Notes from AGATA LLP Interfaces meeting held at CSNSM Orsay on 24th March 2004

Present: Marco Bellato, Lounis Benallegue**,** Damiano Bortolato, Patrick Coleman-Smith, Pierre Edelbruck, Werner Gast (morning only), Xavier Grave(morning only), Roberto Isocrate, Ian Lazarus, Sebastien Lhenoret, Christophe Oziol, Alberto Pullia, Cayetano Santos, Bruno Travers, Dirk Weisshar. (Reiner Kruecken, Juergen Eberth and Dino Bazzacco also attended for short periods.)

Introduction and presentations

After a short introduction from Reiner Kruecken there was a series of presentations about the preamplifier (Alberto Pullia) the Digitiser (Patrick Coleman-Smith) the Pre-Processing (Sebastien Lhenoret), the GTS (Marco Bellato) and System Simulation (Marco Bellato). These presentations are included in the appendix at the end of these notes.

During and after the presentations many questions were discussed and later some people agreed to continue working on the answers to the questions. These notes are an attempt to record the important points from the discussions and our decisions during and after the presentations.

The names against the tasks are the ones we identified in the meeting. If other people want to contribute too then they should contact the people doing the job and offer help. I realised that we forgot to make anyone responsible for starting the discussion within each task. Please don't wait for someone else to do it!

The timescale agreed was that where possible (within the limitations imposed by meetings of other groups) the tasks set out in the rest of the document will be completed (or showing significant progress) within 1 month.

The remaining part of these notes describes the background, discussion and tasks allocated for these ten points:

1) Preamplifier: Inhibit output.
2) ADC Input offset: usage and control.
3) How do we synchronise (line up) all the channels?- dealing with link latencies.
4) Control of the preamplifier's pulser
5) Digitiser and pre-processor spare channels
6) Slow Control Connections
7) How to identify correlation between local triggers and validations
8) Data buffering in the pre-processing
9) System simulation model
10) GTS Mezzanine Pinout

I have added a couple of comments (*in purple italics*) which came to mind while writing the notes.

1) Preamplifier: Inhibit output.

Background
The preamplifier first stage is designed so that it will not saturate even with 50MeV pion events occurring at a rate of 10kHZ. However the second stage can saturate either due to these 50MeV events or a sustained high rate of lower energy events. So the preamps have a mechanism for fast discharge of the second stage using a switched current source, controlled by a voltage comparator which compares the output to a predefined threshold. The signal which enables the switched current source is sent back to the digitiser and labelled "Inhibit". By measuring the duration of Inhibit the amplitude of the saturating pulse can be measured. This combination of current source and time-over-threshold-counter is effectively a Wilkinson ADC and so can be used to measure energy. Tests with a pulser show that the Inhibit pulse has an intrinsic noise jitter of about 4ns rms and a width in the range from 1-10us for a typical energy range with good linearity.

Discussion
At the moment the preamp sends Inhibit to the digitiser, and receives an enable/shutdown signal from the ADC to shutdown or enable the fast reset comparator. The digitiser system sends the Inhibit bit to the pre-processor in the unused (D15) data bit with the ADC data samples (14 bits of data, 1 bit of ADC Over-flow, 1 bit Inhibit). It is proposed to measure Inhibit only on the core contact, not the segments. In fact the segments are grouped so that 3 segments share a single Inhibit, so such a method can't be used anyway. It was noted during the discussion that the pion's 50MeV will usually be localised in 1 segment.

Two possible changes were discussed: firstly to add a TDC in the digitiser using a frequency higher than 100MHz to measure the width of the Inhibit to the best available accuracy. This was rejected for 2 reasons- firstly there is no spare capacity in the data links to the pre-processor so a new fibre link would be needed and secondly it is not obvious how this data could be exactly time-correlated with the ADC data streams. Instead the Inhibit signal will be sampled every 10ns and sent with each ADC data sample as previously planned. *(NB there is a latency in sub-ranging FADCs; 4 clocks in AD6645, 8 clocks in AD9245, 14 clocks in the AD9444 and 16 clocks in ADS5500, so the Inhibit appears in the pre-processor's input data stream before the associated ADC data words.)*

The second possible change which was discussed was to move the control of the fast reset into the digitiser. In this case the ADC's full scale is used as the threshold and the ADC's over-range bit is sent back to the preamplifier to switch on the current source which discharges the 2nd stage of the preamplifier without any further check of the preamplifier output voltage. The advantage of this method is that it saves one connection between digitiser and preamp. The disadvantages are that both the start and the end of the preamplifier discharge is delayed by the ADC output latency period (40 to 160ns depending on the ADC selected for the digitiser) and that the preamplifier becomes dependent on the AGATA digitiser for its fast reset operation rather than a self-contained unit useable in other applications. This discussion was not concluded and no decision was taken.

There were 2 tasks allocated as a result of this discussion:

Task 1
(Alberto Pullia, Dirk Weisshar, Werner Gast, Patrice Medina, Ian Lazarus, Denis Linget)

a) Discussion of the implementation of the fast reset (whether it is to be controlled locally in the preamplifier or remotely by the digitiser's ADC) and whether there is any advantage in timing Inhibit on segments as well as the core (not possible with shared inhibit).
b) Discussion with pre-processing algorithms team about implementing an algorithm which adds the result of measuring the reset pulse width (either from Inhibit or ADC over-range) to the ADC value prior to saturation. This algorithm must take account of the ADC output latency and make appropriate corrections. The size (processing power) of FPGAs in the pre-processing should be reviewed to see if the new algorithm fits in the planned devices.

Task 2
(Juergen Eberth)
Discuss with AMB whether the proposed energy resolution for pions of between 0.2 and 2% (depending on pulse width and hence the energy) using the "free" Inhibit width measurement is good enough. If not then additional money must be spent to add a TDC and a readout fibre to each digitiser.

2) ADC Input offset: usage and control.

Background
The preamplifier output range is set to 100mV/MeV which means that with no baseline offset the digitiser ADC can accept 10MeV full scale. This range is reduced by baseline shifts caused by preamp output pulses sitting on the tails of previous pulses. The digitiser has 2 mechanisms to ensure that the input stays in the ADC's operating range: first a coarse gain switch (x1 or x 0.25 attenuation) and secondly a DAC which can shift the baseline to compensate for rate-related baseline shifts. The preamplifier's fast reset mechanism (see previous section) can be regarded as another way to regulate the baseline shift in cases with extremely high energy-rate products.

Discussion
Questions concerning the offset DAC were discussed-

- who controls it?
- what is the algorithm for setting the DAC?
- what effects do changes in the baseline offset have on the signal processing?
- how does it inter-work with the preamplifier fast reset
- how fast does it need to be updated (i.e. what fibre bandwidth is needed assuming it is externally controlled by the pre-processing)

The answers we came up with were:

- Q Who controls it?
- *A The digitiser offset DAC must be controlled by the pre-processor.*

- Q What is the algorithm for setting the DAC?
- *A We don't know- this is a question for the pre-processing algorithm group (see Task 3). Its not as simple as converting an average baseline value to a DAC setting to shift the ADC input because of*

situations such as pulsed beams where the rate can change very fast and also because the rate of change of the offset is important- this is explored more in the next question.

- Q What effects do changes in the baseline offset have on the signal processing?
- *A If the baseline is changed by the offset DAC during the MWD processing time then it will affect the energy resolution because new values and stored values will have different offsets and so differences between samples will be increased or decreased by the change in offset. In this case we would need to calibrate the offset DAC. With a DC input, values would be set in the DAC by the pre-processor and the changes in the ADC code noted. This calibration allows the DAC setting to be corrected in a way analogous to a sliding scale correction. This is certainly not trivial on a sample by sample basis over all possible DAC values and needs to take in to account the ADC output latency. However if the rate of change is slow compared to the MWD processing then the effect is one of changing the pole zero circuit to give an apparent increase or decrease in the preamplifier output decay time depending on whether the offset is added or subtracted. This will affect the signal processing in that the PZ correction will need to be adapted to account for the offset change.*

- Q How does it inter-work with the preamplifier fast reset
- *A As noted in the initial discussions, the fast reset and the digitiser offset should not be used simultaneously. It was noted that high energy pulses from pions will not be a problem in all experiments so the preamplifier's fast reset is not required in all experiments.*

- Q How fast does the offset need to be updated (i.e. what fibre bandwidth is needed assuming it is externally controlled by the pre-processing)
- *A We didn't come up with an algorithm to control the offset adjustment, so it was not possible to calculate the required update rate and hence the required fibre bandwidth. The decision to continue to control the offset from the pre-processing does, however, confirm that fibre links are needed.*

Task 3
(Alberto Pullia, Werner Gast, Patrick Coleman-Smith, Sebastien Lhenoret)
Discuss with pre-processing algorithms team about implementing an algorithm to control the digitiser's input offset (assuming that it doesn't operate at the same time as the preamplifier reset). This algorithm must take account of the effects of changing the baseline on the other pre-processing algorithms and also consider both fast baseline changes (e.g. pulsed beam) as well as gradual changes. Consideration should also be given to the impact on Pulse Shape analysis (e.g. does the PSA need to know the offset setting and when it changed?) The result of this discussion will affect the selection of the fibre link from pre-processing to digitiser. The size (processing power) of FPGAs in the pre-processing should be reviewed to see if the input offset algorithm fits in the planned devices or not.

3) How do we synchronise (line up) all the channels?- dealing with link latencies.

Background
During the digitiser and pre-processing presentations the question of synchronisation and link latencies was raised several times. The problem is that as a fast serial link is started up, the receiver must lock onto the incoming data stream, deciding where a word starts (which is bit 0?) and this takes some time.

Different links will take different times to do this. In fact the same link will take different times to do this on different restarts. This startup variation means that different links will have different latencies- some will have locked to the clock earlier than others, so will send their data out earlier. The latency affects the links in both directions (to/from GTS; from pre-processing to digitiser and from digitiser to pre-processing). It must be calibrated out dynamically (because it changes every time the link is started its no good to re-use old values).

Discussion
How do we calibrate the link latencies? It was noted that a return path is needed from the signal source to the receiver and back again in order to calibrate the latency- only the transmitter can know when it sent the signal, so it must also be the place to receive the returned copy of the signal and count the time difference. Such return paths are built into the system. In normal operation they are not set up to loop back calibration pulses, so a special setup/training mode will be required. For example the pre-processing can send a command with the clock signal to the pre-processor which can be distributed within the digitiser to all data paths (if the digitiser is in calibration mode). The pre-processor would wait to see when it comes back down each of the data links from each segment and the core, noting the differences. In normal operation these calculated latency variations will be used to set different latency compensation delays in the pre-processing input buffers so that all ADC sample data streams are time-aligned across the whole crystal before processing them. After the latency compensation delays all data relating to a certain instance in time (when a physics event occurs) will emerge from the delays in parallel, ready for processing together.

Finer alignment than 10ns would require the introduction of new timing verniers in the digitiser to phase-shift the ADC clocks. This is not included in the present specification and would incur additional cost and manpower to implement it. The question was raised as to whether sub-clock alignment is required by PSA (or even if it is possible to do).

Task 5
(Ian Lazarus)
Ask PSA group (Thorsten Kroell) whether sub-clock (better than 10ns) time alignment is necessary for PSA and if so how important is it and what time alignment should we aim for. Note that anything better than 10ns requires new money and extra manpower!

Task 6
(Damiano Bortolato, Patrick Coleman-Smith)
Produce a short proposal for how to do the training and calibration for link latencies and clocks for the digitiser, pre-processing and GTS.

4) Control of the preamplifier's pulser

Background
The preamplifier for the core segment has a built-in pulser which couples capacitively through the detector to all 36 segments, allowing the testing of back segments which would see very low count rates from a radioactive source on their crystal's front face. The amplitude and start time of this pulse

are controlled by the digitiser cards in response to commands sent via the slow control. What is not clear is whether there is a need in the digitiser to convert a single slow control command to a sequence of commands to the pulser (for example to slowly increment amplitude over 100 000 pulses to make a ramp to test linearity). It is simpler just to make a 1-1 correlation between the slow control and pulser commands. However the example of the ramp would need 200 000 slow control cycles to change the amplitude 100 000 times and to trigger the pulser 100 000 times. The possibility of firing multiple pulsers simultaneously to test timing was also raised.

Discussion
It was agreed that the digitiser core card will provide all the connections internally to implement state machines, such as the ramp, if necessary. This requires signal paths from slow control to FPGA and from FPGA to pulser control signals.
Testing timing with the pulser is not so easy because of the link latency problems described previously. The link latencies are corrected at the input to the pre-processor such that the signals are in line there. But in the digitisers the latency is not corrected, so the variation will result in a spread of pulse times even if a global start command is issued via the GTS global command link to all pre-processors. So making the detector pulsers useful for timing is not possible without spending extra money to provide a way to calibrate latencies in the digitisers. It was agreed that this decision to use the pulsers for energy only (not timing) can be reconsidered if a real need (backed up by extra money!) is identified. What is required is one more fibre from the digitiser to the pre-processor which is tightly coupled to the fibre issuing global commands and clocks. The new feedback fibre would be connected either to loop-back in the transceiver receiving the global clock and commands (no latency) or to loop back using the output of the global clock receiver (with latency) so that the pre-processor can measure the difference.

No tasks identified.

5) Digitiser and pre-processor spare channels

Background
Some redundancy is built into both the digitiser and the pre-processor so that faulty channels can be switched out and replaced by spares. For the core, the digitiser has 1 spare channel which is switched by enabling/disabling input buffers before the ADCs. There was some discussion about whether the spare channel needs its own fibre link or whether one fibre can be switched in the serialiser FPGA to accept either of 2 ADCs as its data source. For the segments, each group of 6 outputs runs down a 12 way ribbon fibre and one extra digitiser channel is provided to drive a 7^{th} fibre (5 unused). Analogue inputs are switched to the spare channel using a switched input buffer.

Discussion
It was agreed that the core is important for triggering and so it should have 2 complete data paths through the digitiser card and 2 fibres to the pre-processing. It was decided that the pre-processing will receive both core fibres and will use FPGA reconfiguration to choose which is processed. Similarly for the segments, FPGA reconfiguration will be used. In the case of segments there is one FPGA per 2 inputs (3 FPGAs per group of 6)- the spare channel can only be connected to 1 FPGA, so if the failed channel is in one of the other 2 FPGAs then the FPGA with the spare input must process 3 inputs

instead of 2 after reconfiguration. Therefore this FPGA will need to be resized (note that FPGA dimensioning needs to be re-checked anyway as a result of the offset algorithm (point 2).

Task 7
(Patrick Coleman-Smith and Sebastien Lhenoret)
Consider the use of spare channels in the digitiser and the pre-processor and check that what is proposed here will really work for switching spare channels (FPGA reconfiguration, software reconfiguration, database reloads for detector-specific parameters etc..)

6) Slow Control Connections

Background
During the discussion about the digitiser it became clear that we have not yet decided what happens about the slow control connection, in particular where it comes from. In the pre-processor the GLP's slow control Ethernet is connected to the carriers via RJ45 front panel connections and connected to an embedded PowerPC in the Xilinx FPGAs. An option proposed for the digitiser is to use a small external PC with Ethernet connection which drives several USB connections to the digitisers. This solution requires PCs to be bought (probably shared between several digitisers, so the cost would be low but care would need to be taken with earthing). Another option for the digitiser is to copy the method used in the pre-processor (RJ45 connector and PowerPC in the Xilinx FPGA). This requires isolated fibre Ethernet. A third option is to have a bidirectional slow control fibre link between pre-processor and digitiser (no Ethernet port on digitiser).

Discussion
The options described above were discussed a little and no firm conclusions reached. Input from the GLP group (in charge of slow control) is also needed.

Task 8
(Patrick Coleman-Smith, Xavier Grave, Christophe Oziol, Dino Bazzacco (GLP), IRes (Marc Richer?))
Look at options for connecting slow control to digitiser (direct, via PC or via pre-processing) and make a recommendation as to which should be used.

7) How to identify correlation between local triggers and validations

Background
The pre-processor interaction with the GTS was an area where we identified some confusion, so we discussed an example where multiple local triggers are generated in the time when we are waiting for a validation for the first. The question is this- which of the local triggers is to have its data selected for readout? The proposed GTS method is that the GTS trigger decision has a latency depending on the physics of the trigger (but limited to 20us by pre-processing buffer depth as agreed in Legnaro in September 2003). Associated with the latency is also an acceptance window. Local triggers are

accepted if they were generated before the validation in the time period defined by the latency period and the acceptance window. To help keep track of this process, the GTS mezzanine will update a full 48 bit timestamp every clock cycle *(how does this get sent to the other carriers?).* When the trigger validation is issued, it will come with an event number (24 bits) which is actually redundant in a timestamped system, but will be useful for diagnostics. It will also be associated with a particular timestamp from which the trigger latency and acceptance window limits are subtracted to identify the range of timestamps for which local triggers are accepted.

The pre-processing will take the ADC samples, extract the energy, the leading edge and perform any other processing. Then the resulting data will be put into a buffer and identified by a tag such as the timestamp of the local trigger which caused them. This is the data buffer which is searched after a trigger validation. *(What happens if the data are not yet ready when validation comes back?).* Data older than the oldest acceptable timestamp are wiped from the buffer (validations will always come in time order and with known latency so old data can be confidently discarded). Data which is too new to be accepted is retained.

Discussion
It was noted that for wide acceptance windows it is possible to have multiple local triggers and also for a single local trigger to be in multiple acceptance windows. It is not clear whether data from the local trigger should be discarded from the main buffer when it is written to the readout buffer after validation. It is possible that the same local trigger might be in a following validation too. It is not clear whether in this case data should be duplicated or not in the readout buffer. More discussion on this is needed between GTS, physicists, PSA and the pre-processing. There was also a lot of discussion about how close the local triggers can come because of pileup and the SCC algorithm's deadtime- a minimum gap of 1us was suggested as a realistic gap.

Task 9
(Marco Bellato, Xavier Grave, Ian Lazarus, Sebastien Lhenoret)
Consider and discuss the proposal presented for data selection based on trigger latency and acceptance windows. Consider how to implement it and look at alternative trigger protocols.

8) Data buffering in the pre-processing

Background
During the presentation of the pre-processing there was some discussion about the need for a dual ported memory on the carrier board to permit data to be written from mezzanines whilst being read from Compact PCI. The behaviour of this memory, its size and i/o bandwidth requirements can be studied using the System C model of the LLP developed by Marco Bellato.

Discussion
It was agreed that more discussion is needed between the pre-processor carrier designers and the people studying the system model (see item 9) to be sure that the output bandwidth and memory size are sufficient and that different options are modelled.

Task 10
(Xavier Grave, Christophe Oziol with system model people (Marco Bellato, Lounis Benallegue, Patrick Coleman-Smith, Christophe Oziol))
Consider dimensioning of the possible pre-processor carrier memory architectures and the necessary i/o bandwidth (using system model if necessary).

9) System simulation model

Background
Following earlier meetings a model for part of the LLP system has been developed in SystemC which permits system modelling at the clock cycle level. The model covers the pre-processing and GTS, allowing the study of trigger and readout mechanisms. The work now requires more people to get involved in developing the model's components and studying the effects of alternative component implementations (see for example point 8) and of changes to the system.

Discussion
There was some discussion about the availability of software tools (the System C software is free but a viewer needs to be purchased). The possibility of extending the model to cover latencies in the digitiser ADCs and transceivers was discussed.

Task 11
(Christophe Oziol, Marco Bellato, Lounis Benallegue, Patrick Coleman-Smith)
Continue to develop and test the system model, using it to test effects of changes to the system arising out of today's and any future discussions and to compare alternative implementations.

10) GTS Mezzanine Pinout

Background
The proposed mezzanine pinout is currently not compatible with the proposed carrier pinout for mezzanines, so some discussion and changes will be needed to converge the 2 proposals. The GTS mezzanine will be used also for the GTS fan-in/fan-out and GTS control functions so that only 1 design is needed although different FPGAs and different numbers of transceivers will be fitted in the different applications. Also the component heights and placements proposed for the mezzanine and carrier need to be checked for compatibility.

Task 12
(Marco Bellato, Denis Linget or Lounis Benallegue, Roberto Isocrate, Christophe Oziol)
Ensure that the carrier and the GTS mezzanine have compatible pinouts, board size, connector placements and component placements.

Appendix- Presentations.

Preamplifier- Alberto Pullia

Digitiser- Patrick Coleman-Smith

Pre-Processing- Sebastien Lhenoret

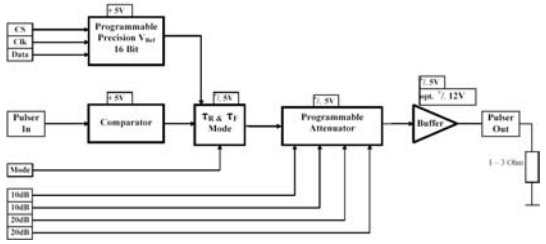
GTS- Marco Bellato

GTS-Pre-processing system model- Marco Bellato.

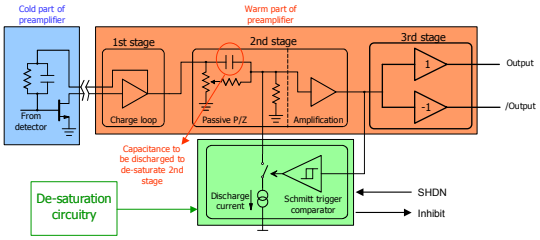
AGATA preamplifiers & interfaces

A. Pullia - AGATA LLP Interfaces meeting, Orsay 24 March 2004

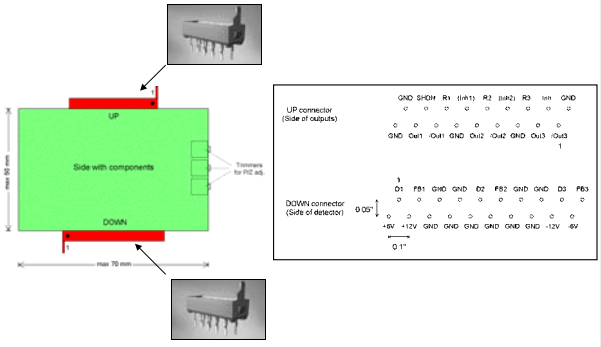
Built-in pulser structure



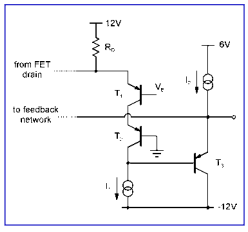
Preamplifier architecture



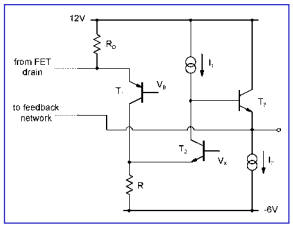
Size and pinout - segment preamps



First-stage structure

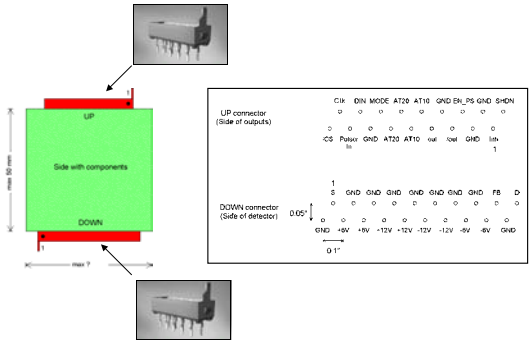


Segment-preamplifier first stage

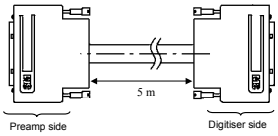


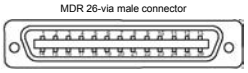
Core-preamplifier first stage

Size and pinout – core preamp & pulser



Cables & interface specifications





Pin/twisted-pair association

Preamp side	Digitiser side	Cable name	Preamp side	Digitiser side	Cable name	Preamp side	Digitiser side	Cable name	Preamp side	Digitiser side	Cable name
1	1	shield	17	10	pair 3+	8	19	pair 7-	24	3	pair 10+
14	14	shield	5	22	pair 4-	21	6	pair 7+	12	15	pair 11-
2	25	pair 1-	18	9	pair 4+	9	18	pair 8-	25	2	pair 11+
15	12	pair 1+	6	21	pair 5-	22	5	pair 8+	13	13	shield
3	24	pair 2-	19	8	pair 5+	10	17	pair 9-	26	26	shield
16	11	pair 2+	7	20	pair 6-	23	4	pair 9+			
4	23	pair 3-	20	7	pair 6+	11	16	pair 10-			

Complete signal list

Type	*Name*	*Source*	*Type / format*	*Quantity*
Analog	Segment	Preamp	Analog signal / 1 differential pair	36
	Core	Preamp	Analog signal / 1 differential pair	1
Digital	Inh_A	Preamp	Digital Signal / +5.x V = High, GND = Low	6
	Inh_B	Preamp	Digital Signal / +5.x V = High, GND = Low	6
	Inh_C	Preamp	Digital Signal / GND = High, -5.x V = Low	1
	SHDN_A	Digitiser	Logic Level / +5.x V = High, GND = Low	6
	SHDN_B	Digitiser	Logic Level / +5.x V = High, GND = Low	6
	SHDN_C	Digitiser	Logic Level / GND = High, -5.x V = Low	1
	AT10	Digitiser	Logic Level / +5.x V = High, GND = Low	2
	AT20	Digitiser	Logic Level / +5.x V = High, GND = Low	2
	EN_PS	Digitiser	Logic Level / +5.x V = High, GND = Low	1
	MODE	Digitiser	Logic Level / +5.x V = High, GND = Low	1
	Clk	Digitiser	Digital Signal / +5.x V = High, GND = Low	1
	DIN	Digitiser	Digital Signal / +5.x V = High, GND = Low	1
	/CS	Digitiser	Digital Signal / +5.x V = High, GND = Low	1
	Pulser In	Digitiser	Digital Signal / +5.x V = High, GND = Low	1
Power	+5.x V	Preamp	Supply for Logic Isolators except for Inh_C	7
	-5.x V	Preamp	Supply for Inh_C Isolator	1
	GND	Preamp	Cable shields & Supply for all Logic Isolators	28

Note: all Digital Signals and Logic Levels across the interface will be isolated at the Digitizer using the Analog Devices part ADuM1100.

Signals in a segment cable

Type	*Name*	*Source*	*Type / format*	*Q.ty*	*Position*
Analog	Segment	Preamp	Analog signal / 1 differential pair	6	Pairs 1 to 6*
Digital	SHDN_A	Digitiser	Logic Level / +5.x V = High, GND = Low	1	Pair 7+
	SHDN_B	Digitiser	Logic Level / +5.x V = High, GND = Low	1	Pair 7-
	Reserved	-	For future expansion	1	Pair 9
	Inh_A	Preamp	Digital signal / +5.x V = High, GND = Low	1	Pair 10**
	Inh_B	Preamp	Digital Signal / +5.x V = High, GND = Low	1	Pair 11***
Power	+5.x V	Preamp	Supply for Logic Isolators	1	Pair 8+
	GND	Preamp	Ground	1	Pair 8-
	GND	Preamp	Cable shields & Supply for Logic Isolators	4	pins 1,13,14,26

*Pair i+ is positive swing, pair i- is negative swing, with i=1, 2,...,6
**Pair 10- swings dynamically, pair 10+ is tied to common ground, i.e. to pins 1,13,14,26
***Pair 11+ swings dynamically, pair 11- is tied to common ground, i.e. to pins 1,13,14,26

One cable serves 6 segment preamplifiers (or 2 triple preamplifiers: e.g. preampA tied to segments 1 2 3, and preampB tied to segments 4 5 6).

Signal description

Segment
Differential Analog signal pair. -2.5 V to +2.5 V.

Core
Differential Analog signal pair. -2.5 V to +2.5 V.

Inh_A, and Inh_B
Active High
(set dynamically by preamplifiers A and B)
This is a digital signal (TTL) that will be pulled up if any of the preamplifiers in triples A and B is undergoing a fast reset. It will be pulled down as soon as the fast reset is over. This can be used to inhibit acquisition of false events due to a reset transient. A source termination on the preamp side will be used. Please put no termination resistor on the receiver side to avoid half splitting of the TTL signal. The analog levels for InhibitAB are detector GND (low) and +5.x V (high) (coming with the cable). So please use such Power Supply voltages on the isolator on the cable side. Use instead the receiver 0V 5V PS levels on the receiver side.

Inh_C
Active High (set dynamically by core preamplifier)
This is a digital signal (TTL) that will be pulled up if the core amplifier is undergoing a fast reset. It will be pulled down as soon as the fast reset is over.
This can be used to inhibit acquisition of false events caused by a reset transient. A source termination on the preamp side will be used. Please put no termination resistor on the receiver side to avoid half splitting of the TTL signal.
The Analog levels for InhibitC are -5.x V (low) (coming with the cable) and detector GND (high). So please use such Power Supply voltages on the isolator on the cable side. Use instead the receiver 0V 5V PS levels on the receiver side. The isolator in this case works also as a voltage translator.
The width of this signal can be used to estimate the amplitude of the energetic event that caused the saturation. Hence this signal should be tied to a precise counter (with the highest possible clock frequency).

Signals in the core+pulser cable

Type	*Name*	*Source*	*Type / format*	*Q.ty*	*Position*
Analog	Core	Preamp	Analog signal / 1 differential pair	1	Pair 1*
Digital	AT10	Digitiser	Logic Level / +5.x V = High, GND = Low	1	Pair 2+
	AT10	Digitiser	Logic Level / +5.x V = High, GND = Low	1	Pair 2-
	AT20	Digitiser	Logic Level / +5.x V = High, GND = Low	1	Pair 3+
	AT20	Digitiser	Logic Level / +5.x V = High, GND = Low	1	Pair 3-
	Clk	Digitiser	Digital Signal / +5.x V = High, GND = Low	1	Pair 4+
	MODE	Digitiser	Logic Level / +5.x V = High, GND = Low	1	Pair 4-
	DIN	Digitiser	Digital Signal / +5.x V = High, GND = Low	1	Pair 5**
	/CS	Digitiser	Digital Signal / +5.x V = High, GND = Low	1	Pair 6**
	SHDN_C	Digitiser	Logic Level / GND= High, -5.x V = Low	1	Pair 7+
	EN_PS	Digitiser	Logic Level / +5.x V = High, GND = Low	1	Pair 7-
	Pulser In	Digitiser	Digital Signal / +5.x V = High, GND = Low	1	Pair 8**
	Reserved	-	For future expansion	1	Pair 9
	Inh_C	Preamp	Digital Signal / GND = High, -5.x V = Low	1	Pair 10**
Power	+5.x V	Preamp	Supply for Logic Isolators	1	Pair 11+
	-5.x V	Preamp	Supply for SHDN_C Isolator	1	Pair 11-
	GND	Preamp	Cable shields & Supply for Logic Isolators	4	Pins 1,13,14,26

*Pair 1+ is positive swing, pair 1- is negative swing
**Pair i+ swings dynamically, pair i- is tied to common ground, i.e. to pins 1,13,14,26

One cable serves the core preamplifiers and the built-in pulser

SHDN_A, and SHDN_B
Active High Logic Level.
Switches off fast reset mechanism in triple preamplifiers A and B.
If this is pulled up the fast reset mechanism will be permanently switched off. If this is pulled down the fast reset mechanism will automatically work in each of the six preamplifiers when needed.
This should not be handled dynamically by the Digitiser. The analog levels for SHDN_A, SHDN_B are detector GND (low) and +5.x V (high) (coming with the cable). So please use such Power Supply voltages on the isolator on the cable side. Use instead the receiver 0V 5V PS levels on the digitiser side.

SHDN_C
Active High Logic Level.
Switches off fast reset mechanism in the Core preamplifier.
If this is pulled up the fast reset mechanism will be permanently switched off. If this is pulled down the fast reset mechanism will automatically work when needed.
This should not be handled dynamically by the Digitizer. The analog levels for SHDN_C are -5.x V (low) (coming with the cable) and detector GND (high). So please use such Power Supply voltages on the isolator on the cable side. Use instead the receiver 0V 5V PS levels on the digitiser side.

AT10
Active High Logic Level.
Attenuates pulser amplitude by 10dB.
The analog levels for AT10 are detector GND (low) and +5.x V (high) (coming with the cable). So please use such Power Supply voltages on the isolator on the cable side. Use instead the receiver 0V 5V PS levels on the digitiser side.

AT20
Active High Logic Level.
Attenuates pulser amplitude by 20dB.
The analog levels for AT10 are detector GND (low) and +5.x V (high) (coming with the cable). So please use such Power Supply voltages on the isolator on the cable side. Use instead the receiver 0V 5V PS levels on the digitiser side.

EN_PS

Active High Logic Level.
Switches on the Power Supply of built-in Pulser. If this is pulled up the built-in pulser circuitry is biased. If this is pulled down the Pulser Power Supply is set to 0V. This permits to save power when the Pulser is not used.
This should not be handled dynamically by the Digitiser. The analog levels for EN_PS are detector GND (low) and +5.x V (high) (coming with the cable). So please use such Power Supply voltages on the isolator on the cable side. Use instead the receiver 0V 5V PS levels on the digitiser side.

MODE

Logic Level.
Selects the shape of built-in Pulser. If this is pulled up the built-in pulser provides a positive exponential decay. If this is pulled down the built-in pulser provides a square wave.
The analog levels for EN_PS are detector GND (low) and +5.x V (high) (coming with the cable). So please use such Power Supply voltages on the isolator on the cable side. Use instead the receiver 0V 5V PS levels on the digitiser side.

Clk, DIN, /CS

Digital Signals (Clock, Data In, Chip Select) used to set the pulser fine gain.
These signals implement a simple, low-frequency, 3-wire interface through which the 16-bit input of DAC AD5542 from Analog Devices is supplied, which sets the fine gain of the pulser. The maximum clock frequency is 25MHz but it is advisable to use a substantially lower frequency (e.g. 1 MHz). The protocol of this interface can be found on the data sheet of AD5542. It is advisable to use only large numeric values (in the range from 50 to 100% of the full scale) for fine-gain corrections. Otherwise the long-term stability of the pulser could worsen significantly. Use instead AT10 and/or AT20 for coarse-gain setting. The analog levels for Clk, DIN, /CS are detector GND (low) and +5.x V (high) (coming with the cable). So please use such Power Supply voltages on the isolator on the cable side. Use instead the receiver 0V 5V PS levels on the digitiser side.

Pulser In

Active High Digital Signal.
This is a digital signal (TTL) that will be pulled up to generate a positive transition of the pulser and down to reset it, or generate a negative transition edge (depending on setting of MODE).
The analog levels for Pulser In are detector GND (low) and +5.x V (high) (coming with the cable). So please use such Power Supply voltages on the isolator on the cable side. Use instead the receiver 0V 5V PS levels on the receiver side.

+5.x V

Power supply provided by the preamplifiers for the Logic isolators except for the core Inhibit. The actual value will range between +5 and +6 V.

-5.x V

Power supply provided by the core preamplifier for the Logic isolator for the core Inhibit. The actual value will range between -5 and -6 V.

GND

All grounds, including the internal shieldings are to be connected at the preamplifier side and should not be connected to the digitiser's ground.

FUTURE UPGRADE

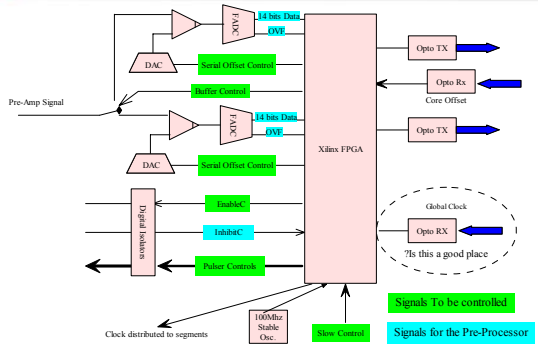
In a future version the spare twisted pairs could be used as a serial transmission link (lvds) from the receiver to the triples/core preamplifiers (using a microcontroller to receive and distribute the signals).
This link could be used to set the P/Z fine adjustment of the triples/core preamplifiers.

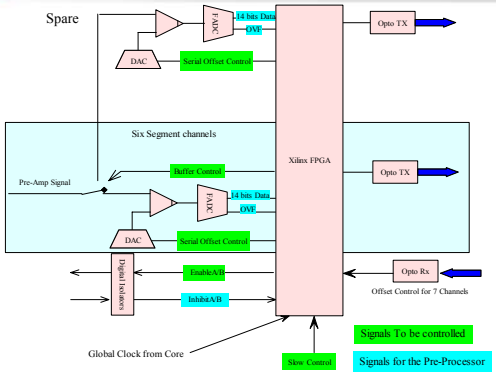
IMPORTANT NOTES:

1) All detector GND's carried to the receiver along the cable must be connected at the receiver end to the cable shields found on pins 1, 13, 14 and 26 of the MDR connector.
2) All digital signals transmitted from the receiver to the detector should enter the cable through a source series termination resistor of 75 Ohms. Instead, no termination resistor should be put at the detector side to avoid half splitting of the TTL signal (with the possible exception of signal "Pulser In", which could be received through a 75 Ohm resistor and a Schmitt-Trigger comparator).
3) All Logic Levels (see column "Format" in Complete Signal List) should be bypassed to ground with 1uF capacitors either at the transmitter or at the receiver sides.

OverView of the Digitiser.

- 37 operational + 7 spare channels
  - Remotely controlled input Offset
  - Input routing to spare channel
  - 100Mhz FADC
  - Xilinx FPGA Serialise, and output as a 2Gb/s data stream.
  - VCSEL Laser transmitter for 6 segments and 1 spare.
  - Laser transmitter for core and 1 spare.
- Control interface to the Detector
- Offset control from Pre-Processor. 1 per 6 +1 spare

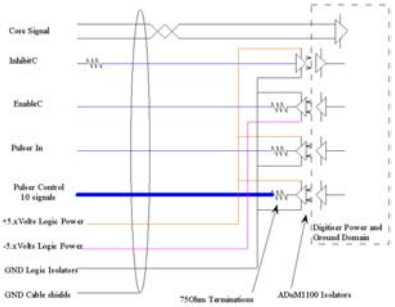




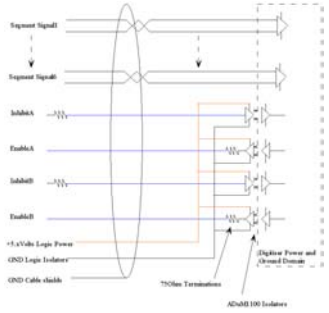
Digitiser Interfaces

1. Detector
   - 37 Differential Analog inputs.
   - 36 isolated logic signals.
2. Pre-Processor
   - 53 Fibre optic links
3. Local Monitoring

To Detector Pre-amplifiers. First the Core.

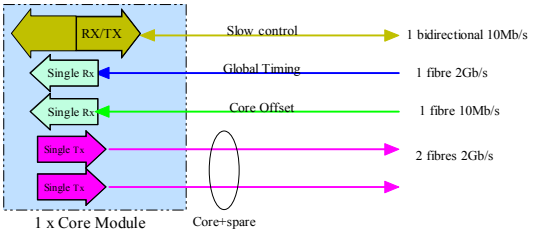


Second the Segments Six per cable



To Pre-Processor

Core + spare and Global connections



Core + spare and Global connections

- Core & spare:
  - 16 bits data at 100Mhz serialised,and 8B/10B encoded. 16bits => 20bits => 2Gb/s
  - Clock is based on Global Clock.
- Global Timing:
  - Transfers the Global Clock.
  - Transfers timing related signals (synchronise??)
- Core Offset control
  - Sends data to the offset DAC at the Core input.
- Slow Control.
  - Transfers control and status between AGATA control and digitiser. (Via Pre-processor)

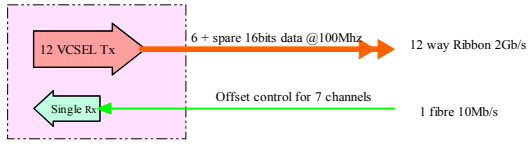
Core + spare and Global connections

Questions:

- Core + spare : Do we need two optical links?
- Global Timing: Is variable Latency a problem in this link.
- Offset Control : What speed is required. Taking account of latency through digitiser. 1/10/100uS.
- Slow Control :Bidirectional, but what speed is needed ?

To Pre-Processor

Six x 6 Segment + spare connections



Segment + spare and Offset connections

- Segments & spare:
  - Same as the Core.
  - 16 bits data at 100Mhz serialised,and 8B/10B encoded. 16bits => 20bits => 2Gb/s
  - Clock is based on Global Clock.
- Channel Offset control
  - Sends data to the offset DACs at the channel inputs.

Segment + spare and Offset connections

Questions:

- Segment + spare : Do we agree on 7 fiber links?
  - Is variable Latency a problem in these links.
- Offset Control : What speed is required. Taking account of latency through digitiser. 1/10/100uS?

Local use only : monitors

- USB : Controls module during test and diagnostics
- Inspection Lines
  - Analog - Input signal
    - FADC output via AD9765 DAC
  - Digital - Fast NIM

Optical Link trial module. What it contains

- Virtex II Pro XC2VP20 (8 Rocket I/O)
- 2 sockets for TX + 2 sockets for RX
- 2 Rocket I/O to SMA for copper link to demo board
- Clock from Metronome, or local Oscillator
- VME interface for power, control and statistics
- Static RAM for test pattern storage.
- Logic Monitor outputs.
- Analog inspection DAC

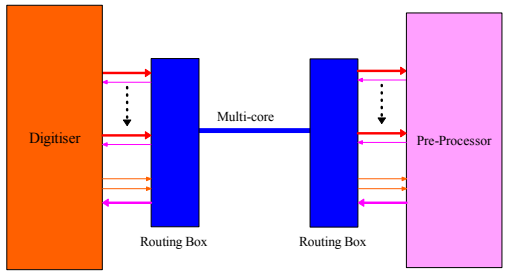
Optical Link trial module. What is tested.

- Link reliability, bit error rate (BER). Long term statistics
- Link Initialisation techniques.
- PCB layout with VCSEL TX on back of board
- Perhaps try link with faster bit rate , clock friendly bit patterns, CRC error check. Use metronome clock interface to check latency, and time drift.

PCB Daresbury, expected July 2004.

Xilinx VHDL IReS.

Using a single Multi-core for the 100 metre connection.



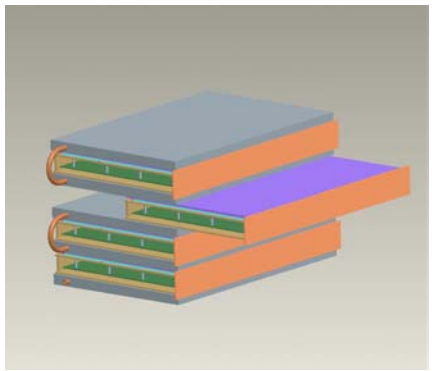
Some Questions:

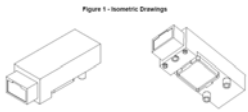
Detector:

1. Definition of Pulser controls.

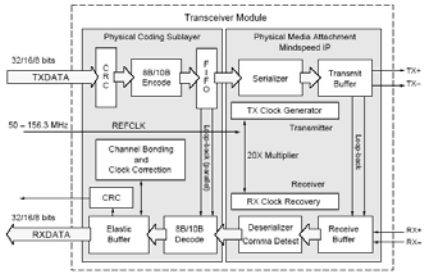
Pre-Processor:

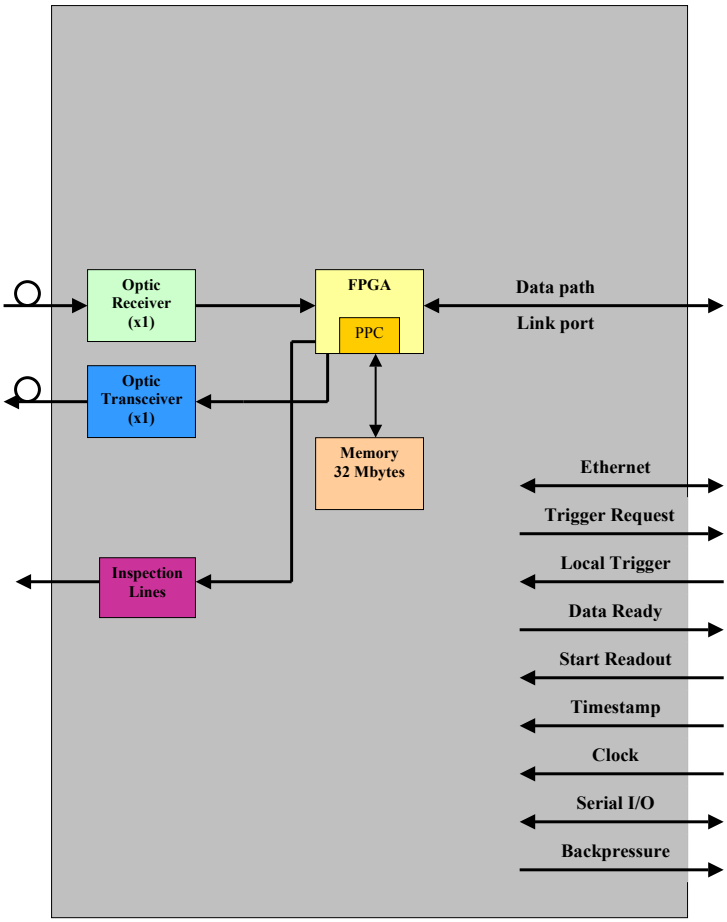
1. How can we solve the latency problem ?
2. Do we need a core + spare optical link ?
3. Do we need a protocol to reduce the BER ?
4. What is the best cabling method.
5. What Protocol for the Offset, and Slow links



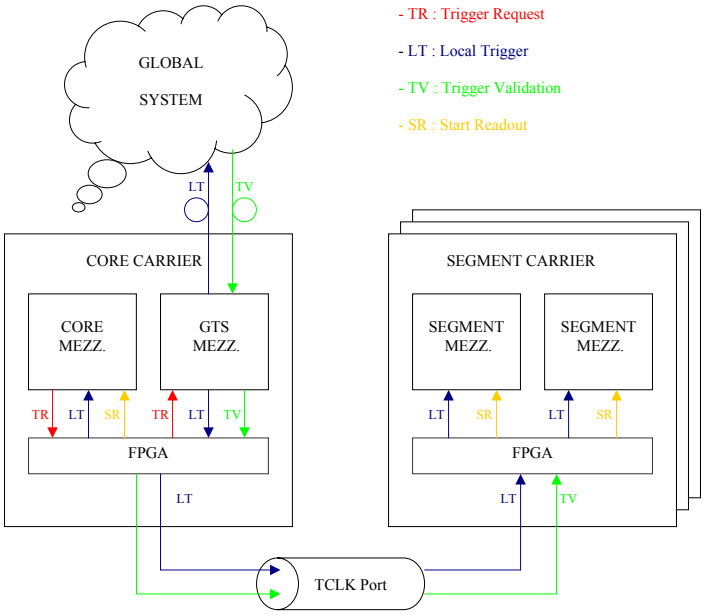


Xilinx Rocket I/O Embedded Core

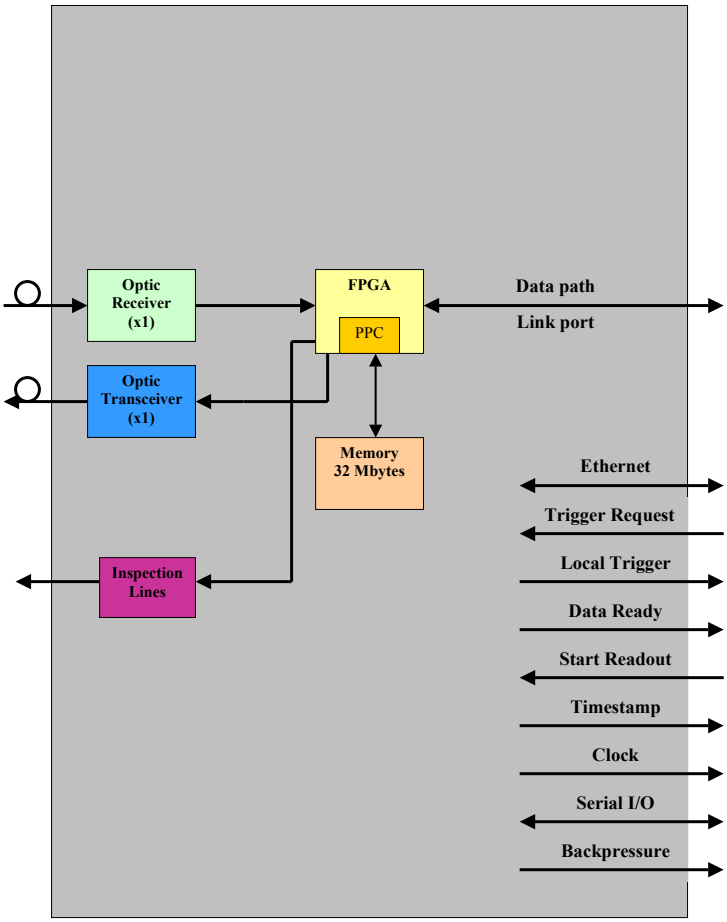




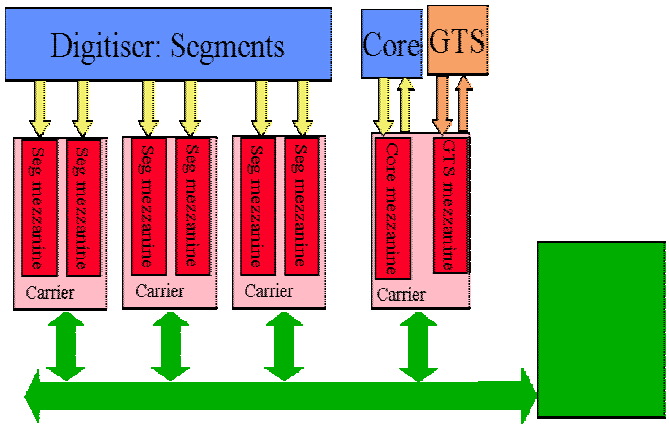
CORE MEZZANINE



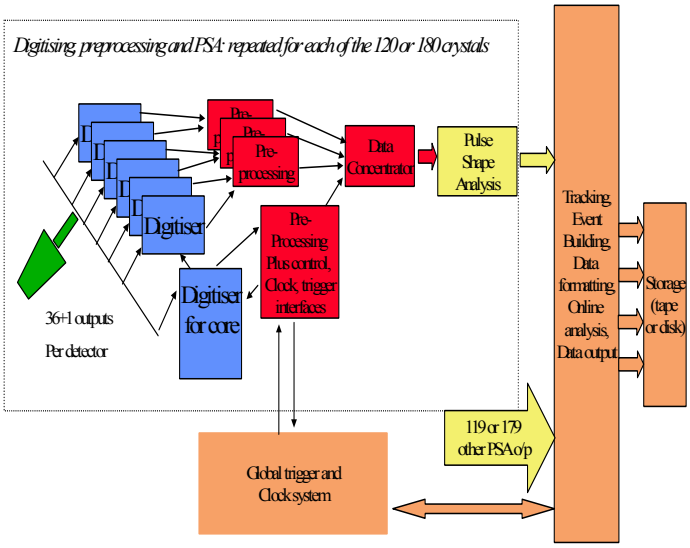
TRIGGER SEQUENCE



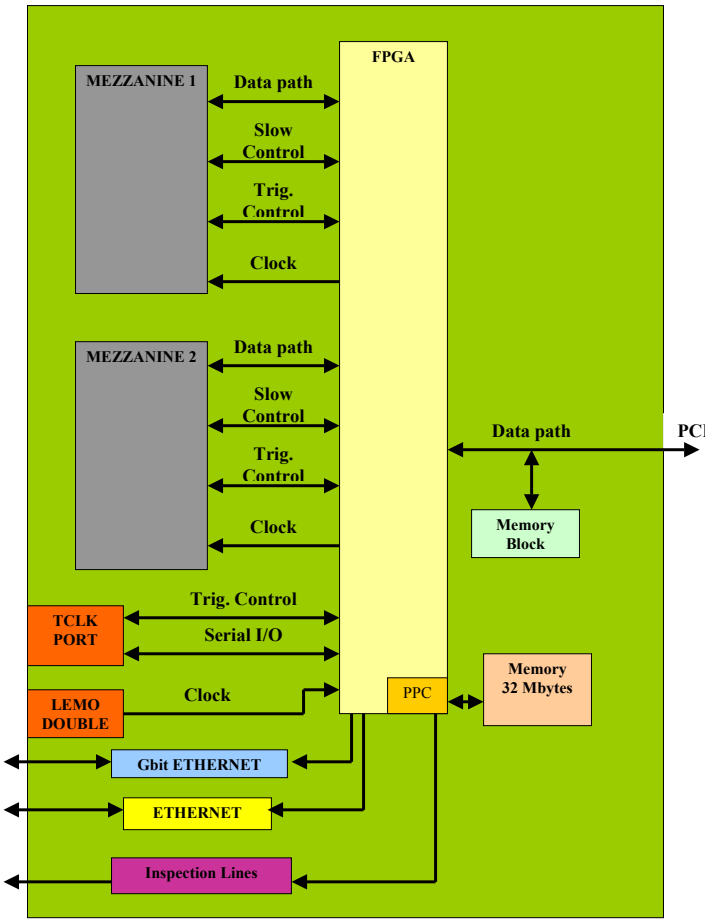
GTS MEZZANINE



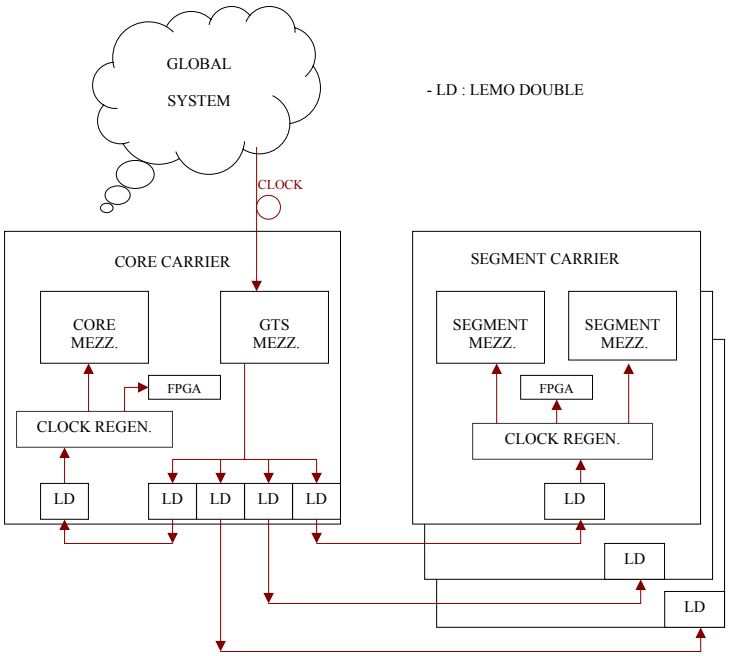
IMPLEMENTATION



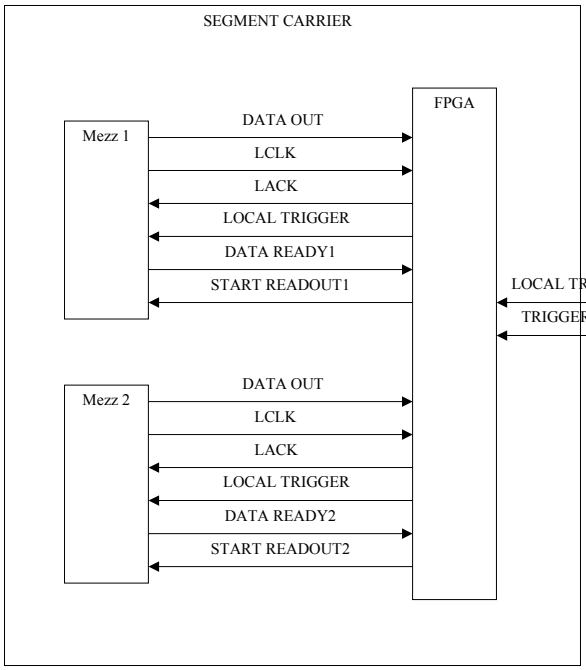
PRE-PROCESSING SYSTEM



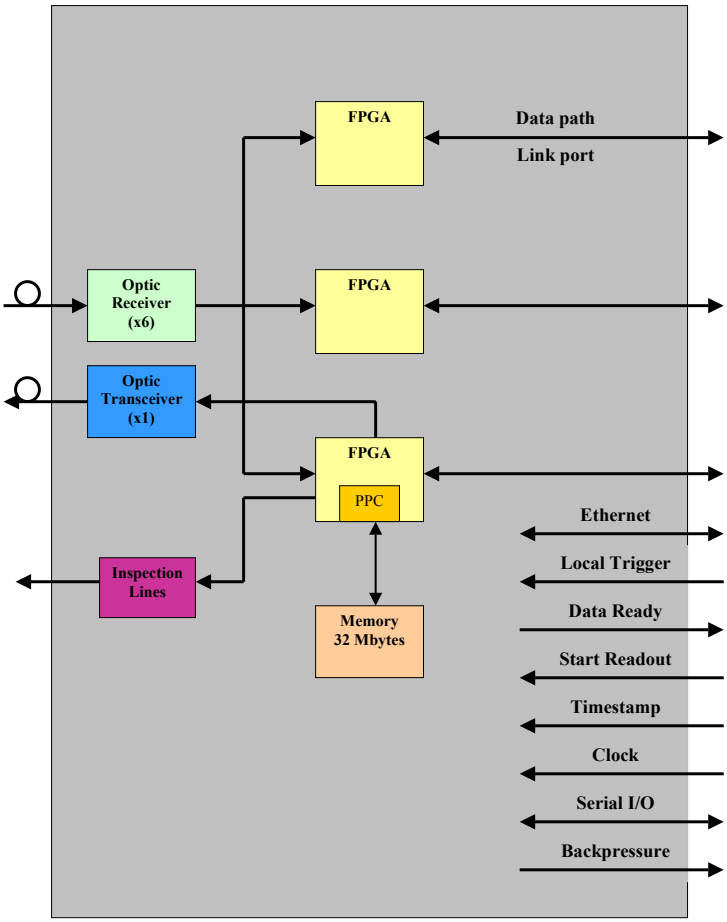
CARRIER



CLOCK DISTRIBUTION



READOUT

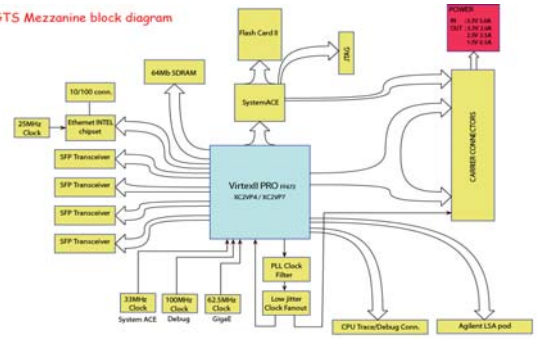


SEGMENT MEZZANINE

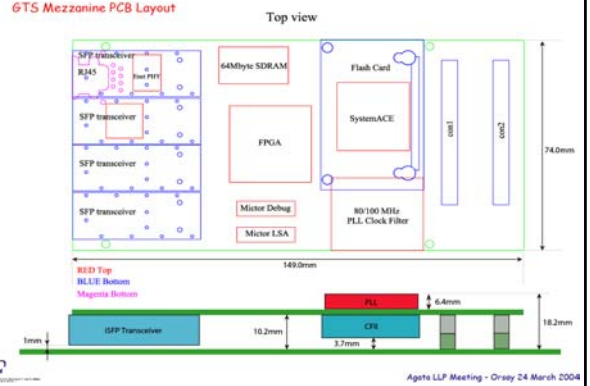
GTS Mezzanine card overview and status

M. Bellato D. Bortolato R. Isocrate
INFN Sezione di Padova

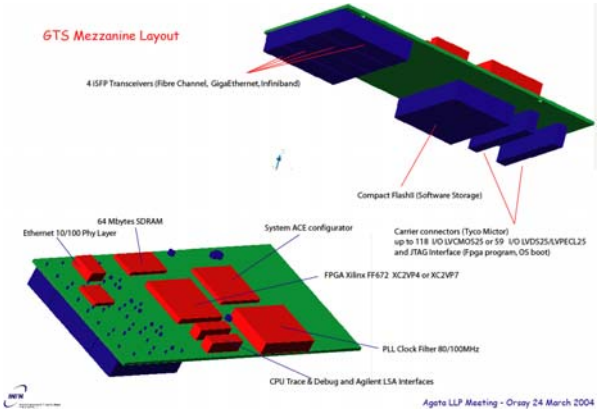
GTS Mezzanine block diagram



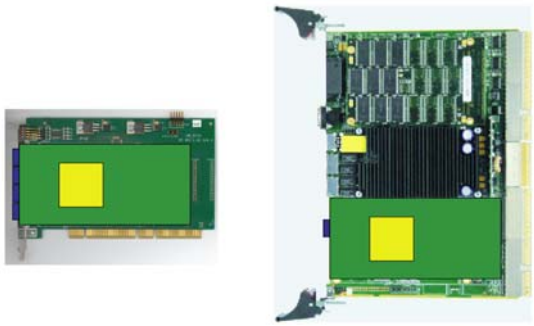
GTS Mezzanine PCB Layout



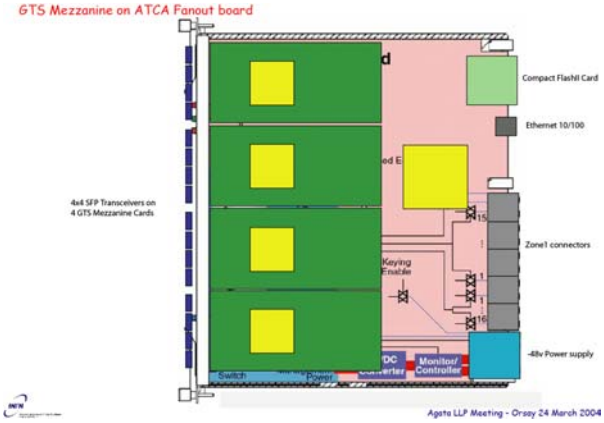
GTS Mezzanine Layout



GTS Mezzanine on GIII PCI Card (Test and Debug) and on a cPCI LLP Carrier



GTS Mezzanine on ATCA Fanout board



GTS Mezzanine Inputs Outputs 1

Front Panel :

- up to 4 iSFP Transceiver
- 1 10/100 Ethernet RJ45 connector (instead of 2 transceiver)
- Led for status indications

On board (for debug):

- SMA clock injector
- Mictor connectors for trace and debug of CPU and for Agilent LSA

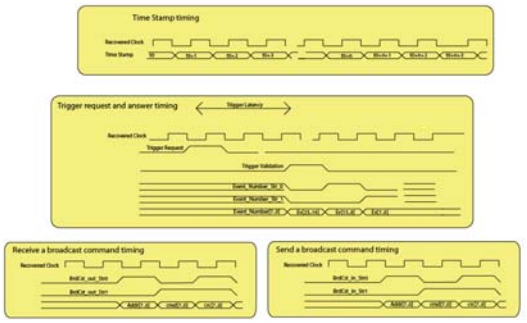
On carrier connectors : Two 114 pin Mictors that include :

- 1 differential clean clock output (LPECL33 80/100MHz Jitter <60ps rms)
- 13 Fpga I/O (LVTTL25)
- 10 Dedicated pins for JTAG interface and SysAce management
- 128 Fpga I/O (LVTTL25) or 64 FPGA Differential I/O (LVPECL25 , LVDS25)

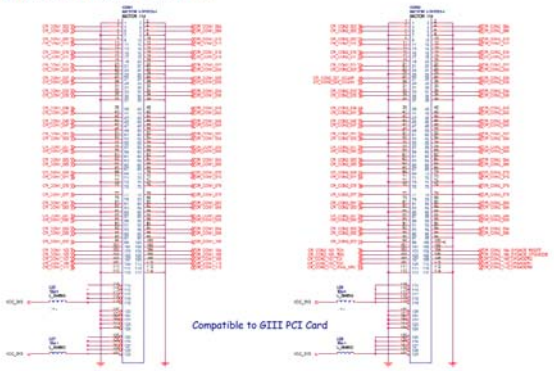
Power Supply :

- 3,3 volt 5A max 16.5Watt (16pin @ 3,3v , 8pin @ GND)

GTS Mezzanine data exange protocol



GTS Mezzanine to Carrier Connectors



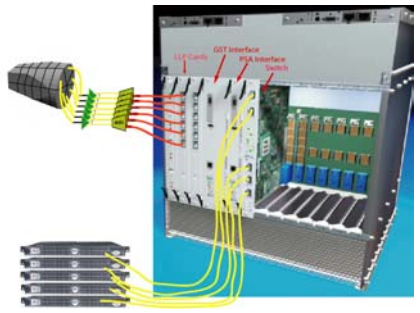
GTS Mezzanine Inputs Outputs 2

Signal Name	Pins	Level	Comments
Time_Stamp [47..0]	48	LVTTL25	Time stamp
Clean_Clock	2	LVPECL33	Clean Clock 80/100MHz Jitter cycle-cycle <60ps rms
Trigger_Request	1	LVTTL25	
Trigger_Validation	1	LVTTL25	
BackPressure	1	LVTTL25	Low latency backpressure command
Event_number [7..0]	8	LVTTL25	Event TAG associate with validation
Event_Number_Bit_0	1	LVTTL25	Strobe (see Timing diagram)
Event_Number_Bit_1	1	LVTTL25	
Brdcst_in [7..0]	8	LVTTL25	Commands TO GTS
Brdcst_in_Bit_0	1	LVTTL25	Strobe (see Timing diagram)
Brdcst_in_Bit_1	1	LVTTL25	
Brdcst_out [7..0]	8	LVTTL25	Commands FROM GTS
Brdcst_out_Bit_0	1	LVTTL25	Strobe (see Timing diagram)
Brdcst_out_Bit_1	1	LVTTL25	
JTAG_Port	4	LVTTL25	JTAG port for the FPGA
JTAG_Sel	1	LVTTL25	Only for debug (on carrier always 0)
Sysfce_Reset	1	LVTTL25	
Sysfce_Clkmode	1	LVTTL25	Only for debug (on carrier always 0)
Sysfce_Clkdiv0 [2..0]	3	LVTTL25	Boot card software selection
Ethernet_I/O_0		TBD!!!	
Ethernet_I/O_1		TBD!!!	WARNING ! High power analog signals
Ethernet_I/O_2		TBD!!!	cannot be routed on VME
Ethernet_I/O_3		TBD!!!	
LVTTBUS	32	LVTTL25	Ethernet Port chipset
Master_Reset_0	1	LVTTL25	Four type of synchronous reset are defined
Master_Reset_1	1	LVTTL25	
System_Error_0	1	LVTTL25	Four type of error condition are defined
System_Error_1	1	LVTTL25	
Power_3v3	8		
Power_Gnd	8		
Power_GNd1	8		
Total # of pins		155/453	

AGATA Front-End Simulation Environment

M. Bellato
INFN-Padova

What are we speaking about



Why do we need it ?

- Agata front-end is complex !
  - The readout must be fully pipelined to cope with high trigger rates
  - Need to gain insight of LLP - PSA Interface- GTS interactions
- Need to investigate behaviour with sampled data
  - We should assess correctness of event building before PSA, at any rate.
  - And dimension fifos and memory depths according to expected trigger rates and service times
- Need a reference model of the front-end hardware to be agreed by different working groups
  - A way to reduce misunderstandings
- Need a way to evaluate quickly the impact of changes
  - Engineering changes are unavoidable, but sometimes underestimated

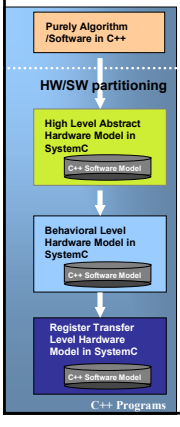
How do we get it ?

- System Level simulation
  - Complete
  - Abstraction is high
  - Lacks timing information
    - Not good for modeling front-end hardware
- Gate Level simulation
  - Accurate but abstraction is poor
    - The description of the model is huge and unmanageable
  - Timing information is complete
  - Simulation is unacceptably slow for complex models
- Cycle accurate simulation
  - Abstraction can be mixed
  - Tming information is accurate at the clock cycle
  - Model is compact
  - Simulation is fast

Tools

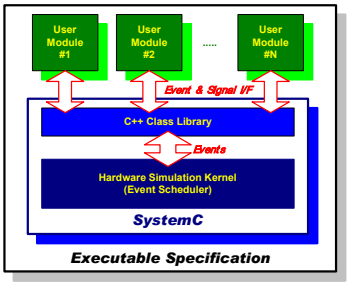
- SystemC chosen as simulation environment
  - http://www.systemc.org
  - Open environment for system level design
  - Can mix structural, RTL and behavioural hardware descriptions.
  - Open source, is a C++ library
  - Dumps traces in VCD or WIF formats
  - Can be debugged with any C debugger
  - Is synthesizable
  - Can be simulated from within major EDA tools (Mentor Modelsim, Cadence NCsim, ...)

Highlights

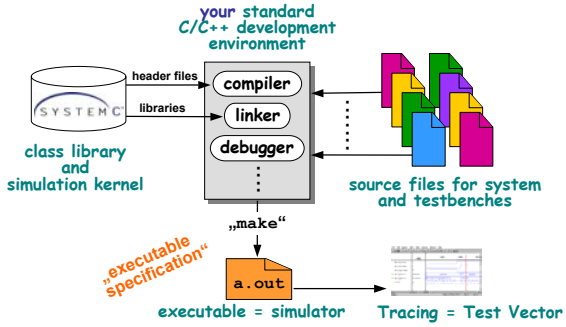


Interface in a C++ environment
- Modules
  - **Container class includes hierarchical Entity and Processes**
- Processes
  - **Describe functionality, Event sensitivity**
- Ports
  - **Single-directional(in, out), Bi-directional(inout) mode**
- Signals
  - **Resolved, Unresolved signals**
- Rich set of port and signal types
- Rich set of data types
  - **All C/C++ types, 32/64-bit signed/unsigned, fixed-points, MVL, user defined**
- Clocks
  - **Special signal, Timekeeper of simulation and Multiple clocks, with arbitrary phase relationship**
- Cycle-based simulation
  - **High-Speed Cycle-Based simulation kernel**
- Multiple abstraction levels
  - **Untimed from high-level functional model to detailed clock cycle accuracy RTL model**
- Communication Protocols
- Debugging Supports
  - **Run-Time error check**
- Waveform Tracing
  - **Supports VCD, WIF, ISBD**

SystemC & Modules



Simulation Flow



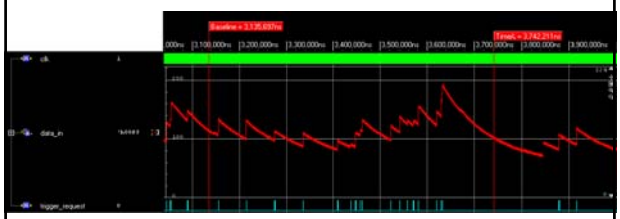
Example – SCC/1

```
SC_MODULE(SCC) {
      sc_in_clk clk;                    // The clock
      sc_in<bool> reset;                // Reset
      sc_out<bool> trigger_request;     // to GTS
      sc_out<bool> local_trigger;       // to mezzanines
      sc_in<sc_uint<ADC_BITS> > data_in; // sample input
      sc_in<sc_uint<12> > threshold;    // threshold
      sc_in<sc_uint<10> > hold_time;    // trigger duration

      // local signals
      sc_uint<ADC_BITS> pipe[NSAMPLE];
      sc_signal <sc_uint<ADC_BITS> > result;
      sc_signal <sc_int<ADC_BITS> > average_out;

      ....

      void count();    // core counting process
      void average (); // average process
      void compare() ; // compare to a threshold

      SC_CTOR(SCC) {
         SC_METHOD(count);
         sensitive_pos << clk << reset;
         SC_METHOD(average);
         sensitive_pos << clk << reset;
         SC_METHOD(compare);
         sensitive_pos << clk << reset;
      }};
```

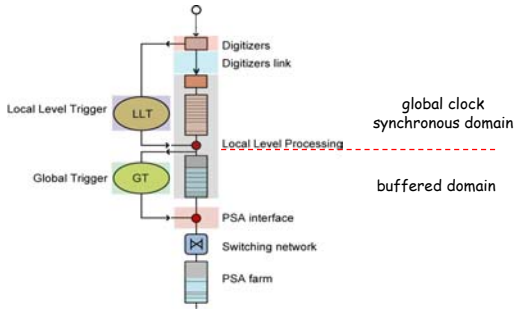
Example – SCC/2

```
//
// SCC accumulator - count the number of samples
// in the pipe that are greater than the midsample
// and smaller than it
//

void SCC::count() {
int i;
   if (reset) { // Reset operations
         for (i=0; i< NSAMPLE; i++) {
               /* synopsys unroll */
               pipe[i] = 0;
               }
   } else {
         tmp = data_in.read();
         pacc=0; nacc=0;
         for (i=0;i<NSAMPLE-1;i++){      // shift
               /* synopsys unroll */
               pipe[i] = pipe[i+1];
               }
         pipe[NSAMPLE-1] = tmp;
         for (i=0; i< MIDSAMPLE; i++) {  // forward count
               /* synopsys unroll */
               if(pipe[i] < pipe[MIDSAMPLE]) nacc++;
               }
         for (i=MIDSAMPLE+1; i< NSAMPLE; i++) {  // backward count
               /* synopsys unroll */
               if(pipe[i] > pipe[MIDSAMPLE]) pacc++;
               }
         result=pacc+nacc;
   } };
```

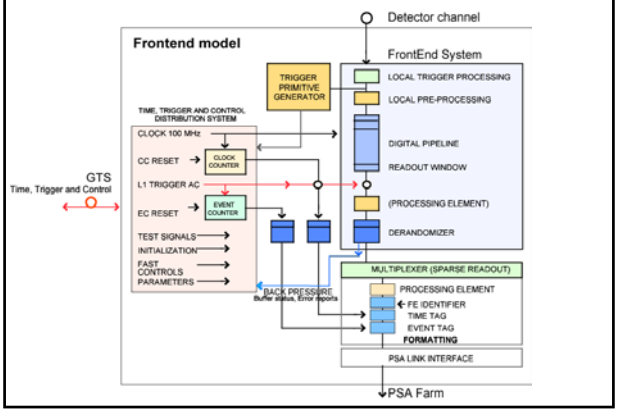
SCC Simulation Result



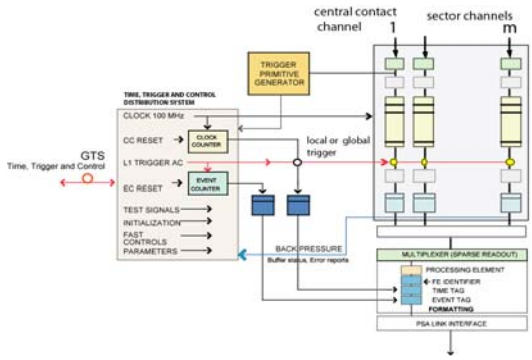
Front-end Data Flow



The Channel model



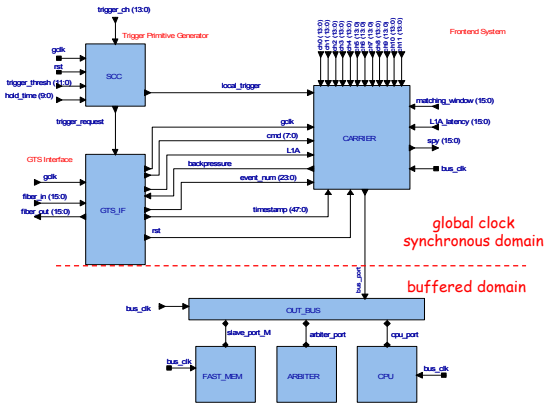
The Crystal model



The simulation hierarchy/1

- agata_fe/
  - bus/ — bus model + cpu
  - carrier/ — carrier = 2 mezzanines+readout
  - channel/ — channel – bottom of hierarchy
  - data/ — sampled waveforms
  - fes/ — fes = GTS + 1 carrier
  - gts/ — GTS master + fanout + GTS interface
    - gts_master/
    - fanin_fanout/
    - gts_if/
  - makefile — global Makefile
  - mezzanine/ — mezzanine = 6 channels
  - mwd_fixed/ — fixed point mwd
  - mwd_float/ — floating point mwd
  - scc/ — local trigger algorithm
  - util/ — fifo, memory, random models, etc

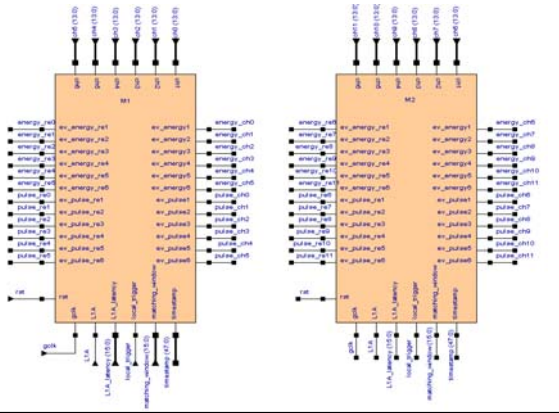
FES Block Diagram



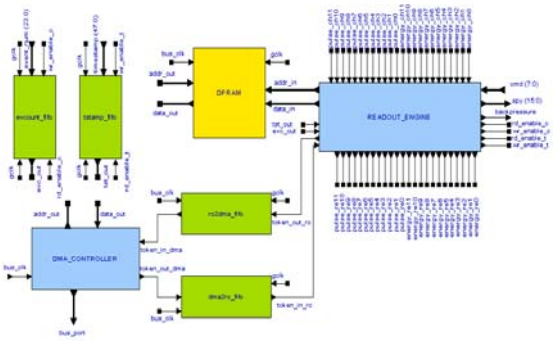
Fes.h

```
class FES: public sc_module
{
public:
  SC_HAS_PROCESS(FES);

  // ports
  sc_in_clk gclk;                          // GTS clock
  sc_in_clk bus_clk;                       // bus clock
  sc_in<sc_uint<16> > fiber_in;            // from GTS
  sc_out<sc_uint<16> > fiber_out;          // to GTS
  sc_in<sc_int<ADC_BITS> > ch[12];         // 12 data channels
  sc_in<sc_uint<ADC_BITS> > trigger_ch;    // 1 trigger data channel
  sc_in<sc_uint<12> > trigger_thresh;      // threshold for SCC
  sc_in<bool> L1A;                         // validation from GTS
  sc_in<bool> backpressure;                // backpressure to GTS
  sc_in<sc_uint<16> > event_count;         // event counter
  sc_in<sc_uint<16> > matching_window;     // trigger matching window
  sc_in<sc_uint<16> > L1A_latency;         // latency value for L1A
  sc_in<sc_uint<10> > hold_time;           // hold time for trigger request
  sc_out<sc_uint<16> > spy;                // a spy channel
  sc_out<bool> trigger_request;            // trigger request to GTS
  sc_in<bool> local_rst;                   // local reset

private:
  fast_mem *M9;                            // a memory bank
  arbiter *arb;                            // a bus arbiter
  bus *bus1;                               // a bus object
  CARRIER *c1;                             // the carrier
  CPU *cpu;                                // a cpu
  SCC *scc;                                // a local trigger facility
  GTS_IF *gts_if;                          // the GTS mezzanine
 ....
```

Carrier Block Diagram/1



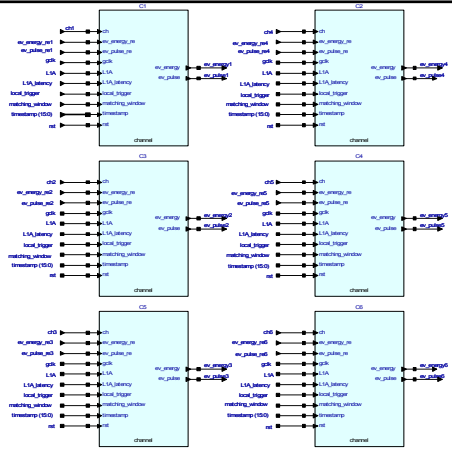
Carrier Block Diagram/2



Carrier.h

```
class CARRIER: public slave_if, public sc_module
{
public:
  SC_HAS_PROCESS(CARRIER);

  // ports
  sc_in_clk gclk;
  sc_in_clk bus_clk;
  sc_in<bool> rst;
  sc_in<bool> L1A;
  sc_in<bool> local_trigger;
  sc_in<sc_uint<48> > timestamp;
  sc_in<sc_uint<16> > event_count;
  sc_in<sc_uint<16> > matching_window;
  sc_in<sc_uint<16> > L1A_latency;
  sc_in<sc_int<DSIZE_P> > ch[12];
  sc_out<sc_int<32> > data;
  sc_out<sc_uint<16> > spy;
  sc_port<blocking_if, 1> bus_port;              // bus port

private:
  MEZZANINE *m1, *m2;                            // the mezzanines
  sc_uint<16> DPRAM[RO_BUFSIZE];                 // the dual port output buffer
  FIFO <sc_uint<48>, MAX_L1A_SERVICE > *tstamp_fifo; // FIFO for timestamps
  FIFO <sc_uint<16>, MAX_L1A_SERVICE > *evcount_fifo; // FIFO for event counter
  FIFO <sc_uint<1>, 2 > *ro2dma, *dma2ro;        // token exchange RO-DMA

  void readout_engine();
  void dma_controller();
  void tags_storage();
  void wait_loop();
  ......
```

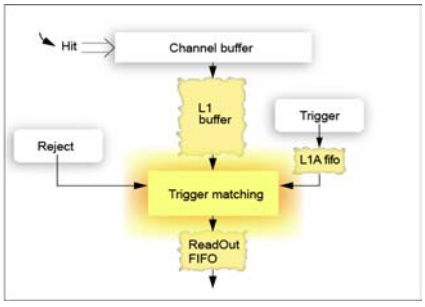
Mezzanine Block Diagram



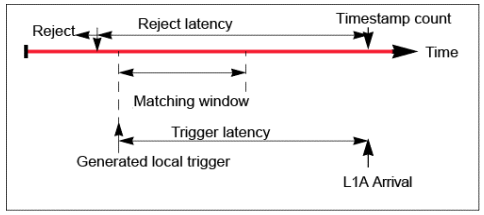
Mezzanine.h

```
SC_MODULE(MEZZANINE)
{
  .....
  sc_in<sc_int<DSIZE_P> > ch1, ch2, ch3, ch4, ch5, ch6;
  sc_out<sc_int<DSIZE_EV> > ev_pulse1,ev_pulse2,ev_pulse3;
  sc_out<sc_int<DSIZE_EV> > ev_pulse4,ev_pulse5,ev_pulse6;
  sc_out<sc_int<DSIZE_EV> > ev_energy1,ev_energy2,ev_energy3;
  sc_out<sc_int<DSIZE_EV> > ev_energy4,ev_energy5,ev_energy6;
  sc_in<bool> ev_pulse_re1, ev_pulse_re2, ev_pulse_re3;
  sc_in<bool> ev_pulse_re4, ev_pulse_re5, ev_pulse_re6;
  sc_in<bool> ev_energy_re1, ev_energy_re2, ev_energy_re3;
  sc_in<bool> ev_energy_re4, ev_energy_re5, ev_energy_re6
  sc_out<bool> ev_empty1, ev_empty2, ev_empty3, ev_empty4, ev_empty5, ev_empty6;
  sc_out<bool> en_empty1, en_empty2, en_empty3, en_empty4, en_empty5, en_empty6;

  CHANNEL *c1, *c2, *c3, *c4, *c5, *c6;

  SC_CTOR (MEZZANINE)
  {
    c1 = new CHANNEL ("c1");
    c2 = new CHANNEL ("c2");
    c3 = new CHANNEL ("c3");
    c4 = new CHANNEL ("c4");
    c5 = new CHANNEL ("c5");
    c6 = new CHANNEL ("c6");
    .....
```

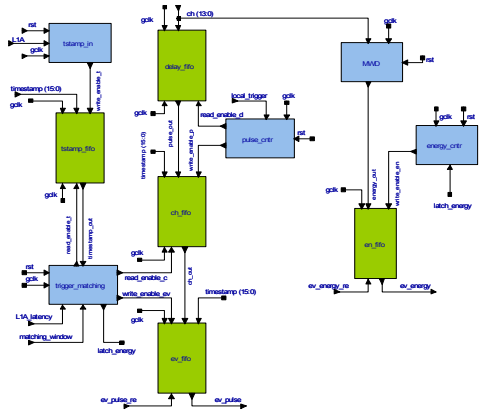

The Channel Architecture



Trigger Matching



Channel Block Diagram



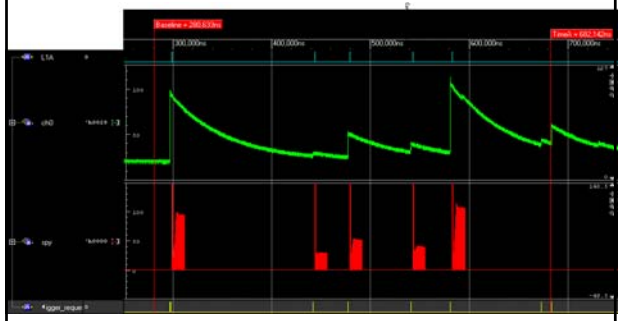
Channel.h

```
SC_MODULE(CHANNEL)
{
  sc_in_clk gclk;
  sc_in<bool> rst;
  sc_in<bool> L1A;
  sc_in<bool> local_trigger;
  sc_in<sc_uint<TSTAMP_SIZE> > timestamp_lsw;     // lower 16 bits of timestamp
  sc_in<sc_int<DSIZE_P> > ch;                     // sampled data in
  sc_in<bool> ev_pulse_re;                        // ev fifo read enable
  sc_in<bool> ev_energy_re;                       // en fifo read enable
  sc_out<bool> ev_empty;                          // empty fifos
  sc_out<bool> en_empty;                          // empty fifos
  sc_out<sc_int<DSIZE_EV> > ev_pulse;             // pulse out
  sc_out<sc_int<DSIZE_EV> > ev_energy;            // energy out
  sc_in<sc_uint<16> > matching_window;
  sc_in<sc_uint<16> > L1A_latency;

  FIFO <sc_uint<TSTAMP_SIZE>, MAX_L1A_SERVICE> * tstamp_fifo;
  FIFO <item_p, FIFOLEN_D> * delay_fifo;
  FIFO <sc_int<TSTAMP_SIZE+1>, FIFOLEN_P> * ch_fifo;
  FIFO <item_ev, FIFOLEN_EV> * ev1_fifo;
  FIFO <item_ev, FIFOLEN_EV> * en1_fifo;
  MWD * mwd_ch;

  void pulse_cntr();                              // pulse controller
  void ener_cntr();                               // energy controller
  void recording_machine();                       // storage controller
  void manage_rollover();                         // time rollover handler
  void trigger_matching();                        // trigger match
  .....
```


Channel.cc sample

```
void CHANNEL::trigger_matching() {
  ......
  case MATCH_S_START:
       trigger_matched = false;
       ev_discard = false;
      // extract timestamp and computes elapsed time
      // for the oldest event in the fifo
      if(ch_fifo->IsEmpty() == false) {
         // take first fifo val without removing
         event_time= ch_fifo->Read();
         if(event_time & (1<<TSTAMP_SIZE)) {
            // it is really a time stamp !
            event_time &= (1<<TSTAMP_SIZE)-1;
            elapsed = (timestamp_lsw.read()- event_time) % (1<<TSTAMP_SIZE);
            trigger_request_time =(L1A_time - L1A_latency.read()) % (1<<TSTAMP_SIZE);
            upper_limit=(trigger_request_time+matching_window.read())%(1<<TSTAMP_SIZE);
            manage_rollover();
            if(trigger_matched || ev_discard)    {
               dump_count=1;
               if(trigger_matched) {
                  ev1_fifo->Write(SOF);
                  latch_energy = true;
                  }
               next_state = MATCH_S_DUMP_START;
               }
            else next_state = CHECK_L1A_FIFO; //event is newer than L1A arrival time
            }
          else next_state = ERROR; //event is misaligned
          }
       else next_state = CHECK_L1A_FIFO; //event fifo is empty
       break;
```

Trigger Matching run example



Interesting Parameters

- Channel Fifo's occupancy vs trigger rate
  - Overflow probability
  - Fifo's depth
- Global dead time vs trigger rate
- Bus throughput
- Data misalignment vs trigger rate
- Backpressure rate
- Throttling system emulation

Status of simulation code

- Building blocks are ok
  - Except GTS trigger processor (only multiplicty trigger available) & GTS fanin-fanout
- Needs revision for uniform management of parameters
- Needs extensive testbench code for
  - Exercising all interactions between GTS and the frontends
  - Checking the event building before PSA
  - Find out most critical situations
- Needs long (> 1 sec) sampled traces (central contacts + 1 segment) to check functionality with real data

Further Development

- Up to us:
  - Use the model as a proof of concept
    - Must be agreed by everybody or adjusted
    - Leave as it is now, make measures and keep as reference
  - Extend the model to a faithful hardware description
    - Use it to validate and debug hardware
    - Use it online as a diagnostic tool
    - Every group must endorse the competent part and keep the model up-to-date with hardware

Conclusions

- System level simulation is effective because we gain insight of the problem
- The detail of description depends on simulation goal but can be changed and mixed
- SystemC works fine both for high and low level hardware description and is fast
- The channel model is the most crucial part of front-end and has impact on design choices
- GTS will extensively use this model for design investigation
- Do we extend it further ?