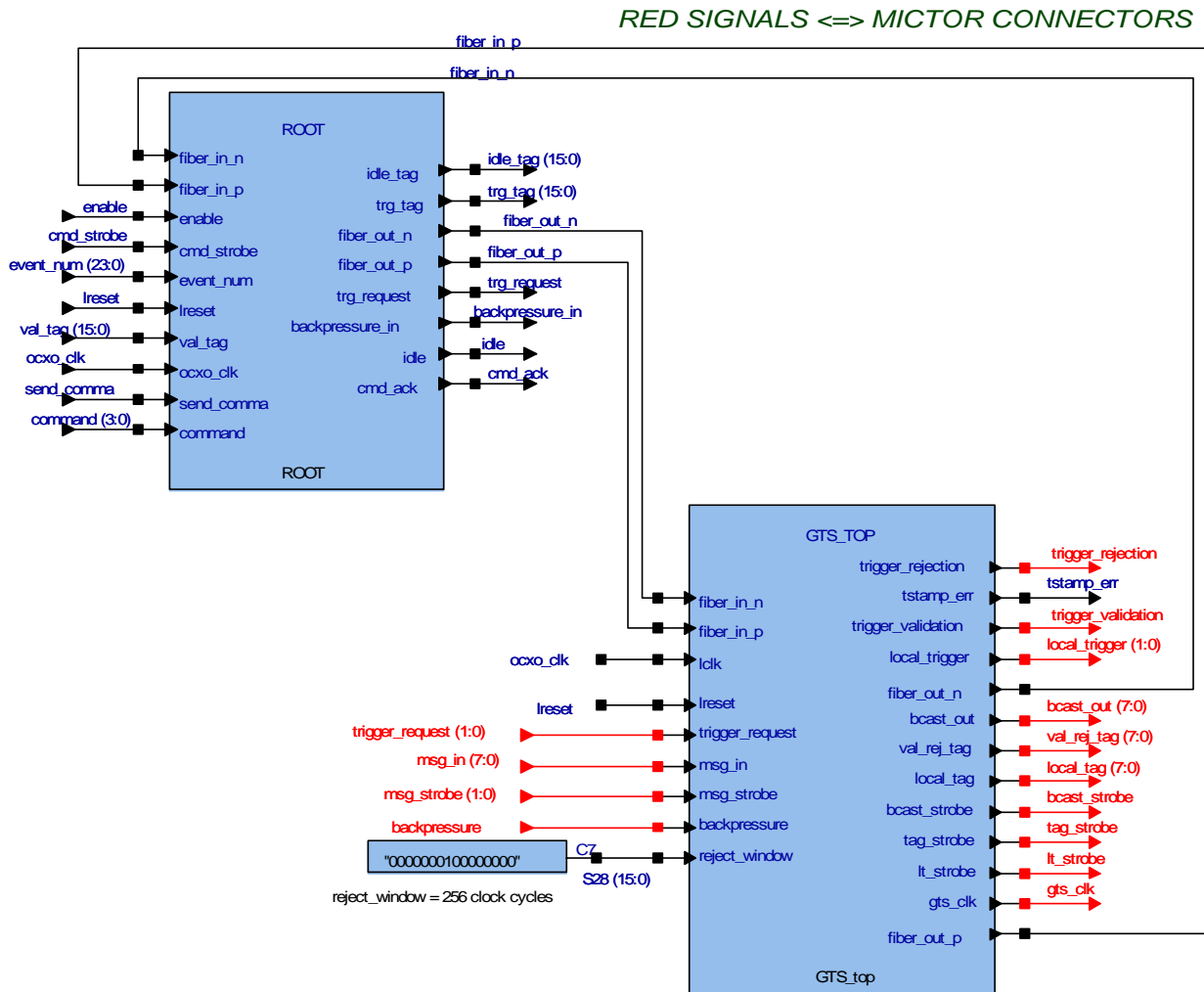

Fig. D1 – Top level block diagram of the mezzanine model.

At the top level is the interface between the root node and the mezzanine on one side and between the mezzanine and the front-end electronics on the other side. For the sake of clarity the LLP/ancillary interface signals are coloured in red. The model depicted in Fig. D1 shows two complementary pairs that connect the root node and the mezzanine in both directions. The pairs are the inputs and outputs of a couple of multi-gigabit transceiver pair connected back-to-back and simulated in the model by means of an encrypted SmartModel core distributed by Xilinx.

The model is missing the description of (and hence the interaction with) the slow control part, e.g. the microprocessor, which is an integral part of the mezzanine and takes care of crucial tasks as the global clock phase equalization and online monitoring. All the relevant setup informations are given to the mezzanine through signals or messages issued from the root node.
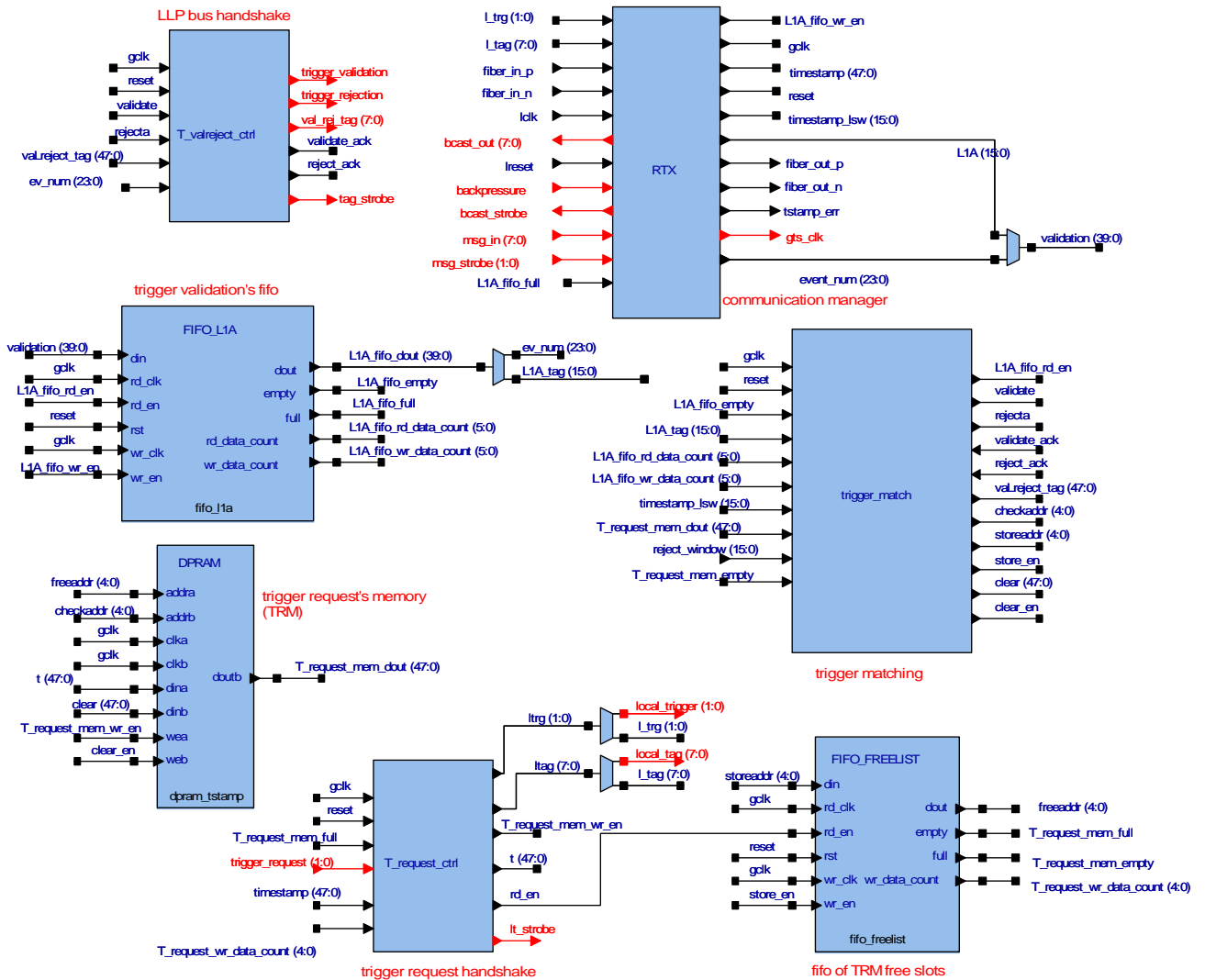
Fig. D2 – Internal block diagram of the mezzanine model.

The figure D2 shows the internal of the GTS mezzanine block. Each sub-block takes care of a given task and coordinates its activity with the others. Below is a list of the blocks and a description of the basic function they accomplish.

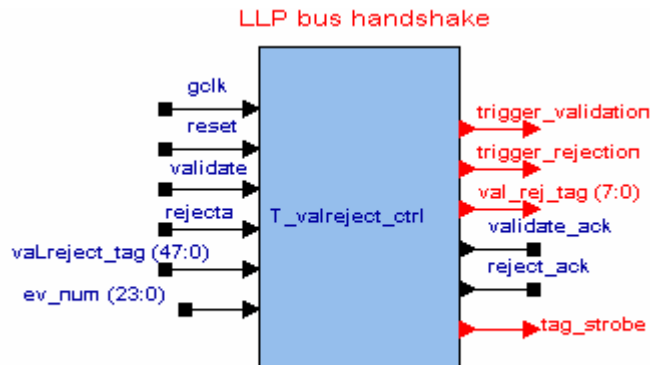## D.1 *T_valreject_ctrl.*

(top left of the fig D2)

Fig. D3 – The T_valreject_ctrl block.

It is in charge of communicating a trigger request validation or rejection on the LLP/ancillary bus together with the relevant informations needed for this task, e.g. the 48bit validation or rejection tag that identifies the request and the event number in case of validation. It drives the val_rej_tag(7:0) bus according to a predefined handshake mechanism upon request of the two incoming signals "*validate*" or "*rejecta*" that are driven by the *trigger_match* block (middle right side of the picture).
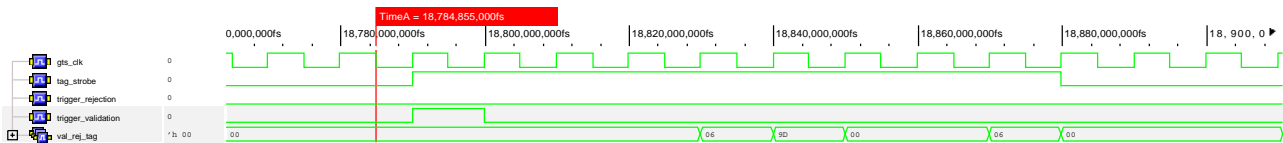
Fig. D4 – Trigger validation waveforms.
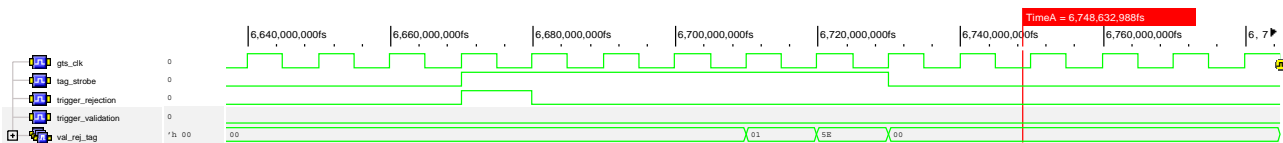
Fig. D5 – Trigger rejection waveforms.

D.1.1 T_valreject_ctrl internal block diagram

The simple state machine reported in the following figure highlights the expected behaviour of this block. Upon an external decision of either validation or rejection of an event one of the two possible branches is followed, producing on the val_rej_tag bus the sequence of event identifier + event number (validation) or only the event identifier (rejection).
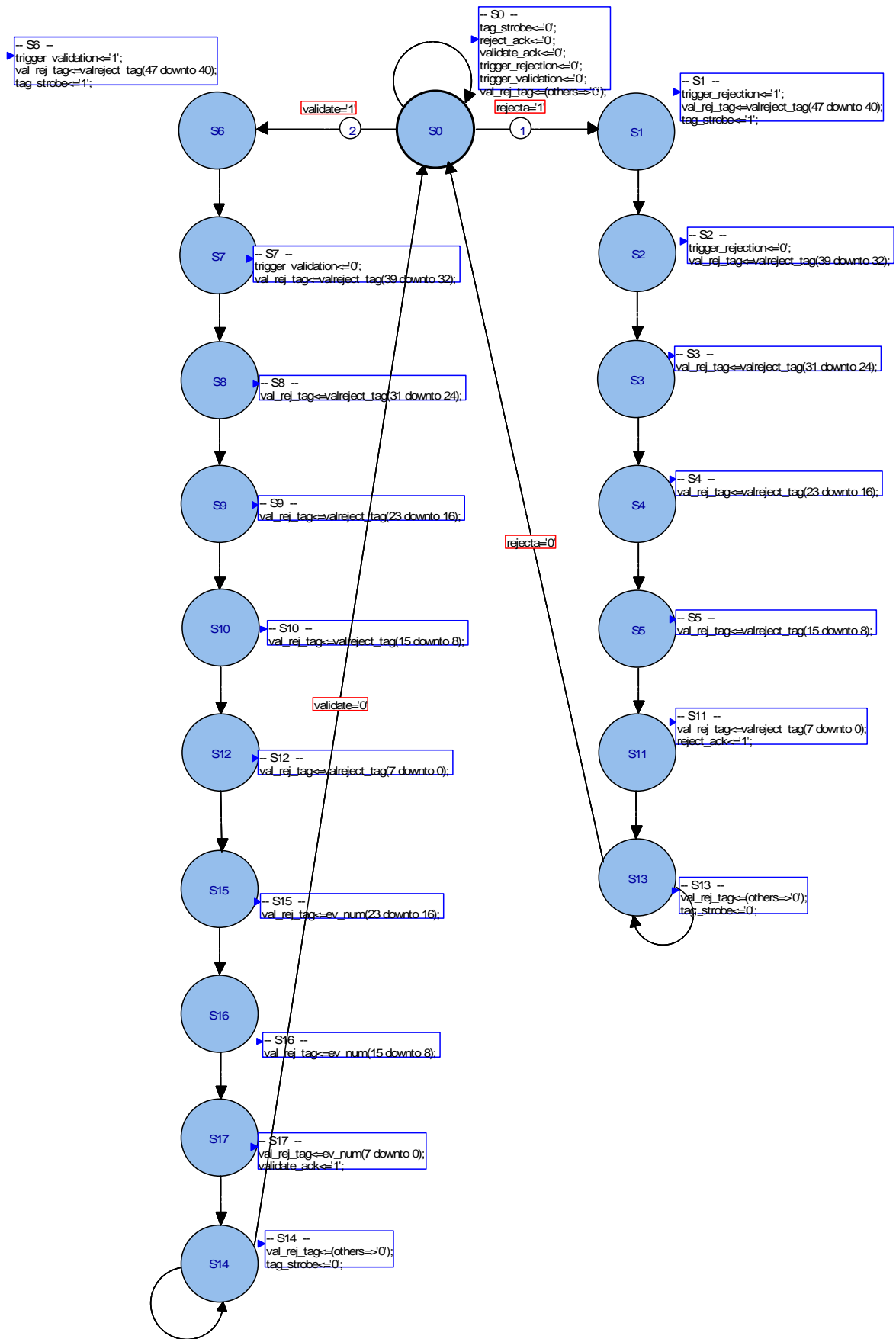
Fig. D6 – T_valreject_ctrl state machine.

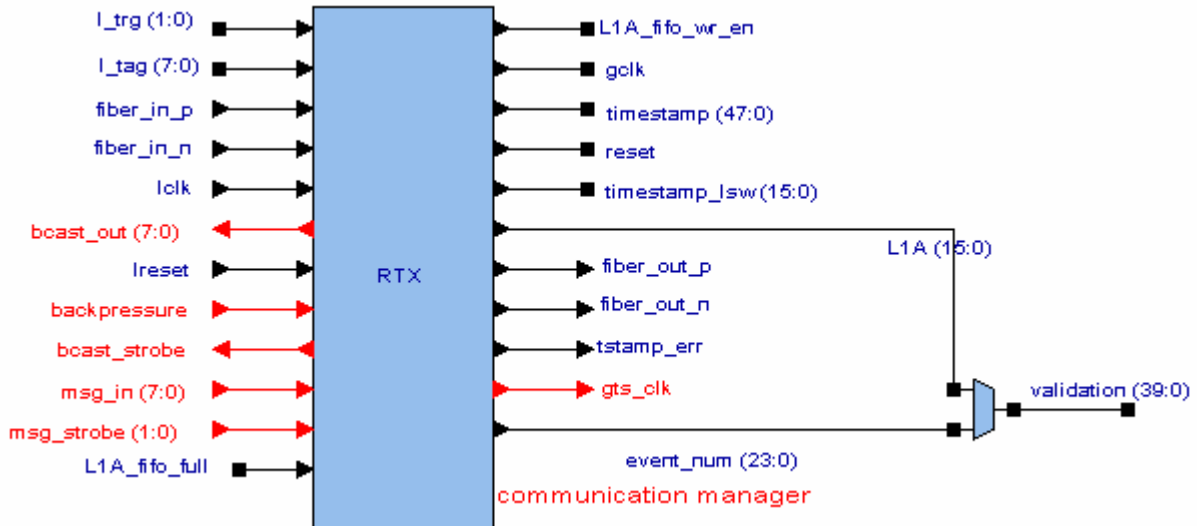## D.2 RTX.

(top right of the fig. D2)

Fig. D7 – The RTX block.

It manages the high speed serial communication channel with the Root node (it sinks the *fiber_in* signals and sources the *fiber_out* signals). It decodes the 48bit timestamp pattern, the global clock (*gts_clk*) and all the messages coming from the root, e.g. the global reset (*reset* signal) and the trigger validation (*L1A* signal) for the time being. The RTX communicates with the root by means of bidirectional messages that are Hamming encoded for increased reliability. In the uplink direction it sources trigger request messages to the root node with a payload specifying an identifier for each individual request. It sources also periodically an "idle" message for use in the central trigger processor.

D.2.1 RTX internal block diagram

The block in violet hides the simulation core of the multi-gigabit transceiver. The 16bit receiver bus *rxdata* is split in (top left part of the figure) the least significant byte (lsb) and most significant byte (msb). The lsb carries the encoded timestamp and gets decoded by passing through 2 lookup tables and a t_decode block that implements the decoding algorithm (due to D. Bazzacco). The msb carries Hamming encoded commands: to retrieve these commands the msb is passed through a decoder block (cmd_dec) that retrieves from the stream the validation informations (L1A and event number) the global reset and the commands to be routed to the bcast_out bus. A cmd_enc block is in charge of encoding commands in the upstream direction (txdata bus of the transceiver). For the time being the encoded commands are trigger requests, idle messages and the backpressure throttle.
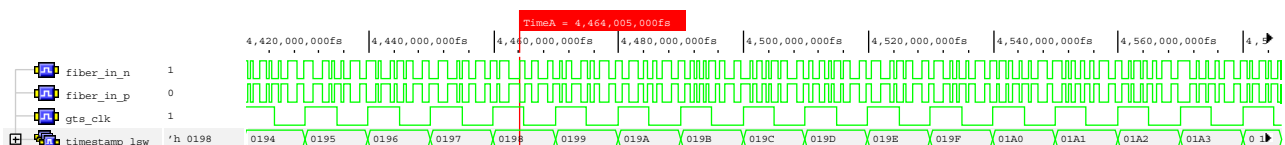

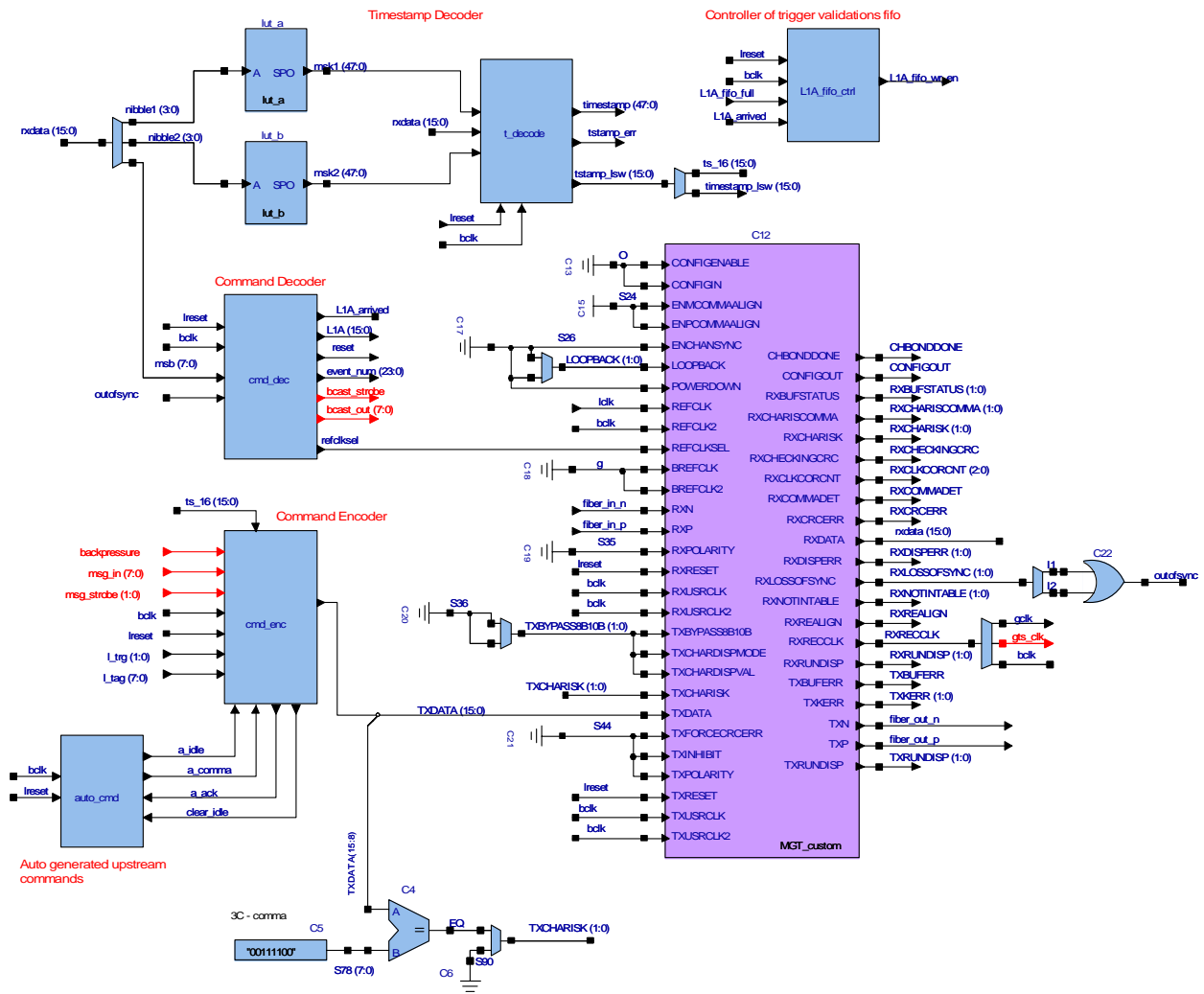**Fig. D8 – Global clock, timestamp and gigabit transceivers waveforms.**

Fig. D9 – RTX block diagram.

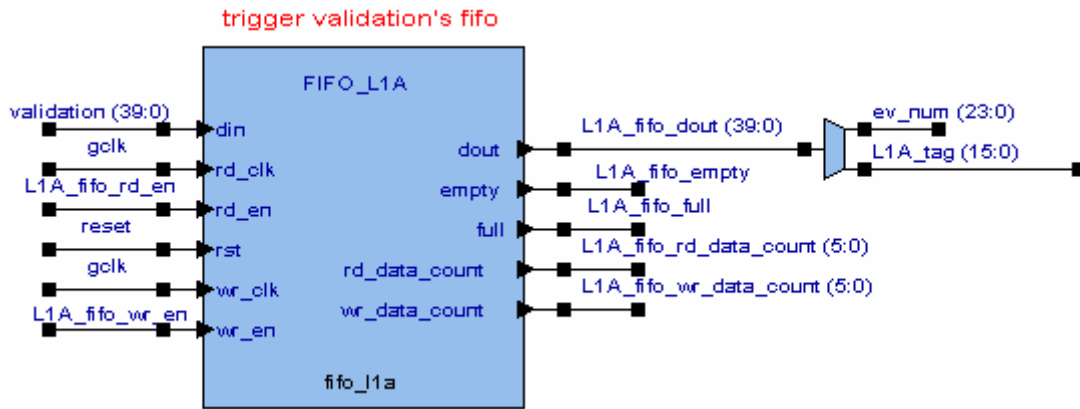## D.3 FIFO_L1A.
(middle left of fig. D2)

Fig. D10 – The FIFO_L1A block.

This block is a first-in-first-out memory that stores temporarily incoming trigger validations (and their identifiers) waiting to be served by the *trigger_match* block.
It's a de-randomizing memory with a depth of twenty locations.

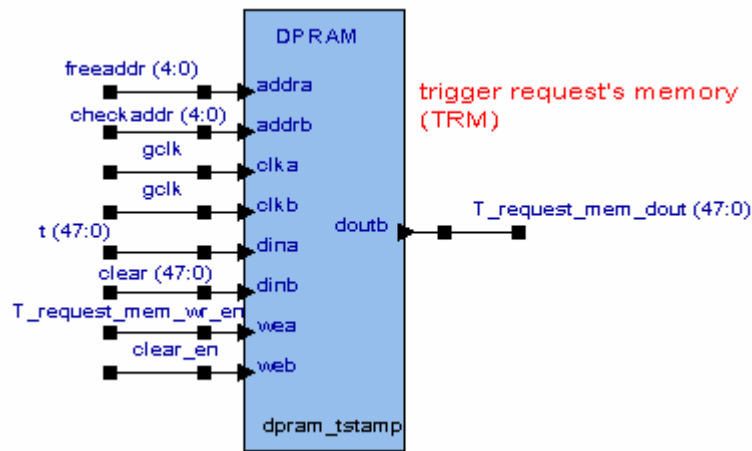## D.4 DPRAM_tstamp.

(low left of fig. D2)

Fig. D11 – The DPRAM_tstamp block.

It'a true dual-port memory holding the trigger requests and their identifiers originated by the LLP/ancillary front-end electronics. The pending trigger requests and associated tags are kept in this memory until validation or rejection or timeout expiry. As the order of arrival of trigger validations is not guaranteed (possibly not time-sorted), the possibility exists of accessing this memory not consecutively, hence the need of a random access memory type.

D.5 FIFO_FREELIST.
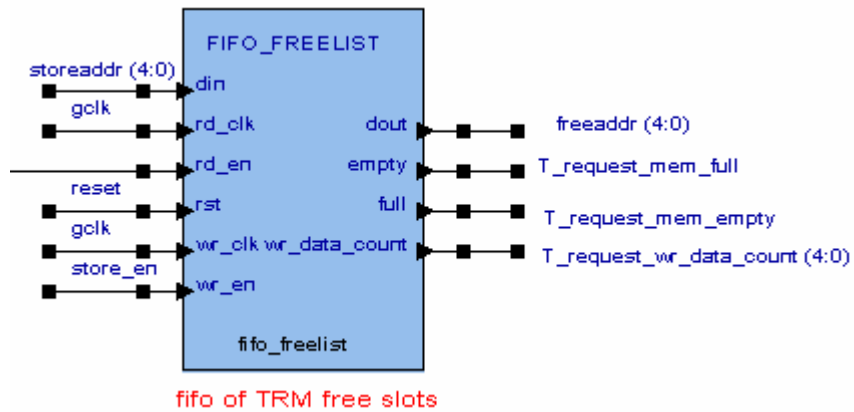
(low right of fig. D2)

Fig. D12 – The FIFO_FREELIST block.

For the reason stated above, there is a need to keep a list of free location addresses in the *DPRAM_tstamp* memory. The FIFO_FREELIST is a first-in-first-out memory holding this list.

D.6 Trigger_match.
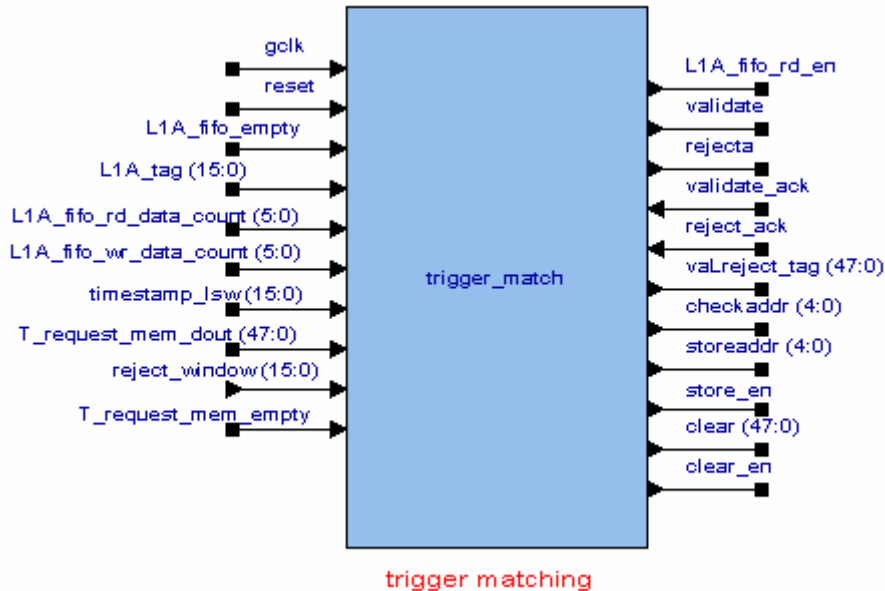
(middle right of fig. D2)

Fig. D13 – The Trigger_match block.

This is a state machine in charge of checking if there is correspondence between the current (the oldest actually) trigger validation and any of the pending trigger requests. In case of match or timeout a proper action must be undertaken (e.g. validation or rejection of that event must be notified on the LLP/ancillary bus by means of the *T_valreject_ctrl* block). To do this, the *Trigger_match* block reads the *L1A_fifo* memory and if a validation exists scans the *DPRAM_tstamp* to look for the corresponding trigger request. It may happen that, during the scan, one or more trigger requests originated at a time that exceeds a fixed timeout are found.
In both cases this must be notified to the LLP/ancillary bus by asserting the appropriate signal between "*validate*" and "*rejecta*".

D.6.1 Trigger_match internal block diagram

The overall state machine that governs the sequencing of operations is shown in the following figure D14. After reset the FIFO_FREELIST is filled with free location addresses of the DPRAM_tstamp memory. During normal operation the machine continuously scans the DPRAM_tstamp memory looking for expired trigger requests to discard. But when a trigger validation is arrived, the machine scans the DPRAM_tstamp memory looking for trigger requests that match the validation. The sequence of operation in this case is shown in fig. D15.
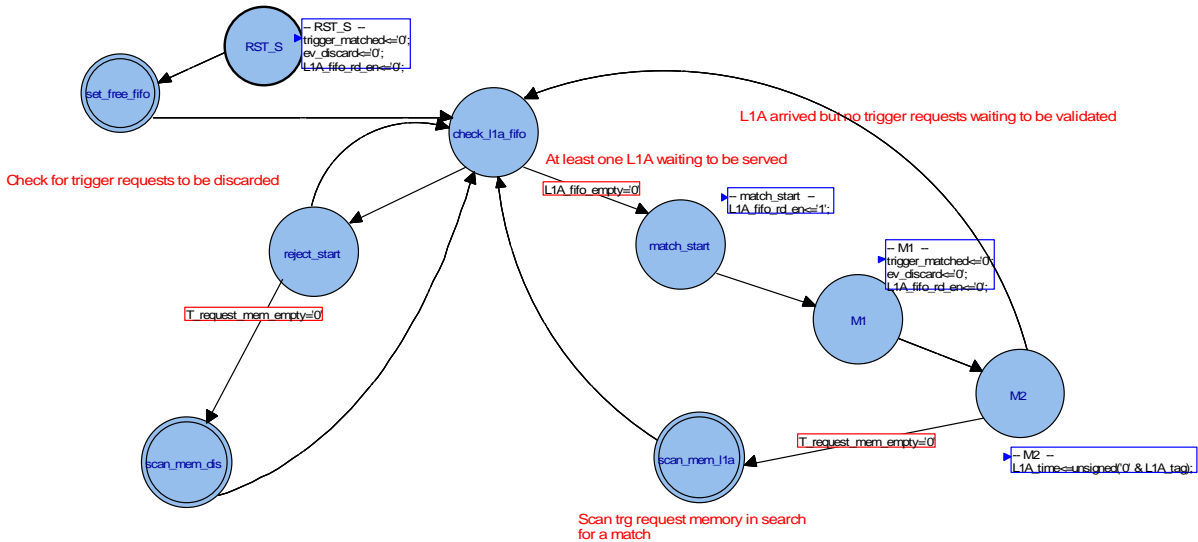
Fig. D14 – The Trigger_match state machine.

D.6.2 Scan_mem_l1a internal block diagram

Upon arrival of a trigger validation this sequence is entered. Each trigger request found in the DPRAM_tstamp memory is compared with the validation tag and one of three possible choices is evaluated (trigger request has matched, trigger request is expired, trigger request is more recent than validation and must be left in place for future validations).
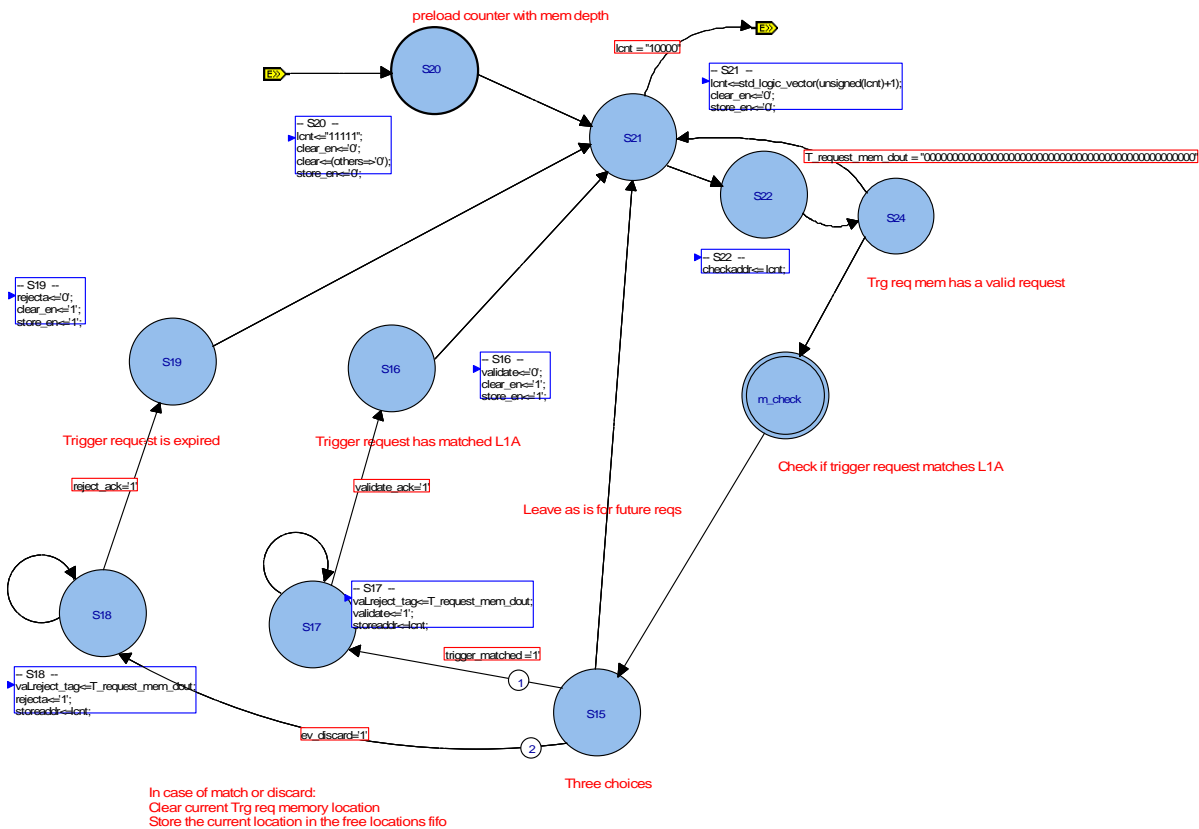

Fig. D15 – The scan_mem_l1a state machine.

D.7 T_request_ctrl.
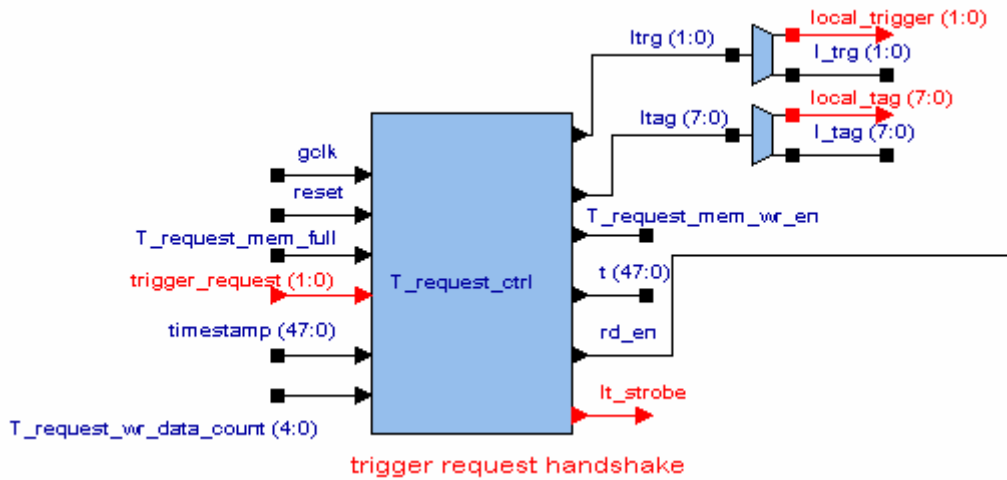
(low middle of fig. D2)

Fig. D16 – The T_request_ctrl block.

The block manages the handshake with LLP/ancillary electronics after an incoming trigger request. A tag is associated with any given trigger request and broadcasted on the *local_tag* bus. The same tag is also stored on the *DPRAM_tstamp* memory after having asked for a free location to the *FIFO_FREELIST* fifo. Note that the tag associated with any given trigger request is actually the 48bit global clock counter value latched at the time of the request. Figure D17 highlights the handshake sequence on the LLP bus that follows a trigger request.
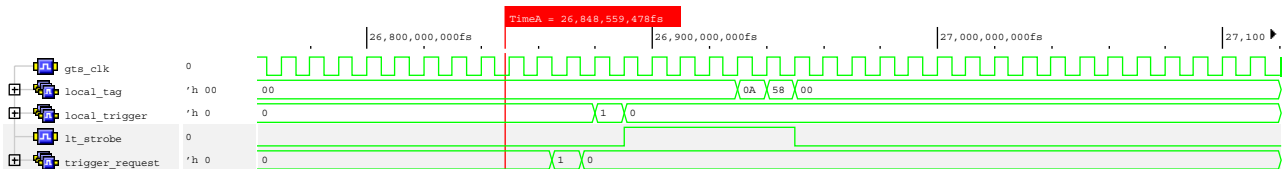

Fig. D17 – Trigger request waveforms.

D.7.1 T_request_ctrl internal block diagram

The simple state machine reacts upon a trigger request made on the LLP/ancillary interface. It latches the current value of the 48bit global timestamp and stores it in the *DPRAM_tstamp* memory. Immediately after, it sends the same timestamp, 8bit at a time, on the *local_tag* bus for use as an event identifier in the LLP/ancillary electronics.

trigger_request="01" AND T_request_mem_full='0'

-- S0 --
T_request_mem_wr_en<='0';
ltag<=(others=>'0');
ltrg<='00';
lt_strobe<='0';
t<=(others=>'0');

-- S9 --
tstamp<=timestamp(47 downto 0);
rd_en <= '1';

-- S1 --
ltrg<="01";
T_request_mem_wr_en<='1';
t<=tstamp;
rd_en <= '0';

-- S2 --
ltrg<="00";
T_request_mem_wr_en<='0';
ltag<=tstamp(47 downto 40);
lt_strobe<='1';

-- S7 --
ltag<=tstamp(7 downto 0);

-- S6 --
ltag<=tstamp(15 downto 8);

-- S3 --
ltag<=tstamp(39 downto 32);
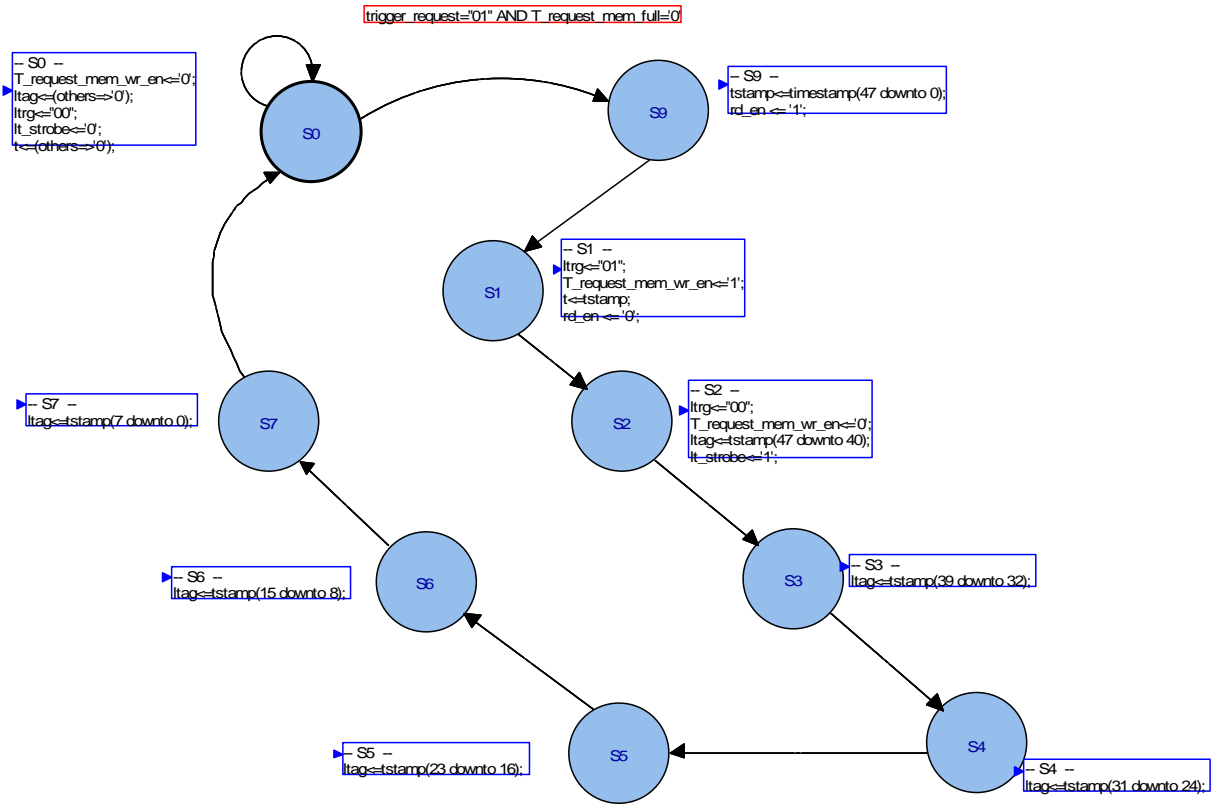
-- S5 --
ltag<=tstamp(23 downto 16);

-- S4 --
ltag<=tstamp(31 downto 24);

Fig. D18 – T_request_ctrl state machine.