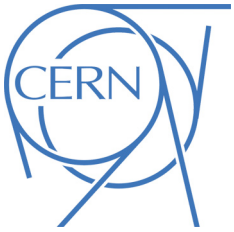
ATLAS NOTE

April 29, 2015



ATLAS LAr Calorimeter trigger electronics phase I upgrade: AMC Firmware Specifications

Nicolas Chevillot[b], Bernard Dinkespiler[a], Nicolas Dumont-Dayot[b], Yuji Enari[c], Rainer Hentges[d], Kenneth Johns[e], Isabelle Wingerter[b], Vladimir Zhulanov[f]

[a] *CPPM – CNRS – France*
[b] *LAPP – CNRS – France*
[c] *University of Tokyo – Japan*
[d] *University of Dresden – Germany*
[e] *University of Arizona – USA*
[f] *Budker INP and Novosibirsk State University – Russia*

Abstract

This document describes the specifications and the structure of the firmware embedded in the Arria™10 FPGA, referenced 10AX115R4F40I4SGES (ARRIA 10 family ALTERA™), located on the advanced mezzanine card, AMC-A10, of the LAr Digital Processing Board. Interfaces between the various blocks of the firmware are defined.

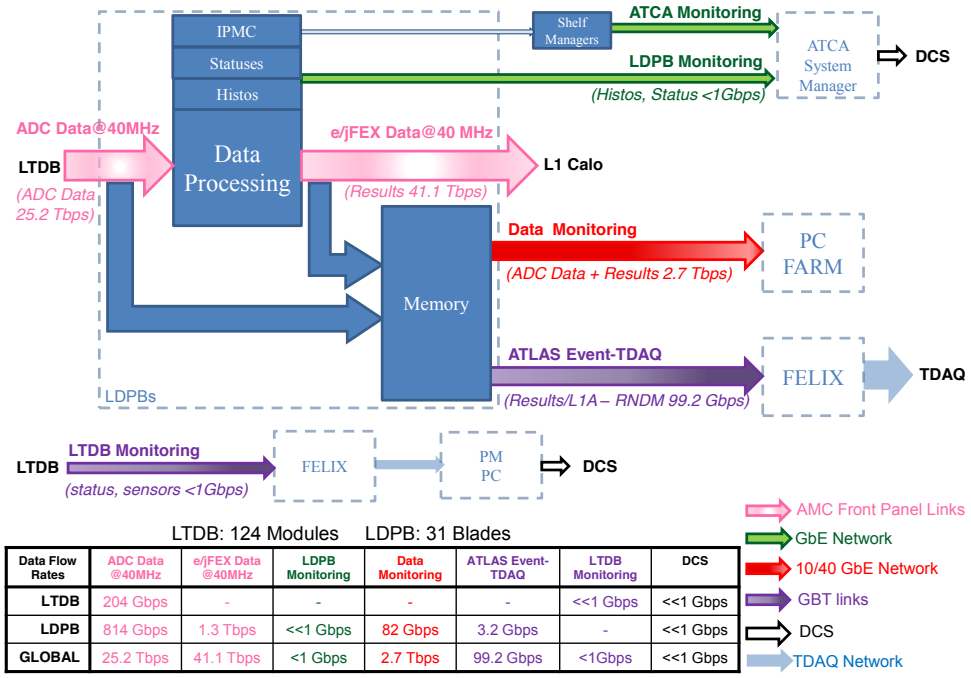
Contents

1 Introduction

This document describes the specifications and the structure of the firmware embedded in the Arria^{TM}10 FPGA, referenced 10AX115R4F40I4SGES (Arria^{TM}10 family ALTERA^{TM}), located on the AMC-A10 of the LAr Digital Processing Board. Interfaces between the various blocks of the firmware are defined.

1.1 Description and specifications

As a reminder of the overall LAr phase I backend system, Figure 57 of the LAr TDR [1] is reproduced in Fig. 1). This document gives the specification of the AMC-A10 firmware which handles the incoming data from the **L**Arg **T**rigger **D**igital **B**oards (LTDB), to deliver reconstructed transverse energy to Ł1CALO and data to TDAQ and monitoring.



Data Flow Rates	ADC Data @40MHz	e/jFEX Data @40MHz	LDPB Monitoring	Data Monitoring	ATLAS Event-TDAQ	LTDB Monitoring	DCS
LTDB	204 Gbps	-	-	-	-	<<1 Gbps	<<1 Gbps
LDPB	814 Gbps	1.3 Tbps	<<1 Gbps	82 Gbps	3.2 Gbps	-	<<1 Gbps
GLOBAL	25.2 Tbps	41.1 Tbps	<1 Gbps	2.7 Tbps	99.2 Gbps	<1Gbps	<<1 Gbps

Figure 1: Figure 57 of LAr TDR: schematic representation of the data paths and associated data flows of the LDPS. Data paths are represented as arrows; the data path name is written above the arrow and the nature of the data, with the total data flow rate, below the arrow. The data sources of the LDPS are represented as blue boxes. The hardware elements of the LDPS are with dashed lines. The table summarizes the data flow rates per LTDB and LDPB for each data path. The last row gives the total rate for the system.

Table 1 summarizes the data flows at the level of one AMC-A10.

The LDPB design was internally (LAr) reviewed: the documentation and the report are available at [2]. The AMC-A10 schematic is presented in Fig. 2.

The firmware on the AMC-A10 performs four main functions, the details of which are described in later sections:

- it handles a number of high speed links in the input from the LTDB,

ADC data @40 MHz	FEX data @40 MHz	LDPB Monitoring	Data Monitoring	ATLAS Event-TDAQ	DCS
204 Gbps	≃300 Gbps	≪1 Gbps	≃20 Gbps	≃ 1 Gbps	≪1 Gbps

Table 1: Data flow for one AMC-A10 board.

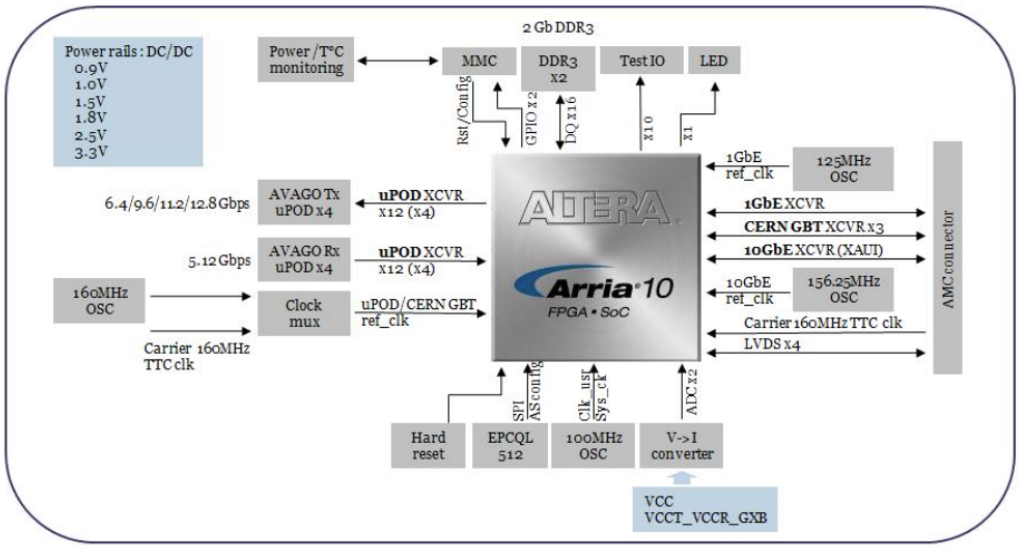


Figure 2: Schematic representation of the AMC-A10 as presented during the LAr internal review [2].

- it performs a filtering algorithm to reconstruct the $E_{\mathrm{T}}^{\text{Super Cell}}$ every 25 ns and identify the Bunch Crossing where the deposited energy was initiated (BCid),
- it outputs results to the Ł1CALO FEXs, and
- it processes and buffers data to be delivered to the TDAQ readout chain and to the monitoring processes, upon a Level-1 accept (L1A).

The first step in the FPGA processing is the deserialization and demultiplexing of the incoming data. The 12-bit ADC data at 40 MHz for each channel must be aligned accordingly for the application of the filtering algorithm.

Next, data words are organised following the detector geometry in a configurable way. Next, the Super Cell transverse energies are calculated with filtering algorithms. The $E_{\mathrm{T}}^{\text{Super Cell}}$ encoding into 10 bits needs to be performed without losing the benefit of the fine energy granularity in terms of cluster reconstruction. The size of the quantization scale for each layer is still under discussion. The nature of the filtering algorithm is not part of this document.

An important point to be considered is the case of saturated inputs which have different shapes than nominal pulses and for which BCid may fail. The proper bunch crossing has to be identified from a dedicated filter on the signal or, for instance, from a neighboring Super Cell (either from a different calorimeter layer or from another Super Cell in the same layer).

The processing has to be performed with a fixed latency.

1.2 AMC-A10 firmware overall structure

Figure 3 presents a block diagram of the AMC-A10 firmware. Each block is described in one section below. The firmware is built around the *low level interface* which controls the hardware components of the AMC-A10. The firmware consists of a main data path (from LTDB to FEX) which has been organised in four logical blocks:

- *input stage* : deserializes, demultiplexes and aligns in time the 12-bit ADC data from the LTDB
- *configurable remapping* : remaps incoming data following the detector geometry, in a configurable way.
- *user code* : applies the filter to reconstruct $E_{\mathrm{T}}^{\text{Super Cell}}$ and determine BCid.
- *output summing* : makes the proper geometrical sums for gFEX and jFEX.

Three other functions are included in the firmware: *tdaq monitoring* which organises the transfer of data to TDAQ and to the local monitoring processes, the *ipbus controller* which is the interface for the AMC-A10 slow-control system and the *ttc* which decodes and provides signals from the TTC system.

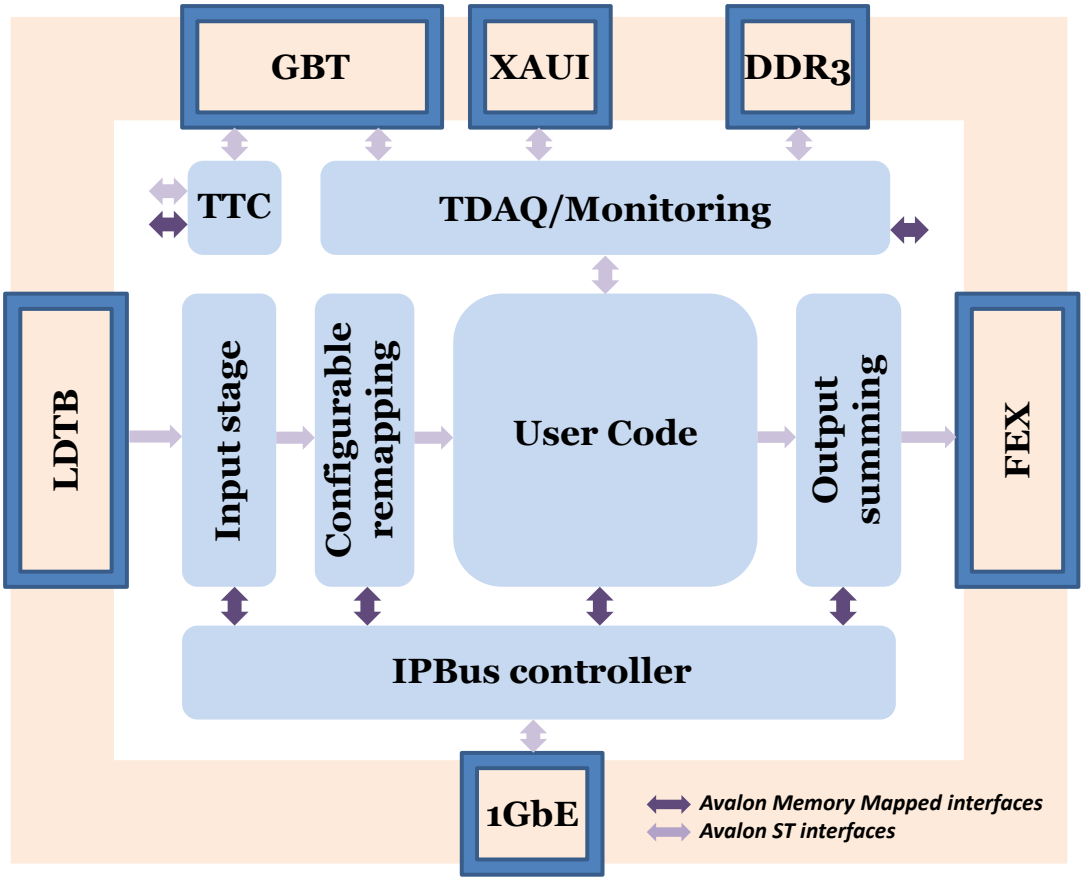


Figure 3: Proposed firmware synopsis.

1.2.1 AMC-A10 firmware clock distribution

Figure 4 presents a block diagram of the various clocks organisation inside the AMC-A10 firmware. Three clocks are foreseen: The 320 MHz clock is used up to the *configurable remapping*. The *configurable remapping* transfers data to the *user code* which receives, computes and transmits, at the speed of 240 MHz; the *output summing* then transfers FEX data to Ł1CALO at 280 MHz.

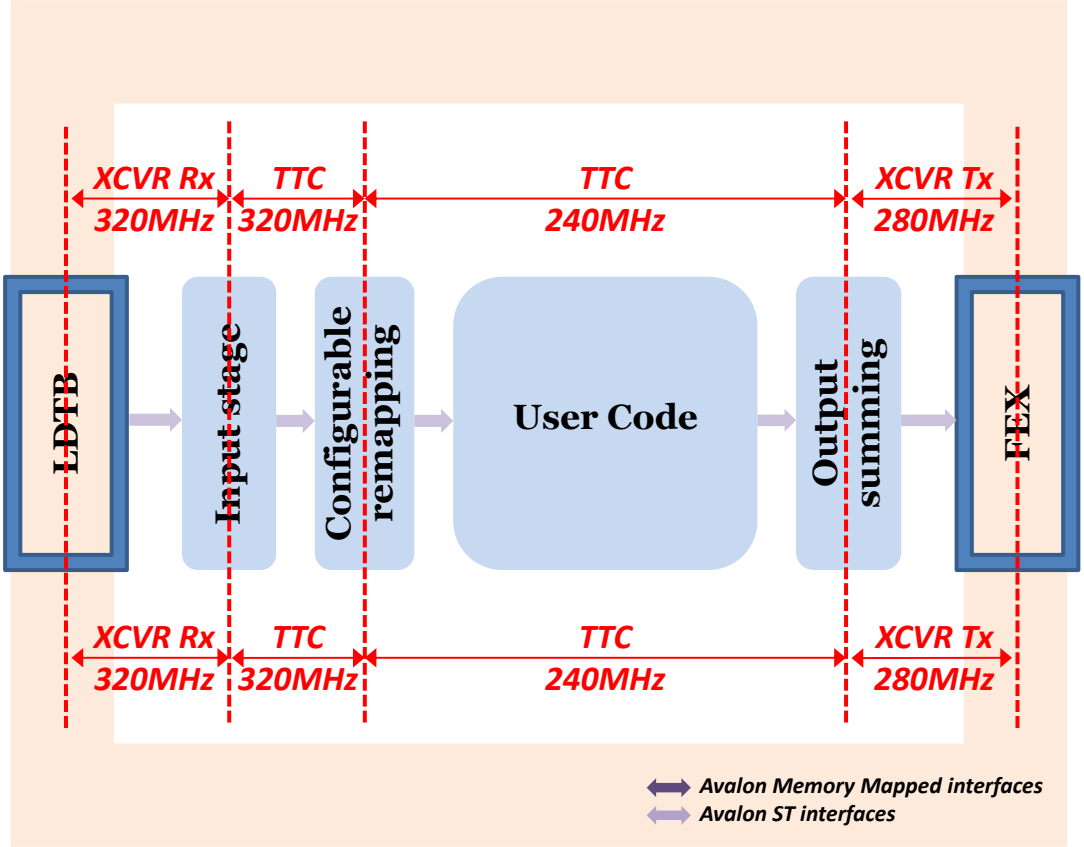


Figure 4: Proposed clock organisation.

1.2.2 Expected firmware footprint in Arria™10

The current estimate of this proposed firmware footprint in the Arria™10 FPGA is presented in Table 2.

1.2.3 Overall latency

Table 3, extracted from [1], presents the summary of the latencies introduced by each step. From the reception of the data to their transmission to FEX, the latency is estimated to 14 LHC clock cycles.

Table 4, presents the summary of the latencies estimated by each processing block as described in this document. From the reception of the data to their transmission to FEX, the latency is estimated to 15 LHC clock cycles.

resource	from FE	to FEX	Filtering	Monitor	1 GbE	GBT	Total	Arria^{TM}10
amount	48 (1)	48 (1)	32x2 (1)	1	1	1		1
Register (10^3)	96 (2)	26 (0.5)	52 (1)	5	4	1	~184	1300
LUT (10^3)	48 (1)	23 (0.5)	38 (0.8)	5	3	2	~120	900
TX/RX	48 (1)		0 (0)	4	1	1	54	96
DSP	0 (0)	0 (0)	768 (16)	10	0	0	778	1518
BlockRam (kbit)	0 (0)	3500 (72)	4320 (90)	5000	92	630	13500	57000

Table 2: List of estimated resources needed for each block of the AMC-A10 firmware and specifications of Arria^{TM}10 FPGA. The line *amount* indicates the number of instances of the object, e.g. 48 fibers from FE, 48 fibres to FEX, 32 paired instances of the *user code* . Numbers in parentheses indicate the corresponding resources for one unit.

from TDR	Latency		Total [BCs]
	[ns]	[BCs]	
Deserializer on LDPS	50	2.0	
Channel Demultiplexing on LDPS	25	1.0	
Pedestal Subtraction	25	1.0	
E_{T} with forward correction	125	5.0	
Digital summation	50	2.0	
Multiplexing 40-320 MHz on LDPS	25	1.0	
Serializer on LDPS	50	2.0	
			14.0

Table 3: Estimated latency budget for the data processing on the AMC-A10 of the LDPB, in the context of the Phase-I upgrade of the LAr trigger readout electronics, as of the presentation of the TDR in 2013.

Current evalutation	Latency		Total [BCs]
	[ns]	[BCs]	
Deserializer from LTDB - LLI	50	2.0	
Input stage	75	3.0	
Configurable remapping	38	1.5	
User code	125	5.0	
Output summing	38	1.5	
Serializer to FEX - LLI	50	2.0	
			15.0

Table 4: Estimated latency budget for the data processing on the AMC-A10 of the LDPB updated for this document.

1.3 Some considered show stoppers

1.3.1 Technical showstoppers

Timing jitter Could the clock jitter induce random errors ?

The reference clocks to the transceivers are purposely cleaned on the carried card; their jitter is therefore very small (<< 1 ps). The clocks used in the processing stages are derived from the recovered LHC clock by a PLL function and distributed over the FPGA with dedicated clock trees of three sorts: global clocks, local clocks and regional clocks.

TimeQuest^{TM} closing Could the timing constraints be too demanding for the TimeQuest^{TM} analyser ?

This would prevent a closure of TimeQuest^{TM}. This is very unlikely. The design being very modular and the modules being quite independent one another, one only needs to solve the timing question for a generic module. All the logic elements used in the design have switching characteristics beyond 400 MHz. The fastest parts of our design runs at 320 MHz only and the user code at 240 MHz. Furthermore, it would be technically possible to run at 160 MHz with twice the number of logic cells. The most demanding part of the design concerning the time is the *configurable remapping*. It has been checked to work at 320 MHz with the TimeQuest^{TM} software.

Latency Could the requested fixed latency achieved be higher than the available latency for the firmware ?

Each blocks in the firmware can be designed to have a fixed latencies and numbers summarised in Table 4 indicate that the total latency is within the specifications of 18 BCs.

FPGA resources Could the firmware footprint exceed the FPGA resources. Following Table 2, this seems unlikely.

1.3.2 Tools limitations

Unstable/incomplete development tools

The following problems related to the QUARTUS^{TM} development tools will most probably be encountered: bugs in all releases, incomplete documentation, lack of reference designs, IP related problems (bad simulation models). Even though these aspects may bring difficulties and slow down the development, they are not considered as show stoppers. The high quality relationship already existing between the labs and the ALTERA^{TM} company will be maintained.

Project development time

The compilation of a complete project in the FPGA can be very long (order of magnitude of 10 hours). Several actions can be conducted in order to optimize the development time:

- Use the modularity of the design to work on small projects before working on the big one.
- Distribute the work within the institutes involved.

1.3.3 Conclusions

The difficulties that we expect to encounter in the project are very classical. Furthermore, similar projects with similar specifications and technological environment have been conducted with success recently (LHCb Muon trigger project). The most serious expected problems are linked to the development tools and not to the project specifications.

1.4 Remarks and some conventions

1.4.1 Remarks

The specifications presented in this document are based on the LAr calorimeter phase-I upgrade TDR [1]. Some of them may be revised in the future like:

- the output fibres speed which would in turn allow to change the data content sent to FEXs,
- encoding on fibers going to FEX (8bit/10 or 64bit/66),
- latency to eFEX/jFEX/gFEX: does it have to be identical ?
- the technique to extract BCid in case of saturation,
- the exact mapping even though it should not affect a priori the proposal made here, because a configurable remapping is proposed and
- the monitoring and TDAQ readout data flows.

A detailed proposal for the FE-BE-Ł1CALO mapping is in preparation; the most up to date information is available at [4] (see the first talk on the agenda).

The section 1 describing *low level interface* is more developed than the other sections; this reflects the development of this part of the firmware which is close to the hardware components of the AMC-A10.

1.4.2 Conventions

The following conventions are used in this document:

- The acronym written as BCID refers to the Bunch Crossing Identifier i.e. the BC number
- The acronym written as BCid refers to the Bunch Crossing Identification i.e. the identification of the bunch crossing where the energy deposit was initiated.

1.5 Document structure

The document is organised as follows: the *low level interface* (LLI) is described in section 2; the *input stage* in section 3; the description of the *configurable remapping* is given in section 4. The structure of the *user code* is presented in section 5. The *output summing* to prepare data for the FEXs is presented in section 6. The way data are prepared to be delivered to TDAQ for readout and for local monitoring is given in section 7 and the *slow control*, via the *ipbus controller* of the AMC-A10 in section 8. The *ttc* decoding stage is presented in section 9. In section 10, a brief description of the code management is given.

2 Low-Level Interface

5

2.1 Introduction

This section describes the **L**ow **L**evel **I**nterface (LLI) part of the Arria10 firmware for the AMC-A10. It describes the firmware interfaces between the external chip (DDR3, GbE, XAUI and LTDB links…) and the FPGA core. The LLI synopsis is shown on Fig. 5.

The list of the LLI input and output signals (interfaces) is presented in Table 5 in section 2.4

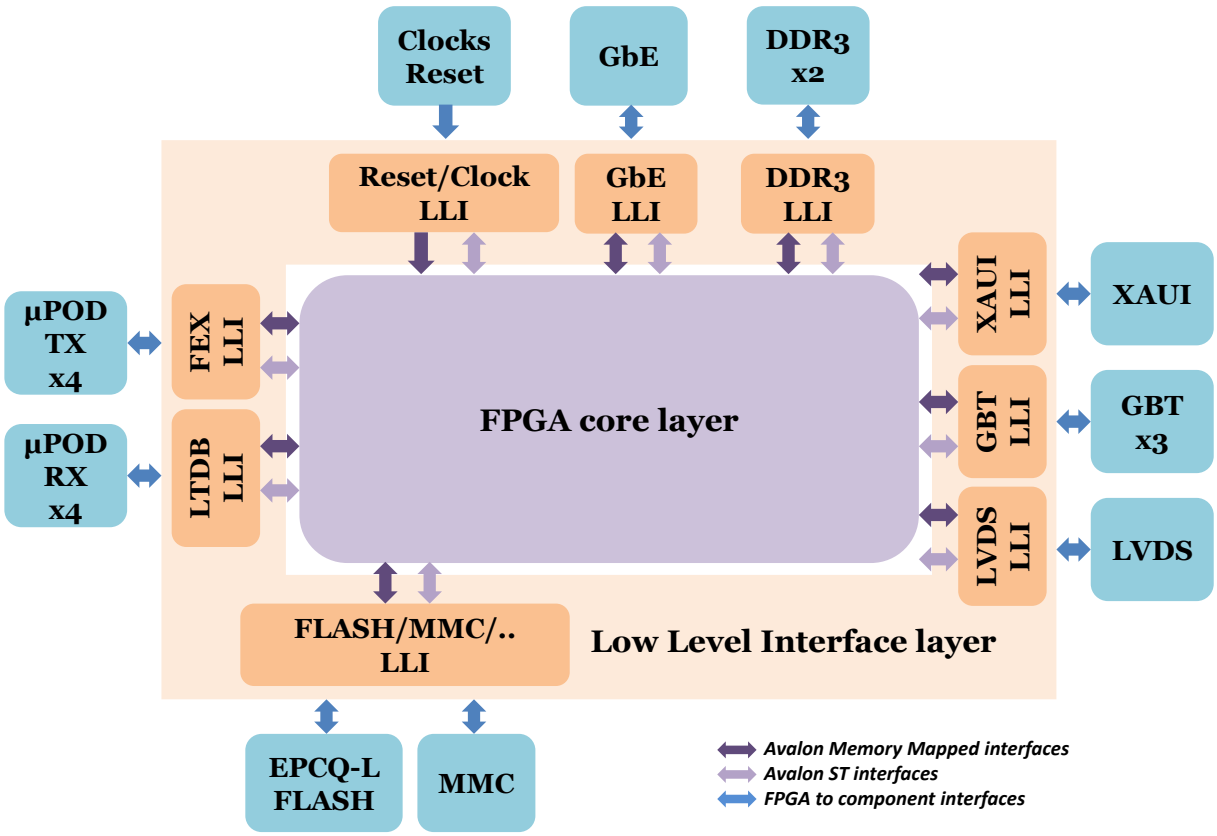


Figure 5: Low-level interface block diagram

All the interfaces between the LLI and the FPGA core layers are designed with standard Avalon Streaming and Avalon memory Mapped interfaces [5]. This allows having well defined and documented standard interfaces between each block.

2.2 Description

2.2.1 Reset/Clocks interface

2.2.1.1 Overview

This LLI part makes the interface between the hardware resets, the different clocks sources and the FPGA core layer. A synopsis of the signal distribution for reset and clocks interfaces is presented on Fig. 6.

Two different sources drive the hardware resets to the Arria^{TM}10 FPGA: the front panel push button and the MMC reset signal. The aim of this block is to provide hardware synchronous reset signals in each clock domain in order to prevent metastability in state machines.

Set apart the high speed transceivers reference clocks, three different types of clocks drive the FPGA clock tree. The 160 MHz TTC clock comes from either the local oscillator or the AMC-A10 TTC clock pin. Another standard 100 MHz clock drives the internal logic and the DDR3 controller. This interface provides the different clock frequency needed by the FPGA core.

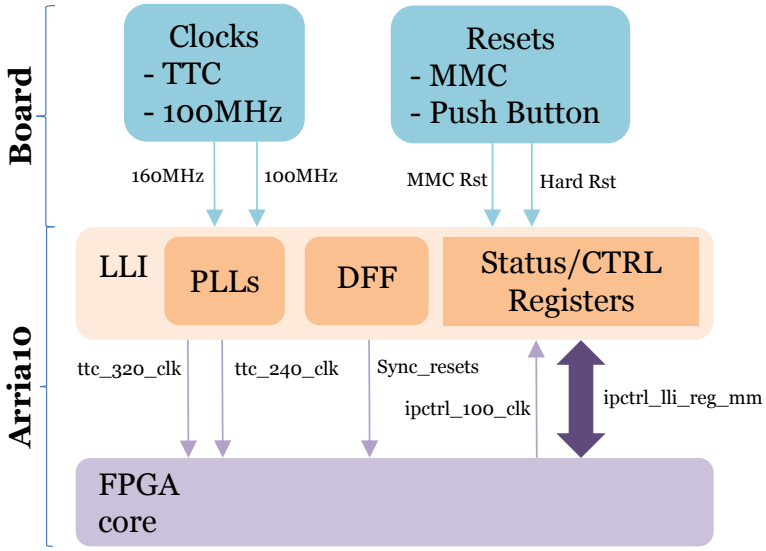


Figure 6: Reset/clock interface block diagram

2.2.1.2 Registers

Around this main path, a status register is implemented to monitor this interface. This status register mainly output the PLL locked signals.

2.2.2 LTDB interface

2.2.2.1 Overview

At the Front End side, one LTDB sends continuous ADC data to the LDPB through 40 optical fibres with a data rate of 5.12 Gbps. Before this serialization, the ADC data are encoded by the LOCic chip with a custom PRBS encoding.

After 70 m of optical fibres, the LDPB receives all these links via 4 AVAGO microPOD^{TM} receivers [6] and the outputs of these 12 channels parallel optics receivers drive the Arria^{TM}10 through high speed CML differential pairs.

This LLI part makes the interface between the microPOD^{TM} and the FPGA core. It performs the serial to parallel interface by implementing the embedded FPGA hardware high speed receivers.

The interface with the FPGA core uses standard interfaces for streaming data and registers access. As there are 48 optics fibres at the input there are 48 output parallel data buses in the streaming interfaces. A synopsis of the LTDB interface block diagram is presented on Fig. 7.

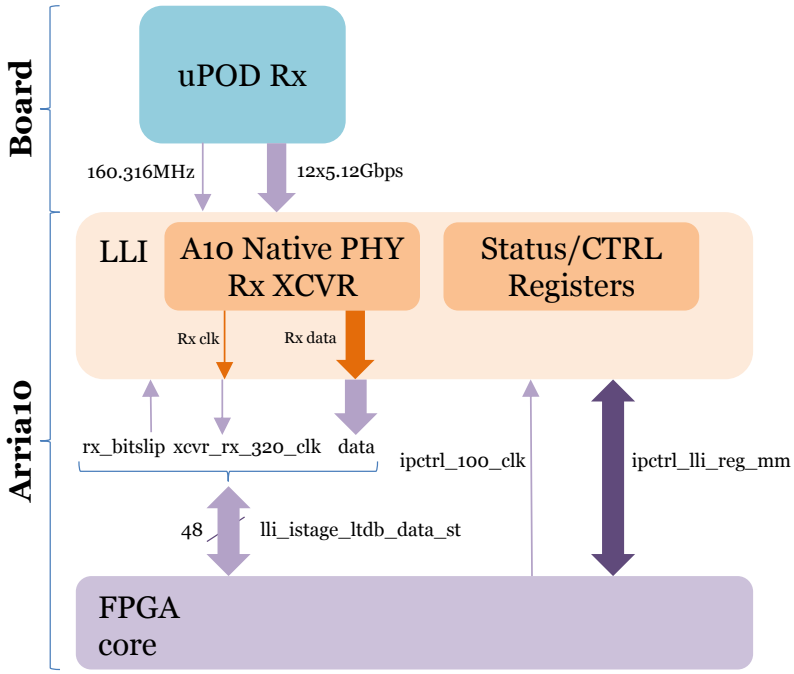


Figure 7: LTDB interface block diagram

Arria^{TM}10 Native PHY receiver block

Configuration The Native PHY part is the hardware IP which implements the receiver part of the Arria^{TM}10 transceiver. Six Arria^{TM}10 transceivers are gathered in a same bank which is synchronized with a 160 MHz external reference clock. The total of 48 receivers will be implemented in 8 banks. A transceiver includes a receiver and a transmitter. These 2 parts are strongly dependent and share the same reference clock but they can handle different speed.

The Native PHY receiver part can be implemented by using the Standard PCS (Physical Coding Sublayer) Native PHY IP (See Arria^{TM}10 Transceiver PHY User Guide in [7]). In that case the output data is a parallel bus with a width of 16 bit running at 320 MHz. But in order to prevent timing issues at FPGA core level the transceiver can be implemented using the Enhanced PCS which allows having a 32 bit data bus running at 160 MHz on the output parallel bus. This clock can be the recovered clock from the input serial data stream or a local clock if the Native PHY is configured with the Rate Match FIFO. In that case this local clock shall have the same frequency as the recovered clock and shall have the same source (TTC system).

Implementation At power up or if a reset occurs, the Native PHY needs a special reset sequence on its analog and digital parts in order to be locked on the input data stream. Once the link is locked, the Rx_freq_locked flags are raised. Otherwise, these flags are not stable and the link has to be reset to start a new sequence to achieve the synchronization on the input serial data.

The Native PHY receiver frame can be detected on a special word by implementing different alignment modes like bit-slip or manual alignment. The word aligner receives the serial data from the Physical Medium Attachment (deserializer and Clock Data Recovery) and realigns the serial data to have the correct word boundary according to the word alignment pattern configured. This alignment sequence should be driven by the “Input Stage” block.

Timing The output parallel data bus is synchronized by the recovered Rx clock or a local clock if the Rate Match FIFO is implemented. Due to the hardware IP (CDR, internal FIFO), there may be

a few clock cycles between the output parallel data bus coming from each optics fibre. So all these data shall be aligned in the “Input Stage” block at the FPGA core level.

2.2.2.2 Registers

A set of status registers will be implemented in order to monitor the locking states of the receiver’s links and the reference clocks. The control registers allows resetting the receiver links.

2.2.2.3 Examples

To be completed. Examples and chronograms describing the output streaming interface to the next stage

2.2.3 FEX interface

2.2.3.1 Overview

At the Backend side the LDPB sends results to the FEX through 48 optical fibres. The data rate is not yet defined but could be 6.4/9.6/11.2 or 12.8 Gbps. The output packets should have a standard format for all FEX with a header and a trailer. This format is not yet defined.

The LDPB drives all these links via 4 AVAGO microPOD™ Transmitters [7] and the inputs of these 12 channels parallel optics transmitters are driven by the Arria™10 through high speed CML differential pairs.

This LLI part makes the interface between the microPOD™ and the FPGA core. It encodes and serializes the incoming data to a serial data frame. The interface with the FPGA core uses standard interfaces for streaming data and registers access. As there are 48 optics fibres at the output there are 48 input parallel data buses in the streaming interfaces.

The FEX interface is built around 2 main blocks which serialize and encode the incoming parallel data bus. The serializer is built with the Arria™10 Native PHY IP. The data encoding is not yet defined: it could be implemented inside the Native PHY or could be an external block. The first idea is to use the 8B10B encoding. Around this main path, a set of control and status registers are implemented to monitor or control this interface. A synopsis of the FEX interface block diagram is presented on Fig. 8.

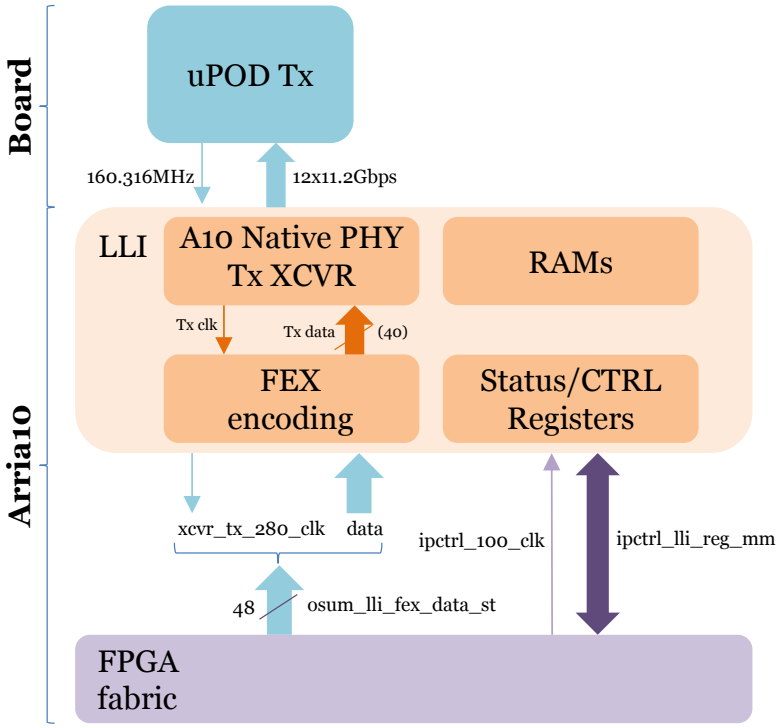


Figure 8: FEX interface block diagram

Arria^{TM}10 Native PHY transmitter block

Configuration The Native PHY part is the hardware IP which implements the transmitter of the Arria^{TM}10 transceiver. Six Arria^{TM}10 transceivers are gathered in a same bank which is synchronized with a 160 MHz external reference clock. The total of 48 receivers will be implemented in 8 banks. A transceiver includes a receiver and a transmitter. These 2 parts are strongly dependent and share the same reference clock but they can handle different speed.

In order to prevent timing issues at FPGA core level the transceiver should be implemented using the Enhanced PCS (See Arria^{TM}10 Transceiver PHY User Guide in [8]) which allows having a 32/40 or a 64 bit data bus on the input parallel bus. The frequency of the synchronization clock depends of the output serial data rate and the input data bus width. For example, if the Native PHY is configured at 11.2 Gbps with a 40 bit input data bus, the clock frequency should be 280 MHz.

Implementation At power up or if a reset occurs, the Native PHY needs a special reset sequence on its analog and digital part in order to lock the output data stream. An external ATX PLL is used to generate the high speed clock which drives the high speed clock of the serializer. When the ATX PLL is locked, an output flag is raised and then the reset sequence may continue. The input parallel data bus is driven by the FEX encoding block when the Native PHY does not implement the internal encoding protocol. For example the 64b/66b protocol is already implemented in the Native PHY.

Timing The input parallel data bus is synchronized by the recovered Tx clock or a local clock if the Rate Match FIFO is implemented. All these data shall been aligned in the previous stages with a fixed latency.

FEX encoding This block receives data from the previous stage "Output Summing block" in a standard format defined for all the FEX boards. This format should have header and trailer in order to identify the packets. Currently the data format is not yet known. In addition and for reliability and control of the system, a CRC word may be added to each output data packet.

These packets shall be encoded before being transmitted to the Native PHY transceiver. Generally for high speed transmission, the 64b/66b encoding is used (40/100G protocol) and allows having only 3.125% overhead. This protocol is implemented inside the Native PHY but, if the 64b/66b encoding is not chosen for the final design, the FEX encoding block shall implement the desired protocol. The other solution could be the 8B10B encoding which is the current foreseen solution. The input and output parallel data buses of this block are strongly correlated to the configuration of the Native PHY. For example, if we want to implement the 8B10B encoding, the Native PHY shall be configured in 40 bit and the input interface of the FEX encoding block shall be 32 bit. In that case the 8B10B encoder will also be implemented in the FEX encoding block.

2.2.3.2 Registers

A set of status registers will be implemented in order to monitor the locking state of the transmitter PLL. The control registers allows resetting the transmitters links.

2.2.3.3 Examples

To be completed. Examples and chronograms describing the input streaming interface from the previous stage.

2.2.4 DDR3 interfaces

2.2.4.1 Overview

The ARRIA 10 is connected to two external MT41JT128M16JT-125 DDR3 for massive storage of data. This DDR3 is a 16 bit data bus memory with a theoretical maximum data rate of 3.2 GB/s. The density is 2 GB distributed in 8 banks of 16K rows, each row containing 1K column of 16 bit (page of 2 KB).

The ARRIA 10 can interface these DDR3 with 2 independent and parallel ALTERA™ EMIF (**E**xternal **M**emory **I**nter**f**ace) IP [9]. This IP can be designed with a hard memory controller and a hard PHY that are part of the Arria™10 chip. The maximum data rate is obtained with the quarter clock rate driven the "TDAQ-Monitoring" block. As only one row per bank can be opened at a given time it is recommended to make accesses from bank to bank for more efficiency. A synopsis of the DDR3 interface block diagram is presented on Fig. 9.

The DDR3 is seen as an Avalon Memory mapped slave by the FPGA core. So the FPGA core interface should act as an Avalon Memory Mapped master in order to read or write data to the DDR3. The Avalon bus timing is given by the EMIF user clock output. This clock has a frequency constrained by the external DDR3 memory and the hard controller clock rate.

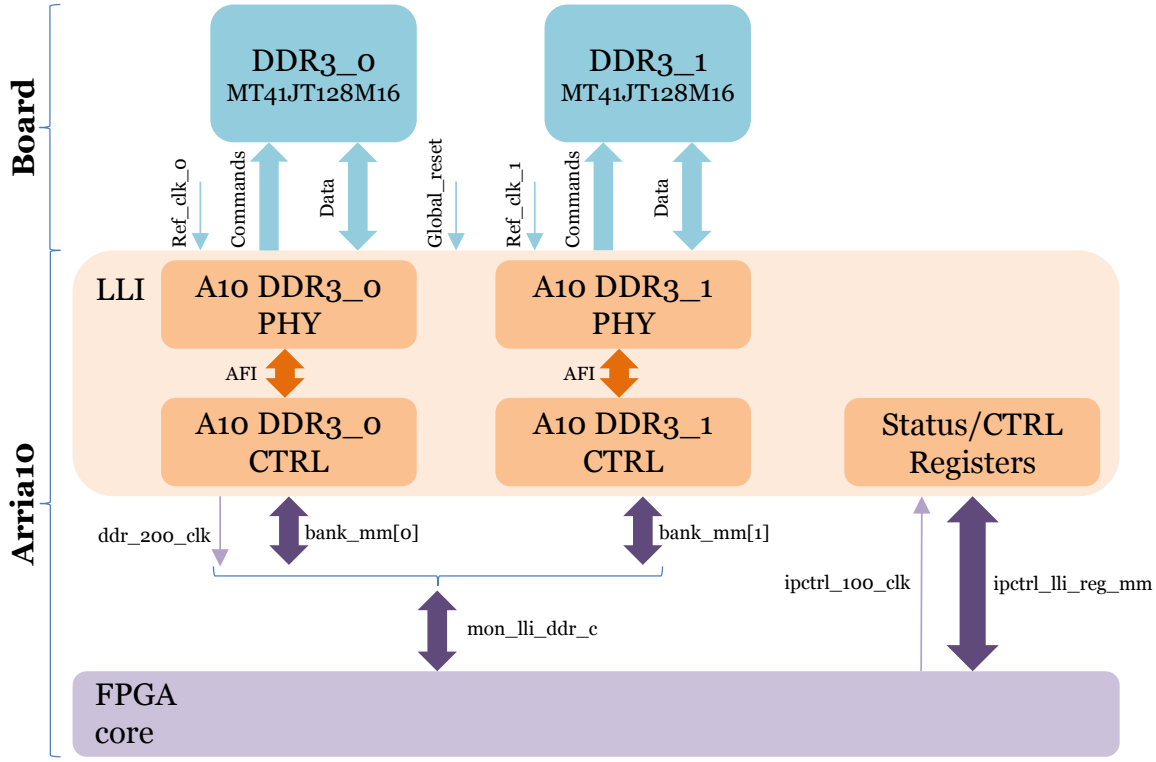


Figure 9: DDR3 interface block diagram

The interface between the ARRIA 10 EMIF hard controller and the FPGA fabric is based on the standard Avalon Interface Memory Mapped interface.

2.2.4.2 Examples

To be completed with chronograms

Quarter-rate DDR3 SDRAM reads

Quarter-rate DDR3 SDRAM writes

Waiting for new release of the ALTERA™ External Memory Interface Handbook for ARRIA 10 EMIF timing.

2.2.5 GbE interface

2.2.5.1 Overview

The GbE port of the LLI is dedicated to slow control and monitoring of the Arria™10 firmware. This port comes from the AMC-A10 connector on a 1.25 Gbps link. The GbE interface makes the interface between the physical link and the next stage which is the IPBus controller block.

This interface use a hardware transceiver coupled with a 10/100/1000 Ethernet MAC. This MAC is an ALTERA™ IP [10].

The GbE interface is designed by the instantiation of the Arria™10 triple speed Ethernet IP which includes the hardware part of the 1.25 PHY and the triple speed Ethernet MAC. In our case, we can use the "10/100/1000 Ethernet MAC with 1000Base-X/SGMII PCS" option. The streaming interface is connected to the "IPBus Controller" block which will be the bridge towards the Avalon MM domain for all the other Avalon MM slaves. The transactions are processed on a 32 bit data bus and are synchronized by a 100 MHz clock. Payload data are UDP packets encompassing IPBus packets.

The GbE has also an Avalon MM interface for internal control and monitoring. We have to see if it can be connected to the Avalon MM domain managed by the IPBus controller block. A synopsis of the GbE interface block diagram is presented on Fig. 10.

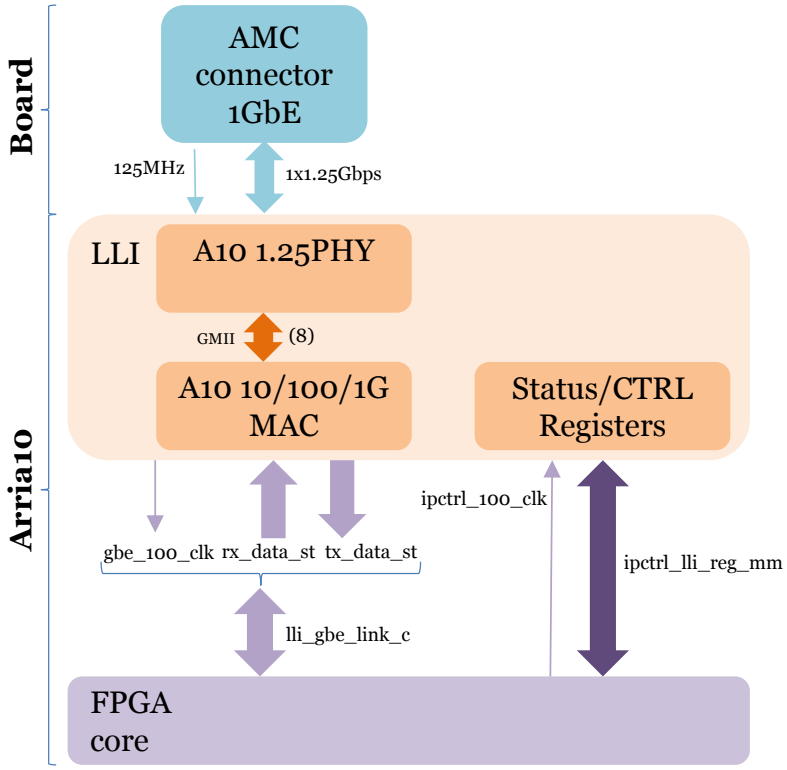


Figure 10: GbE interface block diagram

2.2.5.2 Registers

A set of status registers will be implemented in order to monitor the locking states of the receiver/transmitter link. The control registers allows resetting the receiver and transmitter links.

2.2.5.3 Examples

See section 7 of data sheet [10].

2.2.6 XAUI interface

2.2.6.1 Overview

The XAUI interface of the LLI is dedicated to local fast monitoring using XAUI links coupled with a 10GbE MAC. The XAUI interface comes from the AMC-A10 connector on 4x3.125 Gbps links. The data format exchanged on this interface is not yet defined. The transport protocol can be UDP.

This interface is designed around 2 main blocks. The hardware XAUI PHY includes the hardware transceivers plus the soft XAUI protocol and the 10GbE MAC coming from the ALTERA^{TM} IP. The transport protocol (UDP or others to be defined) is not decoded in this interface but the next stage.

The interface with the FPGA core uses standard interfaces for streaming data and registers access. The streaming interface is performed on a 32 bit data bus running at 312.5 MHz in transmission and reception

The XAUI PHY implements 4 hardware transceivers running at 3.125 Gbps. These transceivers are synchronized by the same 156.25 MHz reference clock. An external ATX PLL shall be implemented in order to drive the high speed clock for the transmitter part. Unlike the Stratix IV FPGA, the Arria^{TM}10 XAUI alignment is not performed inside the hardware transceivers but with a soft IP.

The 10GbE MAC part of this interface is the standard 10GbE ALTERA^{TM} IP [11].

A synopsis of the XAUI interface block diagram is presented on Fig. 11.

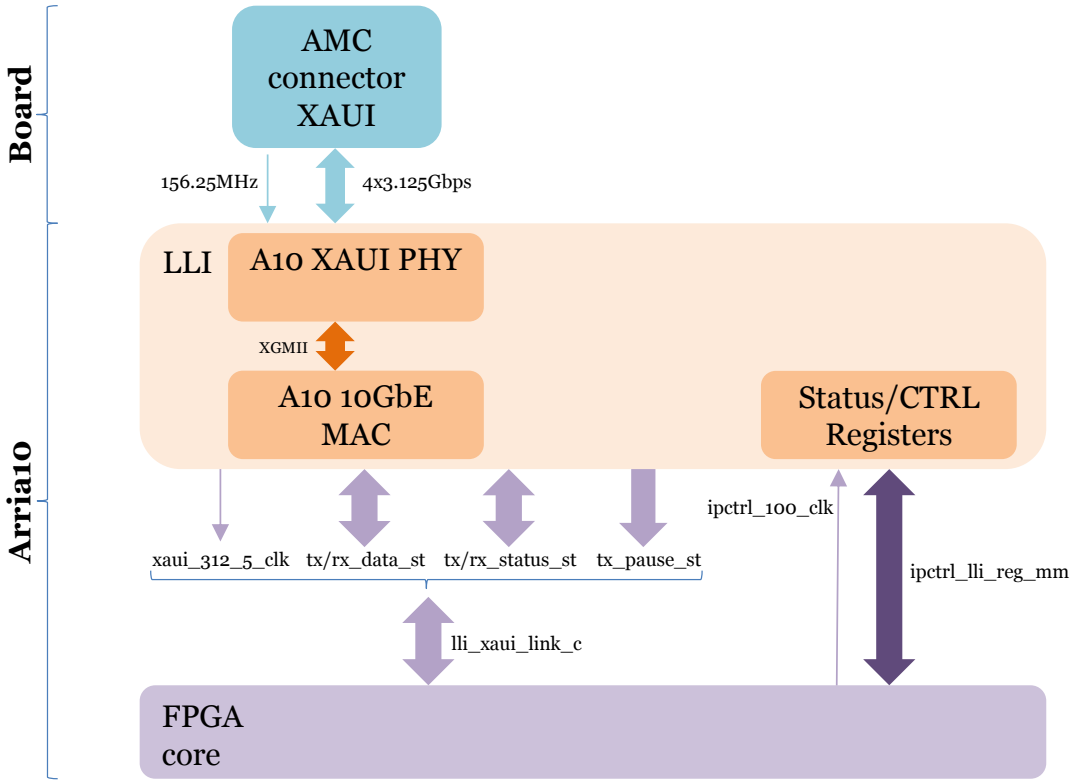


Figure 11: XAUI interface block diagram

2.2.6.2 Registers

A set of status registers will be implemented in order to monitor the locking state of the transmitter PLL and the receiver's links. The control registers allow resetting the receivers and transmitters links.

2.2.6.3 Examples

See section 9 of data sheet [11].

2.2.7 GBT interfaces

2.2.7.1 Overview

A GBT port is available on the AMC-A10 connector. This link targets data transmission between the on-detector and off-detector electronics serving simultaneously applications such as data acquisition, timing, trigger and experiment control. In our application the GBT link will be used to transmit monitoring data to TDAQ system and/or to carry TTC control signals. This link operates at 4.8 Gbps.

The GBT FPGA interface could be divided in 2 main parts. The hardware transceiver part is implemented by the Native PHY IP and the decoding/encoding part is represented by the GBT Tx and GBT Rx blocks. This interface could be based on the GBT-FPGA project[12] which is developed at CERN. For the moment this project targets ALTERA^{TM} Cyclone V or Stratix V. Arria^{TM}10 FPGA is not yet implemented.

This LLI interface should support the "GBT frame" encoding scheme (Reed Solomon) in order to be compatible with Data acquisition and Timing and Trigger Control systems. This encoding is mandatory to obtain a "Latency-Optimized" design. In that case the user data field is 80 bit.

The TDAQ monitoring packets transmitted to the TDAQ system shall have the predefined format given by ATLAS TDAQ system with headers, payload data and trailers. This encapsulation is performed by the "TDAQ monitoring" previous stage. A synopsis of the GBT interface block diagram is presented on Fig. 12.

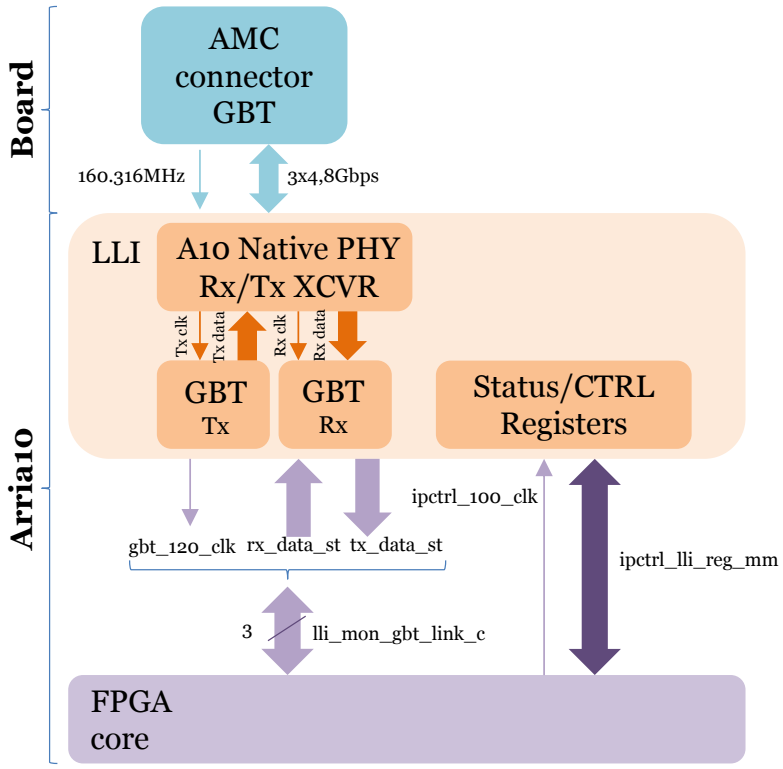


Figure 12: GBT interface block diagram

2.2.7.2 Registers

A set of status registers will be implemented in order to monitor the locking state of the transmitter PLL and the receiver's links. The control registers allows resetting the receivers and transmitters links.

2.2.7.3 Examples

To be completed with chronograms. See also GBT project web page at CERN.

2.2.8 LVDS interfaces

2.2.8.1 Overview

There are 4 LVDS links connected to the AMC-A10 connector. These links could be used to carry TTC signals from the carrier board.

The LVDS links are implemented on Arria^{TM}10 hardware LVDS transmitters and receivers. These modules accept LVDS data up to 1.6 Gbps.

If these LVDS lines are used to carry TTC signals, they shall be synchronized with the 160 MHz TTC clock coming from the carrier. The mandatory signals are Bunch Crossing Reset and the phase of the 40 MHz TTC clock. Other signals like L0A, L1A or Trigger Type signals should be used for TDAQ monitoring purposes. A synopsis of the XAUI interface block diagram is presented on Fig. 13.

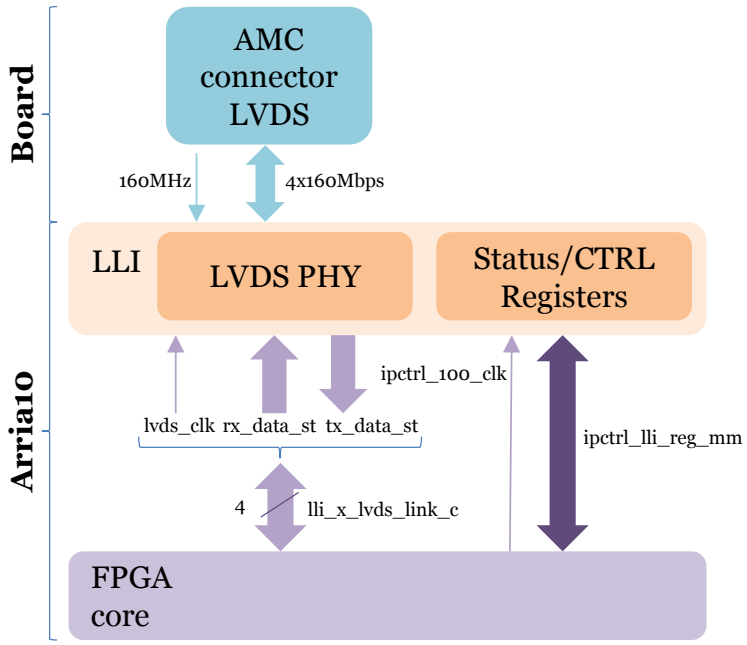


Figure 13: LVDS interface block diagram

2.2.8.2 Registers

A set of status registers will be implemented in order to monitor the locking state of the LVDS links. The control registers allows resetting the receivers and transmitters links.

2.2.8.3 Examples

To be completed with chronograms.

2.2.9 MMC/microPOD^{TM}/Arria^{TM}10 ADC/EPCQ-L FLASH interfaces

2.2.9.1 Overview

Several components are connected to the Arria^{TM}10 with GPIO or serial links like I2C or SPI. The FPGA includes an embedded ADC to monitor external voltage values and an internal sensing diode to check the FPGA temperature. This section of the LLI describes the interface between all these components and the FPGA core.

Arria^{TM}10 device supports an internal temperature sensing diode with a built-in 10 bit ADC circuitry to monitor die temperature. In addition Arria^{TM}10 supports an on-chip voltage sensor. The voltage sensor provides a 12 bit digital representation of the analog signal being observed. The voltage sensor monitors two external differential inputs and six internal power supplies. During standard acquisition, the FPGA temperature and power supplies are monitored by the MMC with external components in order to check all the temperature and power supplies of the FPGA. But for tests and debugging or direct access to the GbE network, it could be useful to use these internal capabilities for the FPGA characterization.

For microPOD^{TM} module control and interrogation, the microPOD^{TM} control interface incorporates an I2C interface and control signals (resets, interrupts). Diagnostic monitors for VCSEL bias, light output power (LOP), temperature, both supply voltages and elapsed operating time are implemented and results are available through the I2C interface. The Arria^{TM}10 implements an I2C master to monitor all these parameters. These values could be read directly by the GbE network and an alarm could be raised and sent to the MMC GPIO when the microPOD^{TM} temperature or power supplies exceed predetermined limits.

At power up or if a reconfiguration reset occurs, the Arria^{TM}10 boots from the EPCQ-L 512 Mbit serial Flash. This Flash can be uploaded through the ALTERA^{TM} Serial Flash Loader IP. In order to update the EPCQ-L from the GbE network, this IP should be managed by a custom bridge linked to the Avalon MM port.

Others GPIO are connected to the MMC for alarm flag, the AMC-A10 clock multiplexer and AMC-A10 ID resistors. A synopsis of the MMC/microPOD^{TM}Arria^{TM}10 ADC/EPCQ-L FLASH interface block diagram is presented on Fig. 14.

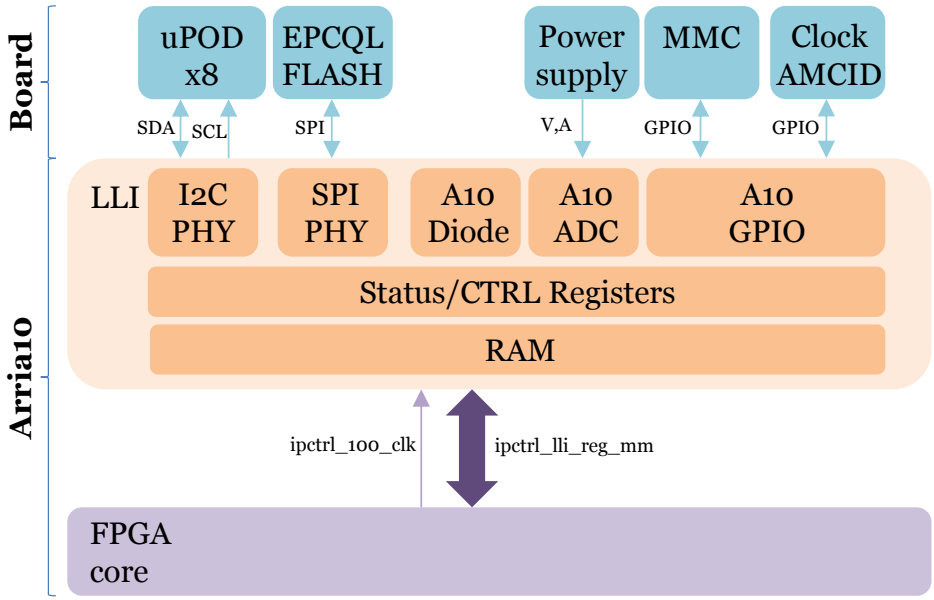


Figure 14: MMC/microPOD^{TM}/Arria^{TM}10 ADC/EPCQ-L FLASH interfaces block diagram

2.2.9.2 Registers

A set of status registers will be implemented in order to monitor the state of the microPOD^{TM}, the FPGA temperature.... The control registers allows resetting the microPOD^{TM}, changing the TTC clock source.....

2.2.9.3 Examples

To be completed.

2.3 Estimated latency

The latency induced by the deserialization of the LTDB data block and by the serialization of the FEX interface can be estimated by simulation. A project has been designed with a simple pattern generator and checker connected to the Arria^{TM}10 transceiver IP. This IP was configured at 6.4 Gbps with a parallel bus of 40 bit and with the bit slip alignment mode. The transmitter and receiver were connected with a delay of 0 ns in the simulation test bench.

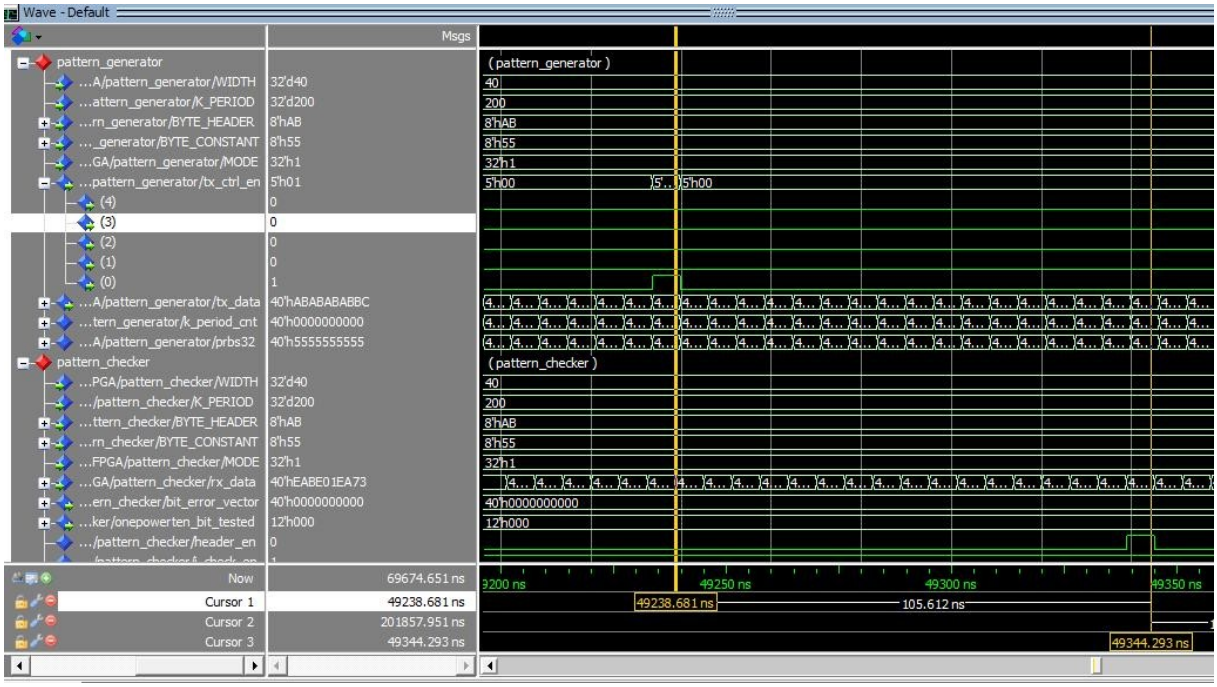


Figure 15: Simulated latency introduced by Arria^{TM}10 transceiver IP (6.4 Gbps/40 bit)

The delay measured between the pattern generator and the pattern checker is ~ 105 ns, as presented on Fig. 15. So if we assume that the serialization and the deserialization induce equal delays, we can estimate that the latency is ~ 2.5 BC for the LTDB interface and also ~ 2.5 BC for the FEX interface.

These values are preliminary and have to be measured and checked with hardware and with Transceiver IP configured at 5.12 Gbs/16 bit and 11.2 Gbps/40 bit.

2.4 Interfaces

The Low-level interface interfaces are described in Table 5.

Interface	#	Description
Clocks		
ipctrl_100_clk	1	IP Bus controller 100MHz clock
Data path		

Table 5: Low-level interface interfaces description (part)

Interface	#	Description	
lli_istage_ltdb_data_st	48	Incoming LTDB data	
	Signal/Bus	**Width**	**Description**
	data	16	Incoming supercell data from the deserializer
	rx_bitslip	1	Signal sent to the Rx part of the input transceiver to shift the data by one bit. Used until the frame is properly aligned.
	valid	1	Transceiver locked
	xcvr_rx_320_clk	1	320 MHz recovered clock from the transceiver
FEX data			
osum_lli_fex_data_st	48	Packed FEX data	
	Signal/Bus	**Width**	**Description**
	data	32	FEX data
	valid	1	FEX data is valid
	xcvr_tx_280_clk	1	Transceiver clock
GBT Link			
lli_mon_gbt_link_c	3	GBT Monitoring link	
gbt_120_clk	1	Transceiver clock	
rx_data_st	1	RX data bus	
	Signal/Bus	**Width**	**Description**
	data	16	TBD
	ready	1	TBD
	valid	1	TBD
tx_data_st	1	TX data bus	
	Signal/Bus	**Width**	**Description**
	data	16	TBD
	ready	1	TBD
	valid	1	TBD
Gigabit Ethernet Link			
lli_ipctrl_gbe_link_c	1	Gigabit Ethernet link	
gbe_100_clk	1	Transceiver clock	
rx_data_st	1	RX data bus	
	Signal/Bus	**Width**	**Description**
	data	32	TBD
	empty	2	TBD
	endofpacket	1	TBD
	error	6	TBD
	ready	1	TBD
	startofpacket	1	TBD
	valid	1	TBD
tx_data_st	1	TX data bus	
	Signal/Bus	**Width**	**Description**
	data	32	TBD
	empty	2	TBD
	endofpacket	1	TBD
	error	1	TBD

Table 5: Low-level interface interfaces description (part)

Interface	#	Description	
	ready	1	TBD
	startofpacket	1	TBD
	valid	1	TBD
LVDS Link			
lli_ttc_lvds_link_c	4	LVDS link	
lvds_160_clk	1	Transceiver clock	
rx_data_st	1	RX data bus	
	Signal/Bus	**Width**	**Description**
	data	16	TBD
	valid	1	TBD
tx_data_st	1	TX data bus	
	Signal/Bus	**Width**	**Description**
	data	16	TBD
	valid	1	TBD
Memory			
mon_lli_ddr_c	1	DDR3 memory bus	
bank_mm	2	DDR3 memory bank	
ddr_200_clk	1	DDR3 200MHz clock	
Registers			
ipctrl_lli_reg_mm	1	Low-level interface registers	
XAUI Link			
lli_mon_xaui_link_c	1	XAUI Monitoring link	
rx_data_st	1	RX data bus	
	Signal/Bus	**Width**	**Description**
	data	32	TBD
	empty	2	TBD
	endofpacket	1	TBD
	error	6	TBD
	ready	1	TBD
	startofpacket	1	TBD
rx_status_st	1	RX status bus	
	Signal/Bus	**Width**	**Description**
	data	40	TBD
	error	7	TBD
	valid	1	TBD
tx_data_st	1	TX data bus	
	Signal/Bus	**Width**	**Description**
	data	32	TBD
	empty	2	TBD
	endofpacket	1	TBD
	error	2	TBD
	startofpacket	1	TBD
	valid	1	TBD
tx_pause_st	1	TX pause bus	
	Signal/Bus	**Width**	**Description**
	data	32	TBD

Table 5: Low-level interface interfaces description (part)

Interface		#	Description	
	tx_status_st	1	TX status bus	
		Signal/Bus	**Width**	**Description**
		data	40	TBD
		error	7	TBD
		valid	1	TBD
	xaui_312_5_clk	1	Transceiver clock	

Table 5: Low-level interface interfaces description

2.5 Registers

TBD

3 Input Stage

3.1 Introduction

As presented in section 2.2.2, the output of this Rx transceiver consists in a 16 bit data path at 320 MHz that feeds the input stage: lli_istage_ltdb_data_st. Each 16 bit data bus has its own 320 MHz clock recovered by the deserializer of the corresponding channel: lli_istage_ltdb_data_st.xcvr_rx_320_clk.

The input stage task can be split into 3 sub-tasks:

- reception and processing of the LTDB data stream as described in section 2.2.2,
- test pattern generator and
- fibre to fibre alignment.

A synopsis of the input stage block diagram is presented in Fig. 16.

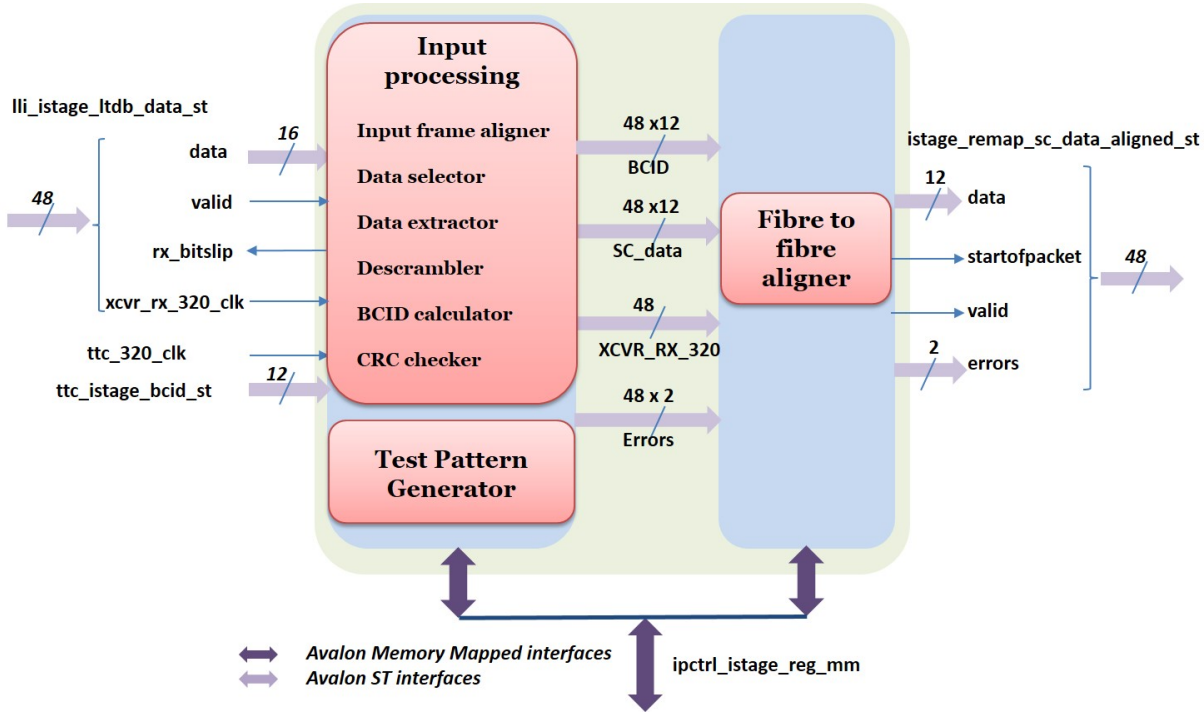


Figure 16: Input stage block diagram

3.2 Description

3.2.1 Input frame alignment

3.2.1.1 Reception and processing of the LTDB data stream

The LTDB data stream uses the LOCic format described in reference [13]. The LTDB data structure consists in a 128 bit frame per BCID represented in Fig 17. This frame contains:

- A special pattern to detect the border of the frame (4 bits = “0101”), represented as T8-T11 in Fig. 17.

- The ADC data, scrambled, corresponding to 8 channels (112 bits) represented as D0-D11 in Fig. 17. D12 and D13 are not used.
- BCID partial information , PRBS5 and PRBS7 (4 bits) , represented as T12-T15 in Fig. 17.
- A CRC word to check the transmission (8 bits) , represented as T0-T7 in Fig. 17.

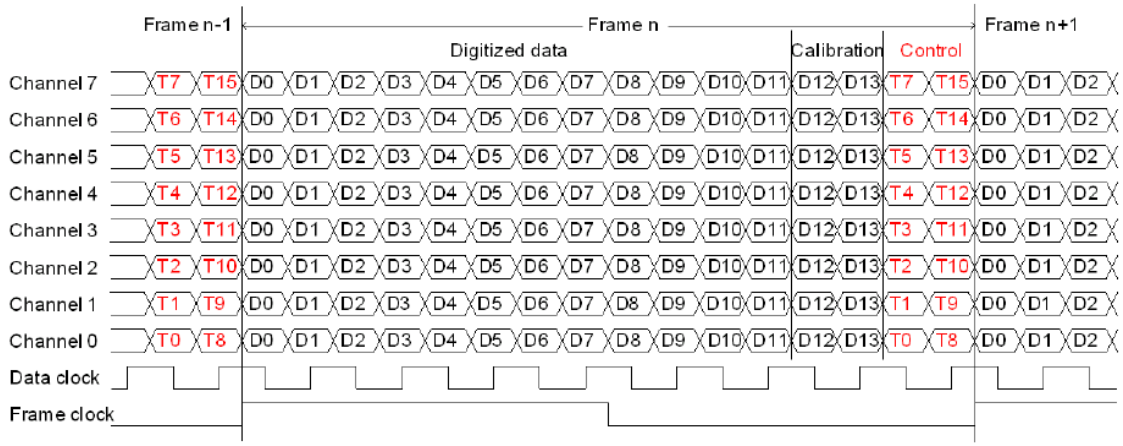


Figure 17: LOCic frame structure

The input stage must:

- Detect the border pattern in the input serial stream to "cut out" 16 bit data words with the proper alignment.
- Extract the ADC data from the parallel data stream. These data are scrambled.
- Unscramble the ADC data stream.
- Extract the CRC from the data stream.
- Compute the CRC from the unscrambled data and compare it to the extracted CRC.
- Extract the BCID information from the parallel data stream (4 bit) and compute the current BCID (12 bit).
- Constantly check that the BCID progression is consistent.
- Provide the fibre to fibre alignment block a data stream and some flags:
  - Unscrambled ADC values : 12 bit @ 320 MHz
  - Start of frame signal (1 bit)
  - BCID value (12 bit)
  - Status flags: error flags (CRC error, BCID sequence error)

3.2.1.2 Test pattern generator

The test pattern generator is provided as a test device that can inject user defined data in the processing stages instead of the data coming from the Front-End LTDB. It can be used when debugging the firmware but also as a tool to show possible failures when the system is under use.

The test pattern generator design is simple. It consists mainly of a dual access RAM containing the patterns to be injected in the processor and of a controller interfaced with the IPBus to store and read back the data and to control the injection into the processor.

A digital multiplexer is used to select the source of the data that will be processed: either the FE LTDB data or the test pattern generator data.

The structure of the data is that of the data encoded in LOCic format. Each frame consists of 8 words of 16 bits, one per channel (Super Cell). These words are delivered at 320 MHz. The LHC machine cycle length being equal to 3564 BCID, the BCID does not return to zero after 4095 (normal binary roll-over) but at 3563. So we need 3564 * 8 = 28512 words to get one full rotation. We retain the value 32768 for the memory depth, since it is the closest power of 2 over 28512. As a consequence, the memory has a 15 bit address bus and a 16 bit wide data bus. The test pattern is seen as a memory block that can be accessed in Write and Read mode by the DCS system (IPBus based).

When the Test_mode bit is not activated (= 0), the address of the memory is blocked at 0 and the multiplexer selects the data from the LTDB. When the Test_mode bit is activated (= 1), the address of the memory is incremented at 320 MHz rate, and returns to zero after address 28511 is reached. In that case the multiplexer selects the data from the dual port RAM.

The memory has 32768 words of 16 bits corresponding to ~ 0.5 Mbit to be compared to the memory available in the Arria^{TM}10 of 50 Mbit. It therefore seems reasonable to allocate one memory per block of 6 transceivers. 8 such blocks are needed to obtain a full 48 fibre system. Therefore the memory dedicated to the pattern generator amounts to about 4 Mbit, i.e $\sim 8\%$ of the total available memory of the FPGA.

3.2.2 Fibre to fibre alignment

Incoming data are not synchronous due to the differences in time of flight and of fibre lengths; data corresponding to one bunch crossing have to be aligned. The principle of operation of the synchronizer is the following:

- there is one FIFO per TX channel (fibre).
- A state machine makes sure that the first word written after the reset of the FIFOs corresponds to the data of BCID = 0.
- Once all the FIFOs are in the state "not_empty", they can be read by the system with a common clock by connecting the valid signal to the "read_enable" signals of all the FIFOs.

A master_clock at 320 MHz derived from the system clock recovered by the TTC can be used: ttc_320_clk. Data are written in the FIFOS and read out from the FIFOs at 320 MHz.

The FIFO depth is relatively small. It only needs to accommodate the biggest time difference between the set of fibres. By construction, the synchronizer can handle any time difference value. For instance, for an alignment depth of four BC, and considering the eight data words to write at each BC, it would require a FIFO with 4 x 8 = 32 words, which is very small. In total, for 48 fibres, the corresponding FIFOs require 3888 (ALMs) and 25576 memory bits, which is less than respectively 1% and 0.5 % of the available resources.

3.3 Estimated latency

The estimated latency of this stage is in the range 2.5 to 3 BCIDs (62 to 75 ns), counted from the input of the TX reception stage to the output of the synchronization FIFO. This number comes from a measurement done on a mock-up implemented on a CYCLONE V Gx design kit, and extrapolated to the Arria^{TM}10. The figure must be handled with some caution.

3.4 Interfaces

The interfaces of input stage block are described in table 6.

Interface	**#**	**Description**	
Clocks			
ipctrl_100_clk	1	IP Bus controller 100MHz clock	
ttc_320_clk	1	TTC 320MHz recovered clock	
Data path			
istage_remap_sc_data_aligned_st	48	Supercell ADC data aligned on the TTC 320MHz clock	
	Signal/Bus	**Width**	**Description**
	data	12	Supercell ADC data
	error	2	Report CRC errors or BCID errors on the corresponding channels
	startofpacket	1	Indicates the first word in the series of 8 words of each BCID packet
	valid	1	Indicates that the data going out of the input stage can be used by the configurable remapping. This signal is activated when all the selected channels are synchronized and re-timed without errors.
lli_istage_ltdb_data_st	48	Incoming LTDB data	
	Signal/Bus	**Width**	**Description**
	data	16	Incoming supercell data from the deserializer
	rx_bitslip	1	Signal sent to the Rx part of the input transceiver to shift the data by one bit. Used until the frame is properly aligned.
	valid	1	Transceiver locked
	xcvr_rx_320_clk	1	320 MHz recovered clock from the transceiver
Registers			
ipctrl_istage_reg_mm	1	Input stage registers	
TTC data			
ttc_istage_bcid_st	1	Current BCID value provided by the TTC receiver	
	Signal/Bus	**Width**	**Description**
	data	12	Received BCID
	valid	1	BCID is valid

Table 6: Input stage interfaces description

3.5 Registers

The registers implemented in the input stage are summarized in table 7.

Register name	Type	Size (32 bits)	Offset	Description
istage_test_data	R/W RAM	8 x 32 k x 16 bits	0x00000	LOCic like data
Istage_test_mode	R/W Register	1	0x10000	Normal or test mode operation selection
istage_fibre_select	R/W Register	48	0x10004	Fibre selection

Table 7: Input stage registers description

4 Configurable Remapping

4.1 Introduction

The configurable remapping block is used to reorder data following the detector geometry, for each BC, coming from the *input stage*. This block groups data from Super Cells belonging to the same Trigger Tower (TT) coming in over **istage_remap_sc_data_aligned_st** interface, and then sends them to one of the *user code* block instances over **remap_user_remap_data_st** interface. A synopsis of the *configurable remapping* interface block diagram is presented in Fig. 18.

There are two main reasons for grouping Super Cells according to detector geometry:

- to prepare data for jFEX and gFEX where the unit is a TT and
- to potentially use the BCid from a neighbouring Super Cell if the pulse shape is distorted because of saturation.

As the data arrangement and the geometry are not uniformed across the detector and therefore are different for each AMC-A10, it is proposed to configure the remapping process at power up. Data from several incoming fibres are grouped to build a TT. The *configurable remapping* stage therefore allows to have a unique version of the *user code* .

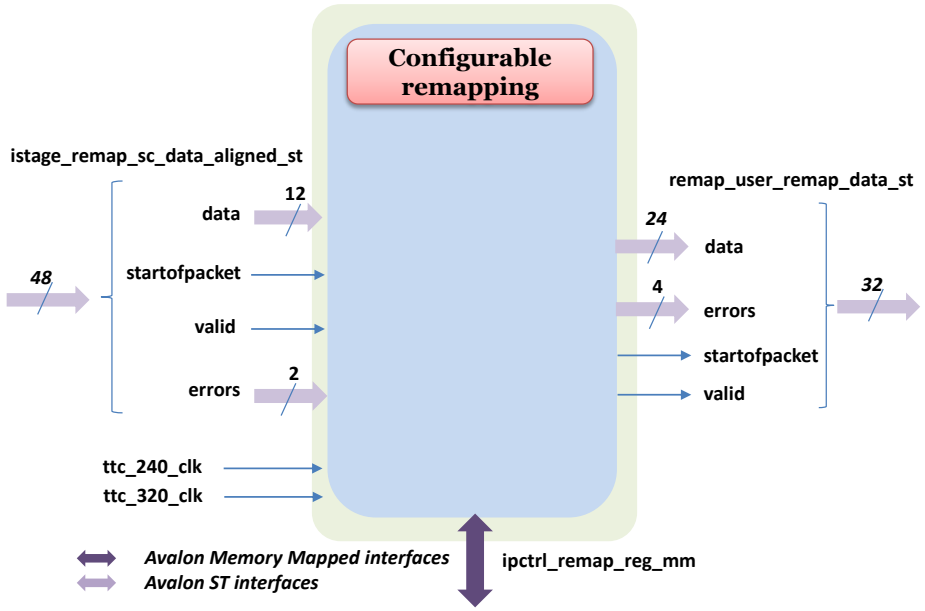


Figure 18: *configurable remapping* block diagram

4.2 Description

Each interface **istage_remap_sc_data_aligned_st** corresponds to typically 10 Super Cells but there are cases where it corresponds to 12 Super Cells. The *configurable remapping* therefore needs to be able to remap 12 data words from one TT. Each output interface **remap_user_remap_data_st** is constructed over a pair of 12 bit buses together with error bits and the frame delimiter.

Thus, the *configurable remapping* block has 48 input data streams aligned on **ttc_320_clk** (8 samples within 1 BC) and 64 output data streams aligned on **ttc_240_clk** (6 samples within 1 BC) paired to 32 data streams to the number of TTs. The waveforms of the *configurable remapping* input and output

signals are shown in Fig. 19. The time-division multiplexing of the data and packing two samples within 24 bit output word can be seen there.

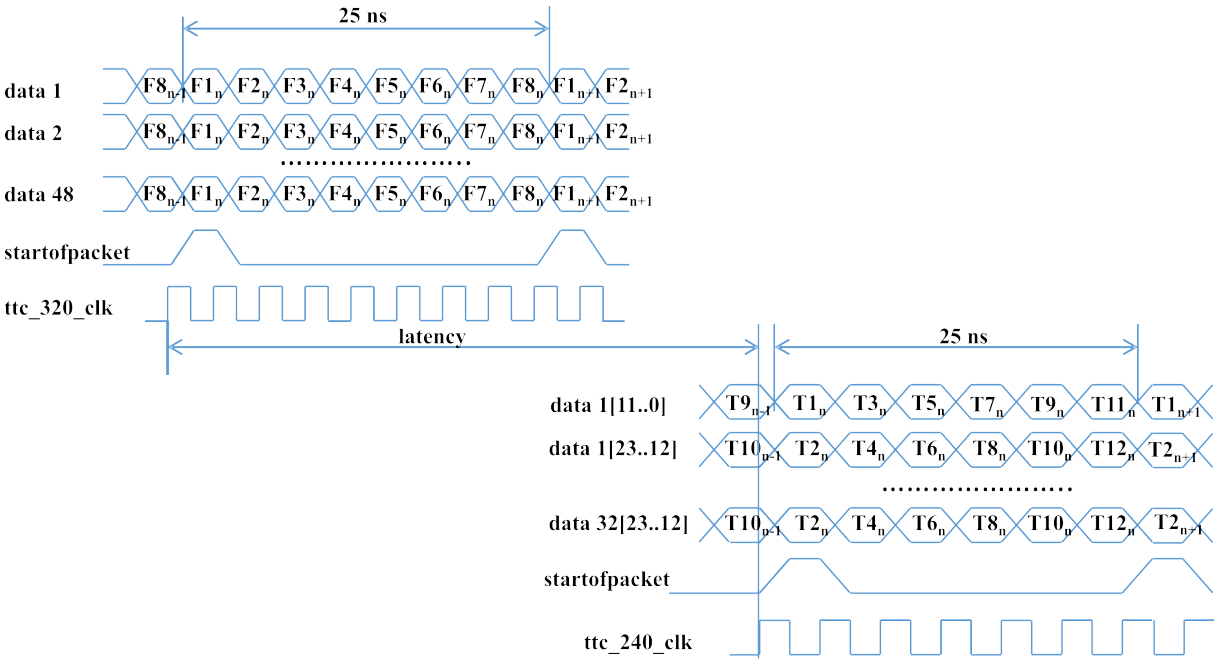


Figure 19: Waveforms of the *configurable remapping* input and output signals. Here: FX_n — X-th fibre sample for the n-th BCID, TX_n — X-th Trigger tower sample for the n-th BCID.

4.3 Estimated latency

The estimated latency for *configurable remapping* block is less than 1.5 BC. This number was obtained by a functional verification of a principle *configurable remapping* realization. Basically 1 BC is needed to store all inputs samples with the same BCID in the FPGA block memory. Several additional clock cycles are needed to move the data from clock domain of **ttc_320_clk** to the one of **ttc_240_clk**.

4.4 Interfaces

The configurable remapping block has three interfaces:

- **ipctrl_remap_reg_mm** used to load configuration data to the block. It is the Avalon MM interface;
- the input data come over 48 streams of the **istage_remap_sc_data_aligned_st** interface. It is the Avalon Stream interface;
- over 32 stream pairs of the **remap_user_remap_data_st** interface the remapped data go to the *user code* block. It is the Avalon Stream interface;

Full description of the interfaces is presented in Table 8.

Interface	#	Description
Clocks		
ipctrl_100_clk	1	IP Bus controller 100MHz clock
ttc_240_clk	1	TTC 240MHz recovered clock

Table 8: Input remapping interfaces description (part)

Interface	#	Description	
ttc_320_clk	1	TTC 320MHz recovered clock	
Data path			
istage_remap_sc_data_aligned_st	48	Supercell ADC data aligned on the TTC 320MHz clock	
	Signal/Bus	**Width**	**Description**
	data	12	Supercell ADC data
	error	2	Report CRC errors or BCID errors on the corresponding channels
	startofpacket	1	Indicates the first word in the series of 8 words of each BCID packet
	valid	1	Indicates that the data going out of the input stage can be used by the configurable remapping. This signal is activated when all the selected channels are synchronized and re-timed without errors.
remap_user_remap_data_st	32	Reordered ADC data aligned on the TTC 240MHz clock	
	Signal/Bus	**Width**	**Description**
	data	24	ADC data of two supercells
	error	4	CRC error, BCID error for two supercells
	startofpacket	1	First word of the packet
	valid	1	Valid data to User-code
Registers			
ipctrl_remap_reg_mm	1	Configurable remapping registers	

Table 8: Input remapping interfaces description

4.5 Registers

The register map for the *configurable remapping* block depends on the realization of this block. So these registers are to be defined.

5 User Code

5.1 Introduction

The *user code* block receives the remapped Super Cell data from the *configurable remapping* block and outputs synchronously the reconstructed transverse energy, $E_{\mathrm{T}}^{\text{Super Cell}}$, the bunch crossing, BCid , and some quality and error bits to the *output summing* block as represented in Fig 3.

The multiplexed input data from the *configurable remapping* block consists of 32 streams of **remap_user_remap_data_st**. Each stream consists of two sets of up to six Super Cells (typically five), associated to one Trigger Tower (TT) as described in section 4. It is foreseen to run one instance of the *user code* per input stream, leading to an input data rate of 24 bit at 240 MHz.

The output to the *output summing* block consists of 40 bit at 240 MHz on **user_osum_out_data_st**. (2x14 bits for $E_{\mathrm{T}}^{\text{Super Cell}}$ 2x4 bit for a quality factor and 2x2 bit for errors) The transverse energy calculation, $E_{\mathrm{T}}^{\text{Super Cell}}$, and BCid are performed by a filtering algorithm.

When the pulse shape is distorted, e.g. in case of saturation, BCid may fail. In that case, a special treatment, to recover the BC where the signal was deposited and to estimate the energy, has to be included in the available latency.

The overall latency introduced by the *user code* is fixed.

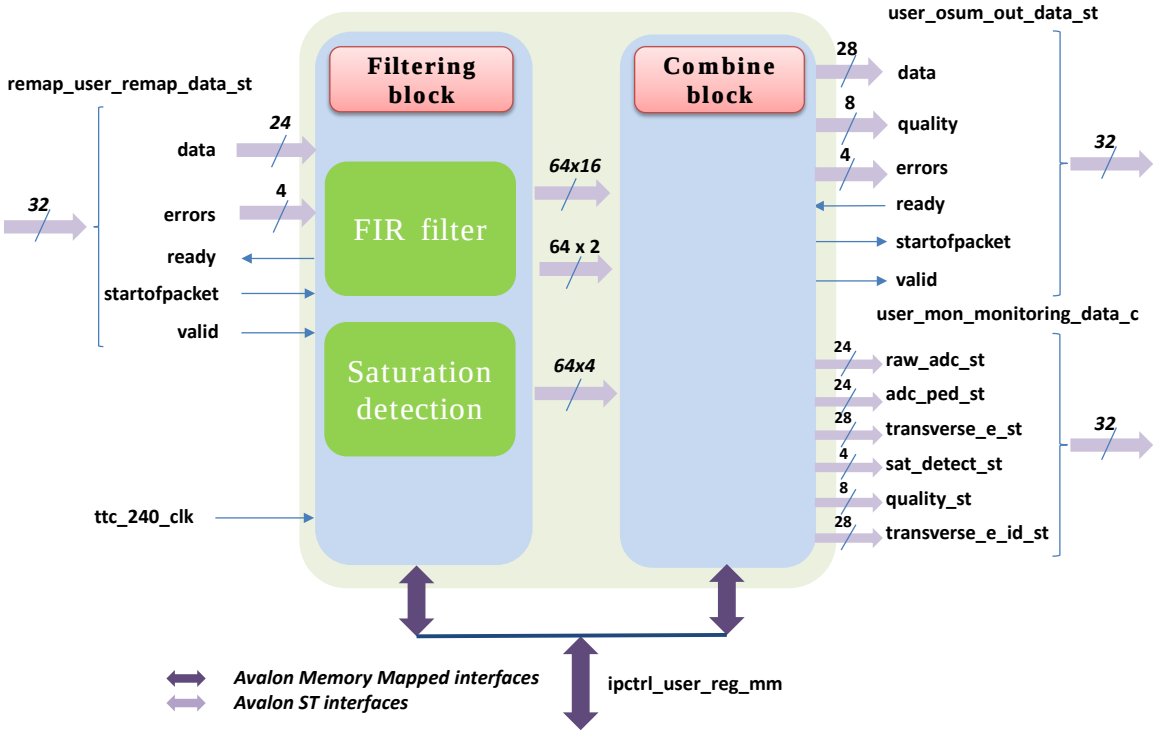


Figure 20: *user code* block diagram. Basic data flow consists from 32 streams of two Super Cells in 2×12 bit (input) or 2×11 bit (output). The numbers associated to the interfaces between the two sub-blocks refer to one Super Cell only.

5.2 Description

The *user code* consists of two sub-blocks as presented on Fig. 20:

- the *Filtering block* reconstructs $E_{\mathrm{T}}^{\text{Super Cell}}$ and identifies the BC, which runs two parallel tasks:

  - the *FIR Filter* converts 12 bit ADC to transverse energy by a filtering algorithm and
  - the *Saturation detection* detects irregular pulse shapes, as induced by upstream saturation, or distorted pulse shape due for instance to noise.

- the *Combine block* combines the outputs from these two tasks and provides $E_{\mathrm{T}}^{\text{Super Cell}}$ in the correct BC and one quality bit.

The *user code* also provides data for monitoring, error detection and possibly histogramming. The monitoring infrastructure is described in section 7.

5.2.1 FIR filter task

Several algorithms, to reconstruct $E_{\mathrm{T}}^{\text{Super Cell}}$ in high pile-up conditions, have been studied; all of them based on FIR filter, such as is presented in equation 1,

$$E_T(m) = \sum_{i=0}^{N-1} a_i \cdot (ADC_{m-i} - ped_{m-i}), \tag{1}$$

where a_i is the calibrated filtering co-efficient, N the number of time samples entering the filter and m the BCID index. In this document, the choice of the filter is not addressed. Some details on the filter performance are available in the TDR [1] and in some presentations [14] [15] [16]. The identification of the BC is performed by applying conditions; $E_{\mathrm{T}}^{\text{Super Cell}}$ is output when the condition is satisfied.

5.2.2 Saturation detection task

If the analog pulse is saturated, the standard algorithm may not provide the correct transverse energy for the corresponding BC. The treatment of saturated signal has not been extensively studied. It is crucial to be able to identify the correct BC and provide high energy in case a very high energy was deposited: a wrong BCid may lead to the loss of a event (as the trigger is blind for four/five BC after a L1A) with a high energy particle. Recent studies done on simulated data [20] show that BCid is achievable even when the pulse shape is distorted because of saturation.

In case BCid cannot be extracted from the pulse itself, one possible idea is to get the BCid from a neighbouring cell, which is known not to saturate. This operation needs to be performed in the shadow of the FIR filter task.

5.2.3 Combine sub-block

This sub-block combines the output of *FIR filter* and of the *Saturation detection* tasks. This sub-block introduces an extra latency. To keep a total fixed latency, all the operations have to be performed even in the absence of saturation.

The input of the combine sub-block is data streams with 14 bits for $E_{\mathrm{T}}^{\text{Super Cell}}$ per Super Cell.

5.3 Estimated latency

FIR filter studies in high pile-up condition have shown that, in order to measure the transverse energy, the FIR filter has to be applied to three samples before, and one sample after the BC. Therefore intrinsic latency is four BCs. The latency of the saturation detection algorithm should be less or equal than the one of the FIR filter[1]. The latency introduced by the *Combine block* is assumed to be less than one BC.

[1]This algorithm evaluates only pulse before the peak of the pulse, therefore the required latency tends to be less than FIR filter.

Adding the individual contributions, one obtains a minimum latency for the *user code* block is five BCs. The implementation of the FIR filter has been studied by several groups, whereas the treatment of the saturation and the combination stage need to be studied in order to provide more solid numbers.

5.4 Interfaces

Interface	**#**	**Description**	
		Clocks	
ipctrl_100_clk	1	IP Bus controller 100MHz clock	
ttc_240_clk	1	TTC 240MHz recovered clock	
		Data path	
remap_user_remap_data_st	32	Reordered ADC data aligned on the TTC 240MHz clock	
	Signal/Bus	**Width**	**Description**
	data	24	ADC data of two supercells
	error	4	CRC error, BCID error for two supercells
	startofpacket	1	First word of the packet
	valid	1	Valid data to User-code
user_osum_out_data_st	32	Processed data block from User-code	
	Signal/Bus	**Width**	**Description**
	data	28	Two 14 bits data words from User-code packed in a 28 bits word
	error	4	CRC error, BCID error for two supercells
	quality	8	Quality information
	startofpacket	1	First word of the packet
	valid	1	Valid data from User-code
		Monitoring	
user_mon_monitoring_data_c	32	Monitoring data	
adc_ped_st	1	ADC data after substracting pedestal. 12 bits per SC	
	Signal/Bus	**Width**	**Description**
	data	24	ADC data without pedestal data
	startofpacket	1	First cell
	valid	1	Valid data from User-code
quality_st	1	Quality from combine block. 4 bits per SC	
	Signal/Bus	**Width**	**Description**
	data	8	Quality data
	startofpacket	1	First cell
	valid	1	Valid data from User-code
raw_adc_st	1	ADC data before substracting pedestal. 12 bits per SC	
	Signal/Bus	**Width**	**Description**
	data	24	ADC data
	startofpacket	1	First cell
	valid	1	Valid data from User-code
sat_detect_st	1	Output of the saturation detection. 2 bits per SC	
	Signal/Bus	**Width**	**Description**
	data	4	Saturation detection data
	startofpacket	1	First cell

Table 9: User code interfaces description (part)

Interface		#	Description	
		valid	1	Valid data from User-code
	transverse_e_id_st	1	Transverse energy E_T from combine block. 14 bits per SC	
		Signal/Bus	**Width**	**Description**
		data	28	E_T data
		startofpacket	1	First cell
		valid	1	Valid data from User-code
	transverse_e_st	1	Transverse energy E_T from filtering block. 14 bits per SC	
		Signal/Bus	**Width**	**Description**
		data	28	E_T data
		startofpacket	1	First cell
		valid	1	Valid data from User-code
Registers				
ipctrl_user_reg_mm		1	User code registers	

Table 9: User code interfaces description

- remap_user_remap_data_st: Raw ADC data from *configurable remapping* block as in described in Section 4. In total 32 data streams for 320 Super Cells with 240 MHz clock. Each stream consists from 24 bits word which corresponding a pair of Super Cell, 12 bit ADC data. Within 1 BC, *user code* block receive 12 Super Cells within a Trigger Tower for each data stream. Corresponding signals (error, ready start of packet, valid and TTC recovered clock) are also input to the *user code*.
- ttc_240_clk: 240 MHz TTC recovered clock directly from *low level interface*, which is used for base clock for *user code*.
- user_osum_out_data_st: Output to the *output summing* from the *user code* block. Transverse Energy after combining the output of the filtering and saturation detection sub-blocks. The output data stream is same format with the input, 32 data stream for 320 Super Cells with 240 MHz clock. Each Super Cell carries 14 bits data, and quality information with 4 bit. Corresponding signals (error, ready start of packet, valid and TTC recovered clock) are also sent to the *output summing* block.
- ipctrl_user_reg_mm: Obtain coefficients through Avalon MM from the register in IPbus block. Details are described in Section 5.5.
- user_mon_XXX_st: Output to the circular buffer in the monitoring block. The format of data stream is similar with main stream to *output summing* block. This stream establishes the access from monitoring block for each calculation stages: pedestal subtraction (optional, before and after subtraction), transverse energy (transverse_et), saturation detection output (sat_detect) and quality bits (quality) and final output after combining two sub-blocks (transverse_et_id).

5.5 Registers

The registers to be implemented in the *user code* block is summarized in Table. 10. These are coefficients for FIR Filter, Saturation detection and combine sub-blocks.

Register name	Type	Size	Description, number of parameters per Super Cell
user_ped_mm	R/W Register	320 x 14 bits	pedestal, 1
user_filco_mm	R/W Register	320 x N-stage x 14 bits	N-stage FIR co-efficients, N
user_con_mm	R/W Register	320 x 3 or 4 x 14 bits	FIR condition parameter, 3-4.
user_sat_mm	R/W Register	320 x 3 or 4 x 14 bits	Parameter for Saturation detection, 3 or 4.
user_com_mm	R/W Register	320 x 3 or 4 x 14 bits	Parameter for combine block, 3 or 4.

Table 10: User block registers description

6 Output Summing

6.1 Introduction

The main task of the *output summing* block it to group the data received from the *user code*, to calculate the sums over specific $\eta - \phi$ areas and to send the data to the FEX output fibres (Fig. 21).

Another task is to adjust the precision (number of bits) for each FEX and prepare the encapsulation of the data, headers and trailers sent to the *low level interface* and further to the FEX. Another task of the block is to duplicate several FEX outputs two or more times according to the number of receivers of the data from specific $\eta - \phi$ areas. Finally the block sends selected sums for monitoring. A synopsis of the *output summing* interface block diagram is presented in Fig. 21.

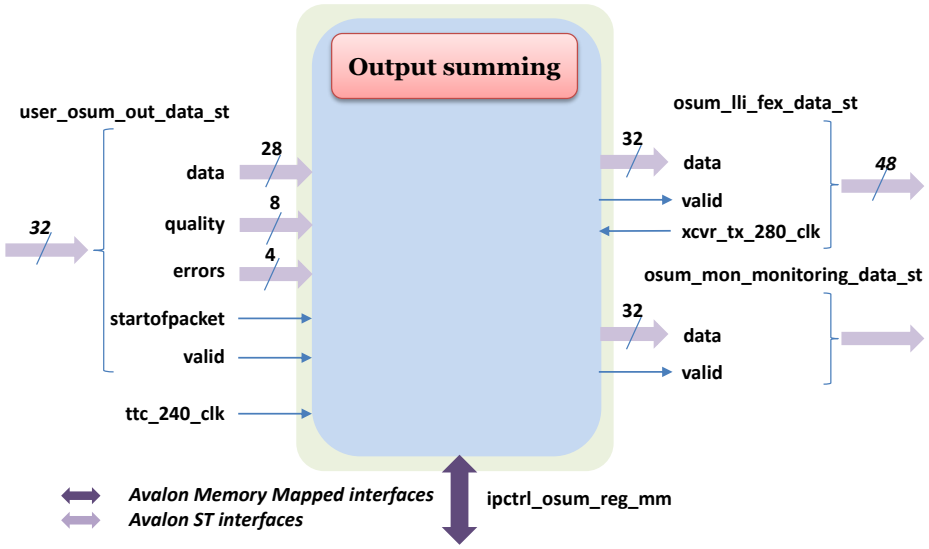


Figure 21: Output summing block diagram

6.2 Description

The data flow of the *output summing* block is shown in Fig. 22. The width of the arrows on the diagram are roughly proportional to the data streams.

Up to 12 $E_{\mathrm{T}}^{\text{Super Cell}}$ are calculated for each TT in the *user code* block. For most TTs only 10 or less data words, out of the 12, are valid data.

For those TTs with all 12 valid data words $E_{\mathrm{T}}^{\text{Super Cell}}$ the special procedure is enabled in the *output summing* block to combine two pairs of Super Cells to two pseudo-Super Cells. This procedure is performed in the input *12 ⇒ 10 adapter* and still needs to be defined. The simple sum can be used as the first approximation for this adapter.

After the adapter the data go to the eFEX packager (where the precision is fixed and headers and trailers are appended) and to the first stage of summing where the sums over TTs are performed. To configure the *output summing* block which Super Cells are valid and should be included in the output and to be summed, special registers are included for *output summing* block. The sums from the first summing stage go to the jFEX packager and to the second stage of summing where sums over 0.2 x 0.2 $\eta - \phi$ regions are performed. These sums go to the gFEX packager.

While the input data come aligned on ttc_240_clk and the output clock has been chosen to 280 MHz, the data width of the output interfaces are still to be defined. The data transfer to the output clock domain is performed in the packager blocks using FIFOs which are pushed in five or six times out of the six input

clock cycles (during 1 BCID) and poped out five or six times out of the seven output clock cycles (for the 280 MHz output clock). The header and trailers are appened also in the packager blocks.

Packaged data are duplicated for selected channels. The set of duplicated channels and the multiplicity of the duplication is set by specific registers of the *output summing* block. The destination (eFEX, jFEX or gFEX) and area covered by specific optical fibre is set by the registers and the logic of the *selective duplication* block, that can be configured. However the total number of the output FEX lines is limited by the number of microPOD^{TM} Tx connectors.

For monitoring and debugging purpose, any sum, within the *output summing* block can be directed for monitoring. To minimize the number of output lines for monitoring the selection registers are used for choosing the sums of interest.

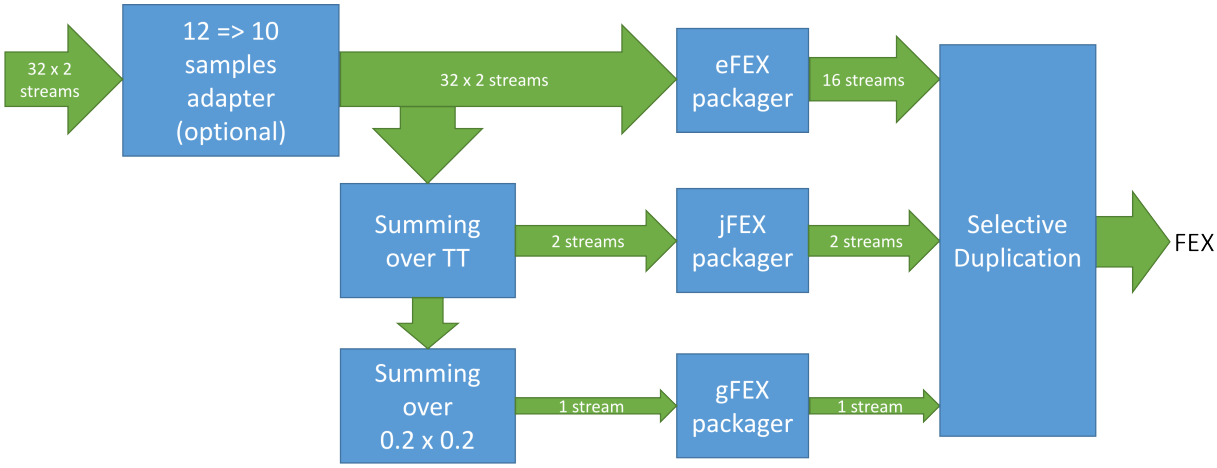


Figure 22: Data flows of the Output summing block

6.3 Estimated latency

The estimated latency on the main path of the *output summing* block for the eFEX stream is less than 0.5 BCs. This time is needed for eFEX data to change the clock domain and to append headers and trailers. Since summing of all TT energies is needed for jFEX and gFEX, the additional latency of 1 BC is incurred for them. Thus the latency for jFEX and gFEX are less than 1.5 BC.

6.4 Interfaces

The Output summing block has four interfaces:

- the input data come over 32 paired streams of **user_osum_out_data_st** interface. It is the Avalon Stream interface;
- the **osum_lli_fex_data_st** interface the output data go to the LLI and further to eFEX. It is the Avalon Stream interface;
- over **osum_mon_data_st** interface the selected FEX data outputs for monitoring. It is the Avalon Stream interface;
- **ipctrl_osum_reg_mm** used to load configuration data to the block. It is the Avalon MM interface.

Full description of the interfaces is presented in Table 11.

Interface	#	Description	
Clocks			
ipctrl_100_clk	1	IP Bus controller 100MHz clock	
ttc_240_clk	1	TTC 240MHz recovered clock	
Data path			
user_osum_out_data_st	32	Processed data block from User-code	
	Signal/Bus	**Width**	**Description**
	data	28	Two 14 bits data words from User-code packed in a 28 bits word
	error	4	CRC error, BCID error for two supercells
	quality	8	Quality information
	startofpacket	1	First word of the packet
	valid	1	Valid data from User-code
FEX data			
osum_lli_fex_data_st	48	Packed FEX data	
	Signal/Bus	**Width**	**Description**
	data	32	FEX data
	valid	1	FEX data is valid
	xcvr_tx_280_clk	1	Transceiver clock
Monitoring data			
osum_mon_monitoring_data_st	1	Monitoring data	
	Signal/Bus	**Width**	**Description**
	data	32	Monitoring data
	valid	1	Monitoring is valid
Registers			
ipctrl_osum_reg_mm	1	Output summing registers	

Table 11: Output summing interfaces description

6.5 Registers

Several configuration register types are used in the *output summing* block.

- registers on whether to activate 12 to 10 Super Cells adapter for specific channel;
- registers on which TT Super Cells are included for total TT energies for jFEX and gFEX. The dummy Super Cells for some TTs with less than 10 valid Super Cells are omitted from summation;
- registers on which channels need duplication;
- registers on which channels go for monitoring

7 TDAQ Readout and Monitoring

7.1 Introduction

The AMC-A10 card has three types of output related to data: output for the calorimeter trigger FEXs, output to TDAQ on receipt of an L1 Accept and output to the monitoring stream for debugging and fast online analysis. ADC data (from each Super Cell) and $E_{\mathrm{T}}^{\text{Super Cell}}$ data must be buffered pending an L1A for the TDAQ readout or pending a Send Monitor Data request for monitoring. The Send Monitor Data request could be an L1A, a TTC channel B command or an internally generated signal.

TDAQ data are sent from the AMC-A10 card via a high speed serial pair either directly to a GBT chip-set located on the Rear Transition Module (RTM) behind the carrier card, or through the carrier card FPGA (that acts as a buffer) and then to the GBT chip-set. The baseline is the latter (using the carrier card FPGA as a buffer) because the signal path lengths are minimized.

Monitoring data are sent from the AMC-A10 via four XAUI pairs to the FPGA located on the carrier card. The monitoring data from the four AMC-A10 of one LDPB are multiplexed on the FPGA located on the carrier card and then sent to the Zone 2 Fabric of the ATCA shelf.

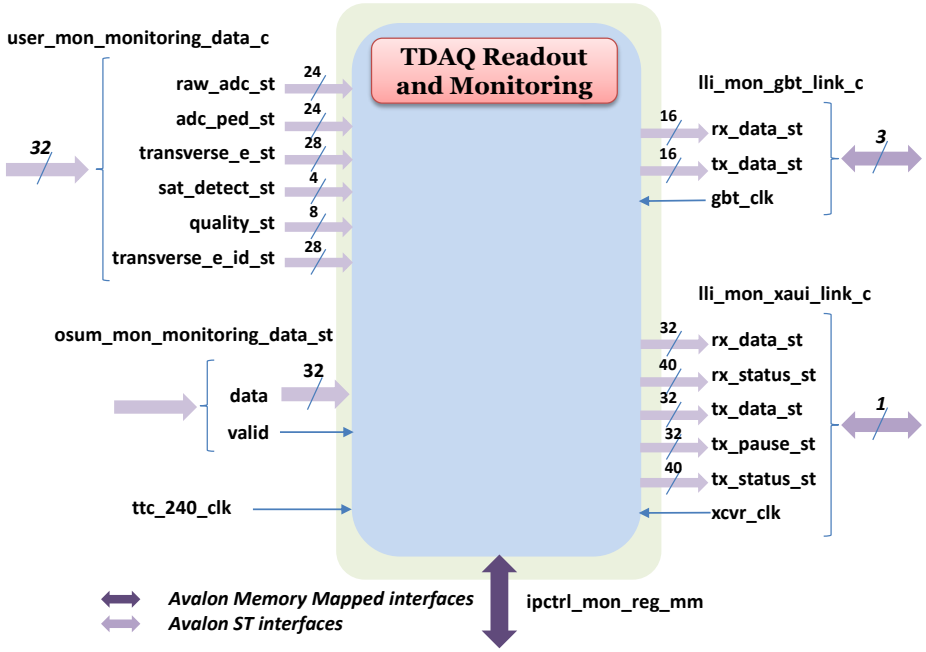


Figure 23: TDAQ Readout and Monitoring block diagram

7.2 Description

7.2.1 Monitor and TDAQ data buffering

Both the incoming ADC data and outputs of the filter ($E_{\mathrm{T}}^{\text{Super Cell}}$ quality bits) results must be buffered prior to their output to TDAQ or the monitor stream.

7.2.1.1 Monitor data buffering

Monitor data is buffered in four stages:

- circular buffer that runs continuously and keeps a moving window of the last 512 words of ADC data, filtered (energy and quality bits) data. There is a dedicated circular buffer for each Super

Cell channel. When a Send Monitor Data signal is received, a contiguous block of N (N typically between 5 and 10) data samples is taken from the circular buffer and pushed on to a dedicated FIFO. The number of samples is the number of BC desired and will depend on which type of Monitor data is being sent. In a trial implementation for a Xilinx Virtex-7, the circular buffer is 512×36, absorbing 3 12 bit values every cycle. This corresponds to a depth of (512-2) $\times$ 25 ns = 12.75 μs

- Data is packed into a second stage FIFO as 64 bit words in preparation for networking. Each channel has its own dedicated second stage FIFO. In a trial implementation for a Virtex-7, a shift register LUT (SRL) was used with a depth of 16.

- When the second stage FIFO has received all the data following the Send Monitor Data signal, it triggers a state machine that merges the results from multiple Super Cell channels into a third stage FIFO. There are five of these third stage FIFOs for the 320 Super Cells of information and each of the 5 FIFOs corresponds to data in a jumbo UDP packet. In a trial implementation for a Virtex-7, the FIFO size was 512×64.

- The UDP header information is contained in a separate BRAM. Finally, the UDP header and FIFO data are merged and sent out sequentially as 5 separate UDP jumbo packets. The jumbo packet format is shown in the Figure 24.

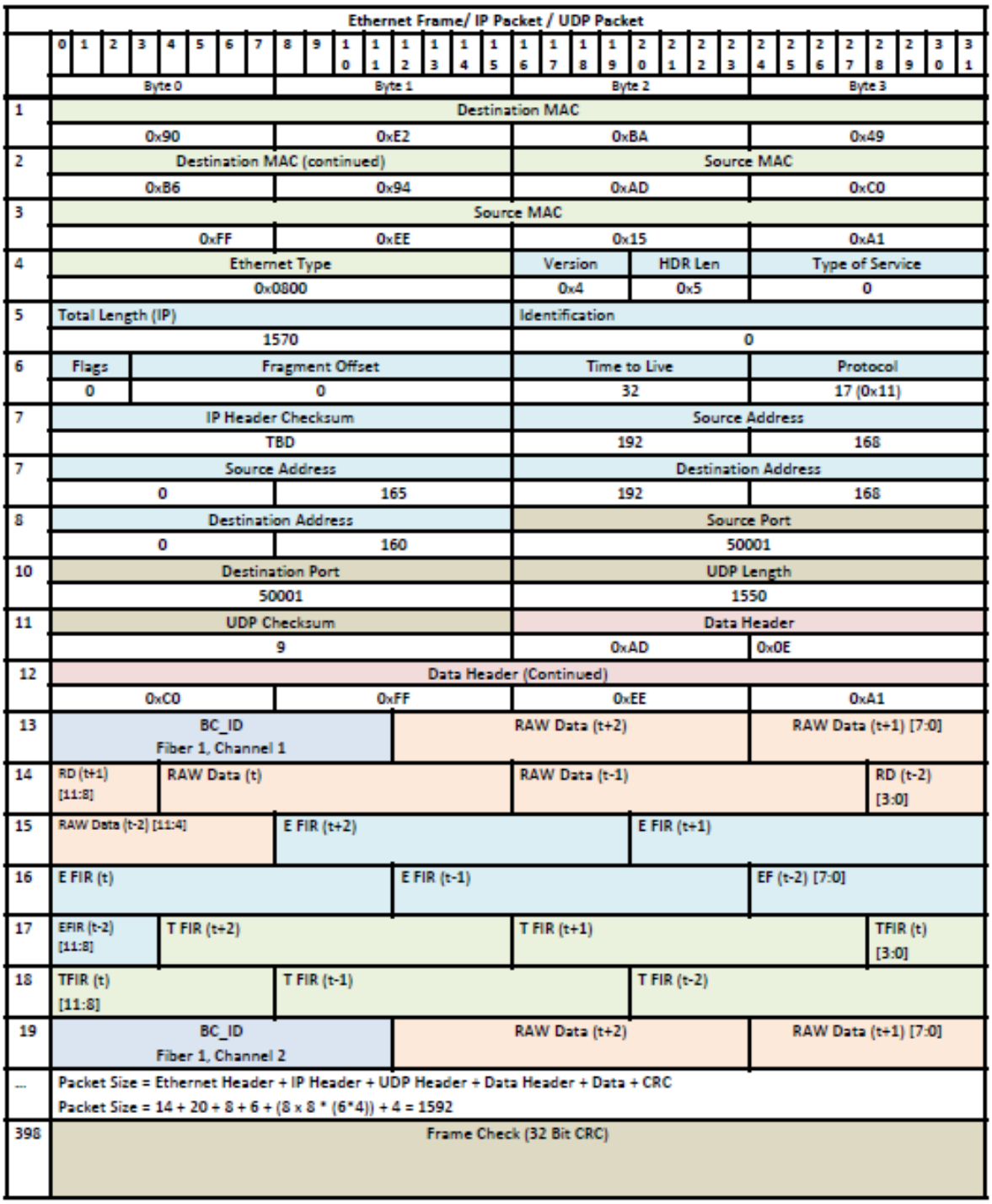


Figure 24: UDP packet format

A second type of monitor data that has been considered is the *oscilloscope* format. In this case a continuous flow of data (every BC) is sent out for debugging. Whether or not to send the associated $E_{\mathrm{T}}^{\text{Super Cell}}$ is still to be decided. The amount of data to be sent must also be decided. Ideally the number of jumbo UDP packets sent would be the same as for the previous case.

7.2.1.2 TDAQ data buffering

The TDAQ data is buffered in a similar manner as the monitor data:

- The first stage is a circular buffer that runs continuously and keeps a moving window of the last 512 words of ADC data and filter results data. There is a dedicated circular buffer for each Super Cell channel.
- When an L1A Send signal is received, a contiguous block of N data samples is taken from the circular buffer and pushed on to a dedicated second stage FIFO.

  The L1A received must also be buffered to fulfill the TDAQ requirements on consecutive L1A.
- When the second stage FIFO has received all the data following the L1A, it triggers a state machine that merges the results from multiple Super Cell channels into a third stage FIFO. This FIFO then serves as input to the GBT serialization block. It is likely there will be two third stage FIFOs, with one being used to store the next L1A data being sent while the other is readout into the GBT serialization block.

7.2.2 TDAQ Data

TDAQ data is sent to the ATLAS TDAQ system upon L1A. The input is from a circular buffer previously described. The output is a serial pair, encoded in GBT format, which is used as input to the GBT chip-set that resides on the RTM.

In Phase 1, the L1 trigger latency is 2.5 μs and the maximum L1 rate is 100 kHz. In Phase 1, data is sent in response to an L1A.

In Phase 2, the L0 trigger latency is 6 μs (with headroom to go to 10 μs) and the maximum L0 rate is 1 MHz. The L1 trigger latency is 30 μs (with headroom to go to 60 μs) and the maximum L1 rate is 400 kHz.

The firmware to be developed will assume Phase 1 conditions. However, much larger circular buffers corresponding to Phase 2 conditions should be investigated as to their impact on FPGA resources. TDAQ data from each AMC-A10 is sent to a GBT, where it is subsequently sent to FELIX and then on to the ATLAS TDAQ system. Several TDAQ data formats must be accommodated.

Several use cases have been considered:

- In normal operation, the data sent to TDAQ are $E_{\mathrm{T}}^{\text{Super Cell}}$ coded on 10 bits but they can be stored in 16 bit words if needed. The bandwidth is 320 Super Cells (SC) $\times$ 16 bits $\times$ 100 kHz = 0.5 Gbps. This easily fits within the approximately 5 Gbps bandwidth of the GBT protocol used by FELIX. One should also send a few words of header information including the BCID corresponding to the L1A.
- A second TDAQ mode is for calorimeter noise readout. In this mode, both the ADC and $E_{\mathrm{T}}^{\text{Super Cell}}$ data are transmitted. This data is sent for a total of 10 BCs, five before and five after the BC corresponding to the L1A (RNDM).

  These data are collected at 1 Hz L1A rate (RNDM) so the amount of data to be transmitted is 320 SC $\times$ 2 (ADC and $E_{\mathrm{T}}^{\text{Super Cell}}$) $\times$ 16 bits $\times$ 10 BCs $\times$ 1 Hz = 0.1 Gbps. This rate fits easily within the GBT bandwidth.
- A third TDAQ mode is used to compare the filtered transverse energy calculations to those determined using the readout of individual calorimeter cells and offline filtering. One can imagine using heavily prescaled jet triggers in special runs for L1A for this case. The data to be sent would be both ADC, $E_{\mathrm{T}}^{\text{Super Cell}}$ and quality bits data. The bandwidth would be slightly larger than the second TDAQ mode.

7.2.3 GBT Transceiver

GBT (GigaBit Transceiver) firmware is used to transmit TDAQ data to FELIX and to receive TTC and other control information from the same. The firmware is designed to communicate with the GBTx ASIC and Versatile link. An HDL library and example firmware have been provided by the CERN GBT-FPGA Firmware group for a limited number of target FPGAs and IDE versions. This provides a starting point for most applications but we will have to modify the code and migrate to the Arria^{TM}10 and appropriate QUARTUS^{TM} version.

For our application we will likely use the "GBT-frame" encoding frame and the Standard latency. In this mode, there is 80 bits of User Data and 40 bits of reserved data (header, slow control and forward error correction). Thus 120 bits of data are transmitted at 40 MHz. The GBT core firmware generates or receives a 20- or 40- bit word that must be then integrated with the Arria^{TM}10 transceivers.

TDAQ data are placed in the 80-bit User Data words for transmission. The format of the TTC data that are received via the GBT is still to be defined. The recovered clock from the receive firmware becomes the BC clock for the AMC-A10 card.

7.2.4 Monitor Data

The function of the monitoring stream is to provide validation of the data processing on AMC-A10. The Monitor data is meant to be sent over the 10 GbE ATCA Zone 2 fabric though an ATCA switch to a host PC or farm. It is not part of the usual ATLAS TDAQ system. The input is from a circular buffer described above. The output is to four XAUI lanes that connect the AMC-A10 FPGA to the carrier card FPGA. The transmitted data are in the form of jumbo UDP packets. The aggregate bandwidth from the ATCA crate is $\approx$ 10 Gbps.

- In normal operation, monitor data is sent to a host PC or farm. The stimulus for sending data could a prescaled L1A or a RNDM one. One monitoring mode would send ADC data, $E_{\mathrm{T}}^{\text{Super Cell}}$ and the local BCID in response to the stimulus. UDP and data header information would be sent as well. The data to be sent might correspond to 5 BCs, two BCs before and two after the peak BC. In this case the bandwidth would be 320 SC $\times$ 3 (ADC, $E_{\mathrm{T}}^{\text{Super Cell}}$, Q) $\times$ 5 (BCs) $\times$ 16 bits x S, where S is the stimulus rate. An S of 10 Hz gives a bandwidth of 0.76 Gbps. While this is below the 10 GbE bandwidth, recall that four AMC-A10 feed the Carrier Card FPGA/Zone 2 fabric and that there are multiple carrier cards in the ATCA crate.

- Another mode of operation would send an oscilloscope type of monitor data output. In this case, data from 32 BCs would be sent for a given Super Cell. The data to be sent might be 80 SC $\times$ 2 (ADC and $E_{\mathrm{T}}^{\text{Super Cell}}$) $\times$ 16 bits $\times$ 32 BCs $\times$ S, where S is the stimulus rate. An S of 10 Hz gives a bandwidth of 0.82 Gbps.

All, or at least many, of the use cases need to be specified in writing by the LAr group before the final firmware is written. One consideration is packing the data into jumbo UDP packets. This is driven by the number of BCs to be sent, the different forms of data to be sent and the number of Super Cell. An additional selection criteria of $E_{\mathrm{T}}^{\text{Super Cell}} > E_T$(threshold) has also been proposed for the data. While this can be done it is an additional complication in the firmware in terms of packing data in UDP packets.

7.3 Estimated latency

This part of the design is not in the critical data path, therefore the latency of this block is not relevant.

7.4 Interfaces

In the case of the monitor data, the input to the Monitor firmware block are the outputs from three circular buffers. The three circular buffers hold ADC and filtered data for each Super Cell. The circular buffers are built from BRAM elements.

The output from the monitor block is a 64-bit FIFO. Additionally there is a fixed UDP preamble (Ethernet frame, IP header and UDP header information that is stored in BRAM and sent before the actual Monitor data. A UDP preamble block provides this data. The preample and Monitor data are subsequently merged and sent to the 10 Gbe MAC. The data is sent to the 10 GbE MAC via the AV_mm_bus.

In the case of the TDAQ data, the input to the TDAQ firmware block are the outputs from three circular buffers. The three circular buffers hold raw SC data and filtered energy and time data. Alternatively, they could hold raw SC data, filtered energy data and quality data. The circular buffers are built from BRAM elements.

The output from the TDAQ block is an 84-bit FIFO. This FIFO is subsequently read by the GBT firmware block at 40 MHz.

7.5 Registers

Some of the registers to be implemented in the TDAQ Readout and Monitoring block are given in the Table 12. They were used in a Xilinx Virtex-7 implementation of the Monitoring block. There will be a similar set of registers for the TDAQ Readout block, some of which are given in the table.

Register name	Type	Length	Description
mon_preamble_storage	R/W	32	10 GbE preamble storage
mon_test_storage	R/W	128	10 GbE test packet storage
mon_preamble_size	R/W	1	10 GbE preamble size
mon_test_size	R/W	1	10 GbE test packet size
mon_packet_channel_enable	R/W	1	10 GbE packet channel enable bits
mon_test_packet_control	R/W	1	10 GbE test packet control
mon_xaui_status	R/W	1	10 GbE XAUI/MAC status
mon_mac_tx_status	R/W	1	10 GbE MAC TX status
mon_packet_bytes	R/W	1	10 GbE packet size bytes
mon_status_control	R/W	1	10 GbE status and control
mon_id_1	R/W	1	10 GbE ID 2
mon_id_2	R/W	1	10 GbE ID 1
mon_error_reg_1	R/W	1	10 GbE monitor error register 1
mon_error_reg_2	R/W	1	10 GbE monitor error register 2
mon_gbt_preamble_storage	R/W	1	GBT header (if needed)
mon_gbt_test_storage	R/W	128	GBT test data
mon_gbt_status_control	R/W	1	GBT status and control
mon_gbt_error_reg_1	R/W	1	GBT error register 1

Table 12: TDAQ Readout and Monitoring block registers description

Interface	**#**	**Description**
Clocks		
ipctrl_100_clk	1	IP Bus controller 100MHz clock

Table 13: TDAQ monitoring interfaces description (part)

Interface	#	Description	
ttc_240_clk	1	TTC 240MHz recovered clock	
GBT Link			
lli_mon_gbt_link_c	3	GBT Monitoring link	
gbt_120_clk	1	Transceiver clock	
rx_data_st	1	RX data bus	
	Signal/Bus	**Width**	**Description**
	data	16	TBD
	ready	1	TBD
	valid	1	TBD
tx_data_st	1	TX data bus	
	Signal/Bus	**Width**	**Description**
	data	16	TBD
	ready	1	TBD
	valid	1	TBD
Memory			
mon_lli_ddr_c	1	DDR3 memory bus	
bank_mm	2	DDR3 memory bank	
ddr_200_clk	1	DDR3 200MHz clock	
Monitoring			
user_mon_monitoring_data_c	32	Monitoring data	
adc_ped_st	1	ADC data after substracting pedestal. 12 bits per SC	
	Signal/Bus	**Width**	**Description**
	data	24	ADC data without pedestal data
	startofpacket	1	First cell
	valid	1	Valid data from User-code
quality_st	1	Quality from combine block. 4 bits per SC	
	Signal/Bus	**Width**	**Description**
	data	8	Quality data
	startofpacket	1	First cell
	valid	1	Valid data from User-code
raw_adc_st	1	ADC data before substracting pedestal. 12 bits per SC	
	Signal/Bus	**Width**	**Description**
	data	24	ADC data
	startofpacket	1	First cell
	valid	1	Valid data from User-code
sat_detect_st	1	Output of the saturation detection. 2 bits per SC	
	Signal/Bus	**Width**	**Description**
	data	4	Saturation detection data
	startofpacket	1	First cell
	valid	1	Valid data from User-code
transverse_e_id_st	1	Transverse energy E_T from combine block. 14 bits per SC	
	Signal/Bus	**Width**	**Description**
	data	28	E_T data
	startofpacket	1	First cell

Table 13: TDAQ monitoring interfaces description (part)

Interface	#	Description	
	valid	1	Valid data from User-code
transverse_e_st	1	Transverse energy E_T from filtering block. 14 bits per SC	
	Signal/Bus	**Width**	**Description**
	data	28	E_T data
	startofpacket	1	First cell
	valid	1	Valid data from User-code
Monitoring data			
osum_mon_monitoring_data_st	1	Monitoring data	
	Signal/Bus	**Width**	**Description**
	data	32	Monitoring data
	valid	1	Monitoring is valid
Registers			
ipctrl_mon_reg_mm	1	Monitoring registers	
XAUI Link			
lli_mon_xaui_link_c	1	XAUI Monitoring link	
rx_data_st	1	RX data bus	
	Signal/Bus	**Width**	**Description**
	data	32	TBD
	empty	2	TBD
	endofpacket	1	TBD
	error	6	TBD
	ready	1	TBD
	startofpacket	1	TBD
rx_status_st	1	RX status bus	
	Signal/Bus	**Width**	**Description**
	data	40	TBD
	error	7	TBD
	valid	1	TBD
tx_data_st	1	TX data bus	
	Signal/Bus	**Width**	**Description**
	data	32	TBD
	empty	2	TBD
	endofpacket	1	TBD
	error	2	TBD
	startofpacket	1	TBD
	valid	1	TBD
tx_pause_st	1	TX pause bus	
	Signal/Bus	**Width**	**Description**
	data	32	TBD
tx_status_st	1	TX status bus	
	Signal/Bus	**Width**	**Description**
	data	40	TBD
	error	7	TBD
	valid	1	TBD
xaui_312_5_clk	1	Transceiver clock	

Table 13: TDAQ monitoring interfaces description (part)

Interface	#	Description

Table 13: TDAQ monitoring interfaces description

8 Slow Control

8.1 Introduction

The slow control interface connects the firmware blocks to the outer world to allow the user to load or change all the configuration parameters of the system. Furthermore, it monitors the status of the system and implements the integration into the ATLAS Detector Control System (DCS).

The transport mechanism for the slow control interface is the IPbus protocol [23] over Ethernet. Since our colleagues from the Compact Muon Solenoid (CMS) experiment already developed an IPbus controller firmware it is proposed to use this firmware. The IPbus controler is is available from the CACTUS package [26]. This IPbus controller interfaces an IPbus over Ethernet to an adjustable, synchronous System on Chip (SoC) bus in accordance to the Wishbone specifications [21]. Since the LAr firmware working group has further decided to use Avalon interfaces for all connections, a Wishbone to Avalon Memory Mapped interface adapter needs to be implemented.

8.2 IPbus Protocol and CACTUS firmware

The proposed IPbus Controller can be found at Code Archive for the The UpgradeS [26].

It is intended to modify the CACTUS firmware as less as possible to allow easy integration of future developments done by the CACTUS group. Therefore, a design with interface adapters is chosen. The adapter between Wishbone and the Avalon Memory Map interface is labeled with wb2amm in the bock diagram of Figure 26. The archive also contains a documentation of version 2.0 of the IPbus protocol

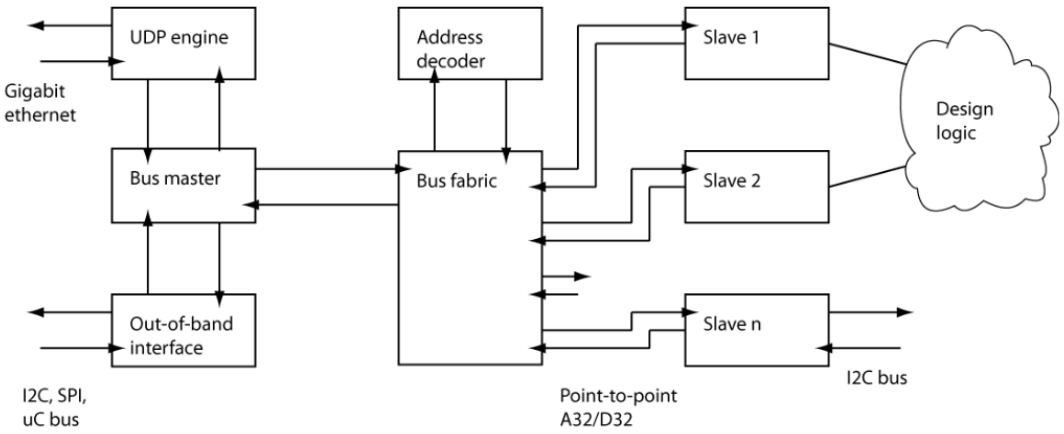


Figure 25: Schematic view of the SoC bus topology. The bus is implemented as a set of point-to-point signals. The figure is taken from the CACTUS firmware documentation [25]

[23]. The firmware directory [24] is located at CERN and contains implementations for different XILINX development boards and some example slave designs which could serve as a basis for our modules. Figure 25 shows a schematic view of the SoC bus topology. The bus is implemented as a set of point-to-point signals. The figure is taken from the CACTUS firmware documentation [25].

8.3 ATLAS Detector Control System

The general organization of the Atlas Detector Control System (DCS) is described in [22]. The Finite State Machine (FSM) mechanism is described in chapter 7.1 therein. It is a framework used to model the behavior of a system by means of limited number of states, transitions between states, actions and events. The DCS is designed to

i) be flexible and platform-independent,

ii) be fault-tolerant and allow remote diagnostics,

iii) keep the dependence of low-level control procedures on external services such as computer networks as small as possible.

There is no IPbus port to the ATLAS DCS. Therefore, a computer should serve as the interface between the slow control firmware and the ATLAS DCS. The CACTUS team has also developed an IPbus software suite, called uHAL [26], to simplify the development of control software. The control software should be integrated into the LAr TDAQ software to have only one common software to be maintained. Since the CACTUS firmware is designed as a master-slave architecture, the software needs to poll the firmware modules to monitor their states. The connection to the ATLAS DCS system needs to be developed with the ATLAS DCS group and will be described in a separate document. The control software should observe the ATLAS Finite State Machine (FSM) states and react on changes if necessary by setting the control registers of the firmware modules to initiate the corresponding behavior. Special attention should be taken to the implementation of a reset procedure to guarantee signal integrity.

In summary, the connection between the AMC-A10s and the ATLAS DCS is done via a software control program which interfaces the IPbus protocol to a DCS protocol.

8.4 Interfaces

In order to connect the AMC-A10 firmware modules by means of a Avalon MM interface to the CACTUS control ports, an output interface adapter is required as depicted in figure 26. The Gigabit ethernet connection of the CACTUS firmware follows the 8 data bit AXI4 streaming standard. Most probably, a second interface adapter needs to be implemented to connect the `lli_ipctrl_gbe_link_c` custom interfaces to the AXI4 standard.

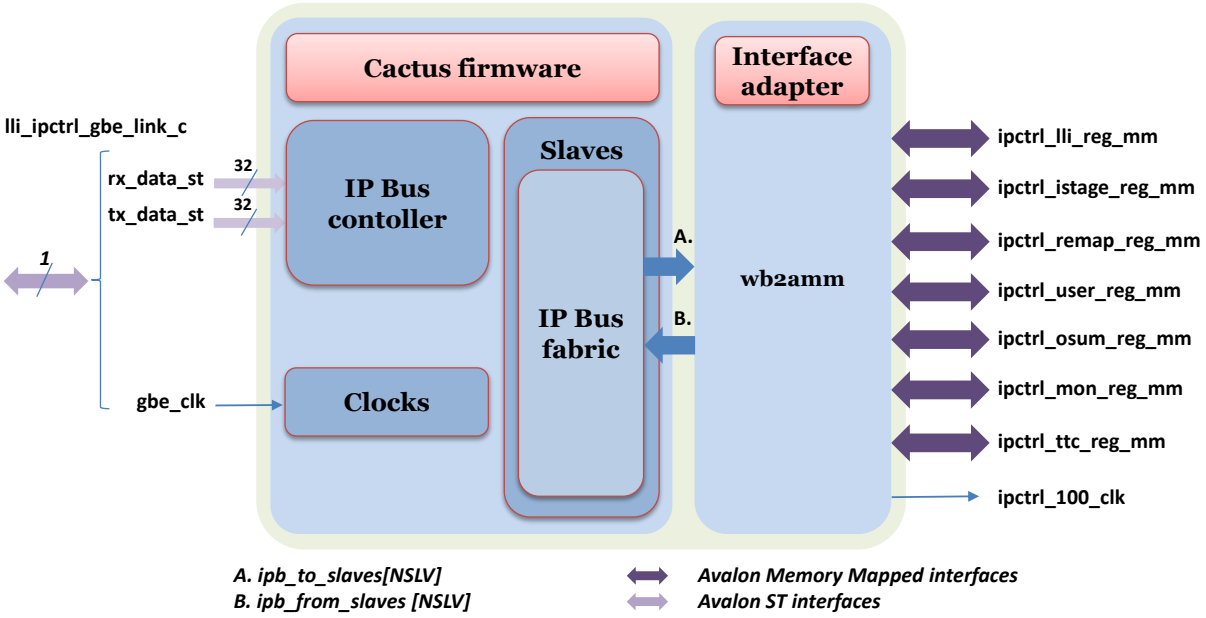


Figure 26: Slow control block diagram. The left block represents the CACTUS firmware with its main modules ipbus_ctrl, slaves, ipbus_fabric, and clocks. The Wishbone interface arrays labeled A. and B. have the names used by the CACTUS firmware where NSLV is the number of slaves. The slaves are defined in the block slaves and the address decoding is done in the block ipbus_fabric utilizing the auto generated - according to the users definitions - VHDL code ip_bus_addr_decode.vhd.

All interfaces to the slow control firmware block are listed in table 14. In case of the Ethernet link, the interface is described in detail with the corresponding signal names.

Interface	**#**	**Description**	
Clocks			
ipctrl_100_clk	1	IP Bus controller 100MHz clock	
Gigabit Ethernet Link			
lli_ipctrl_gbe_link_c	1	Gigabit Ethernet link	
gbe_100_clk	1	Transceiver clock	
rx_data_st	1	RX data bus	
	Signal/Bus	**Width**	**Description**
	data	32	TBD
	empty	2	TBD
	endofpacket	1	TBD
	error	6	TBD
	ready	1	TBD
	startofpacket	1	TBD
	valid	1	TBD
tx_data_st	1	TX data bus	
	Signal/Bus	**Width**	**Description**
	data	32	TBD
	empty	2	TBD
	endofpacket	1	TBD
	error	1	TBD

Table 14: Slow control interfaces description (part)

Interface		#	Description	
		ready	1	TBD
		startofpacket	1	TBD
		valid	1	TBD
Registers				
ipctrl_istage_reg_mm		1	Input stage registers	
ipctrl_lli_reg_mm		1	Low-level interface registers	
ipctrl_mon_reg_mm		1	Monitoring registers	
ipctrl_osum_reg_mm		1	Output summing registers	
ipctrl_remap_reg_mm		1	Configurable remapping registers	
ipctrl_ttc_reg_mm		1	TTC registers	
ipctrl_user_reg_mm		1	User code registers	

Table 14: Slow control interfaces description

8.5 Registers

Most probable, it is not necessary to have any registers for the slow control itself.

9 TTC

9.1 Introduction

The *ttc* block receives the LHC TTC information on either an LVDS or GBT link, this needs to be clarified. It recovers the LHC clock and provide all other blocks a synchronous clock. Informations such as BCID, Trigger-Type, Level-1 Accept are also provided. The synopsis of the *ttc* block is shown on figure 27.

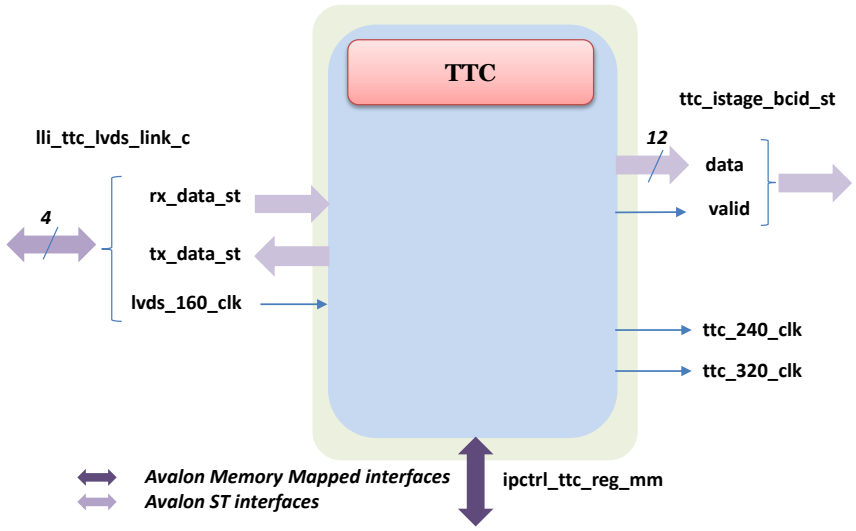


Figure 27: *ttc* block diagram

9.2 Description

TBD

9.3 Estimated latency

The *ttc* block is not in the main data path hence does not introduce any latency.

9.4 Interfaces

Full description of the interfaces is presented in table 15.

Interface	#	Description	
Clocks			
ipctrl_100_clk	1	IP Bus controller 100MHz clock	
ttc_240_clk	1	TTC 240MHz recovered clock	
ttc_320_clk	1	TTC 320MHz recovered clock	
LVDS Link			
lli_ttc_lvds_link_c	4	LVDS link	
lvds_160_clk	1	Transceiver clock	
rx_data_st	1	RX data bus	
	Signal/Bus	**Width**	**Description**

Table 15: TTC interfaces description (part)

Interface		#	Description	
		data	16	TBD
		valid	1	TBD
	tx_data_st	1	TX data bus	
		Signal/Bus	**Width**	**Description**
		data	16	TBD
		valid	1	TBD
Registers				
ipctrl_ttc_reg_mm		1	TTC registers	
TTC data				
ttc_istage_bcid_st		1	Current BCID value provided by the TTC receiver	
		Signal/Bus	**Width**	**Description**
		data	12	Received BCID
		valid	1	BCID is valid

Table 15: TTC interfaces description

9.5 Registers

TBD

10 Code Management

10.1 Naming conventions

Some naming conventions apply to the interface and the signal names of the firmware. The naming conventions are intended to improve the readability and the maintainability of the code.

10.1.1 Interface naming convention

The port names of the modules are made out of 4 parts separated by an underline character:

<master block>_<slave block>_<interface name>_<interface type><[cardinality]>

The meaning of the different parts are described in table 16. There can be no spaces in the names nor double underlines.

Part of the name	Description
master block	Short name of the block which is the master of this interface, refer to table 17. If the master block is not specified, the signal/interface is a global signal/interface, usually a global reset or clock, i.e. ttc_240_clk is a clock generated by the *low level interface* that connects to multiple slaves.
slave block	Short name of the block which is the slave of this interface, refer to table 17. If the slave block is not specified, the signal/interface is connected to multiple slaves, typically clocks.
interface name	Name describing the interface and its purpose.
interface type	Short name describing the type of interface, refer to table 18.
cardinality	Number of interfaces to instantiate

Table 16: Building parts for high level interface names

Each block has a abbreviation which is listed in table 17.

High-level block	Block short name
low level interface	lli
input stage	istage
configurable remapping	remap
user code	user
output summing	osum
tdaq monitoring	mon
ipbus controller	ipctrl

Table 17: Abbreviations for the high level blocks

Each type of interface is identified by an abbreviation shown in table 18.

Interface type	Short name
Avalon Streaming interface	st
Avalon Memory Mapped interface	mm
Custom interface	c
Signal	s
Clock	clk

Table 18: Abbreviations for the interface type

An example of an interface definition is `lli_istage_ltdb_data_c[48]`. It represents a custom interface between the *low level interface* and the *input stage* for transferring data from the LTDB. There are 48 instances of this interface.

10.1.2 Signals within an interface

The signals within an interface should be in accordance with the the following convention. The name is made out of two parts separated by an underline character:

<signal name>_<signal direction><[cardinality]>

Table 19 lists the description of the different parts of the signal name.

Part of the name	Description	
signal name	Name describing the signal and its purpose	
signal direction	signal direction	short name
	Signal is driven from a source	o
	Signal is received by a sink	i
cardinality	Number of signals to instantiate	

Table 19: Building parts for signal names

Two examples for signal names are `data_o[16]` and `ready_i`. In order to assign these signals within an interface utilizing a record in the type definitions one will get e.g. `lli_istage_ltdb_data_st[48].data_o[16]` or `lli_istage_ltdb_data_st[48].ready_i`, respectively. Where `lli_istage_ltdb_data_st[48].data_o[16]` is the 16th bit of data of the 48th instance of kind Avalon Streaming used to transfer the LTDB data from the *low level interface* to the *input stage* and `lli_istage_ltdb_data_st[48].ready_i` is the ready signal used by the *input stage* to inform the *low level interface* that it can or cannot receive LTDB data at present.

10.2 GIT Repository

All the code related to the AMC-A10 firmware is stored on a GIT repository located in CERN:

- `https://git.cern.ch/web/atlas-lar-ldpb-firmware.git`

This allows for all contributors to have access to the same code wherever they are.

A dedicated simulation/compilation environment has been built in order to have a defined process common to everyone. It is *Makefile* and *Python* based, originally made from *hdl – make* project [27]. The environment takes care automatically of all dependencies between source files and is compatible with Modelsim/Questa and Quartus software for both Windows and Linux machines.

For more information about the working environment and how to simulate/compile the design, please refer to [28].

10.3 High-level interfaces and code skeleton

In the GIT repository described in section 10.2, is stored a python file used to describe all high-level interfaces. All interfaces names are according to the convention described in section 10.1.

The table is used to generate all the interfaces tables used in the present document. Therefore all blocks are coherent between each other and chances of mistakes are less. A python script in the GIT environment has been developed to generate the Latex code required for the tables.

The table is also used to generate (with the same script) a code skeleton where VHDL files for each modules are automatically generated:

- VHDL interfaces definition
- VHDL component definition
- VHDL entity definition
- VHDL architecture definition, to be used as a base for development
- VHDL testbench to simulate the module

Those files should be used by each developer as a base skeleton. It should also reduce the number of mistakes made at the high-level interfaces.

List of Figures

List of Tables

References

[1] Liquid Argon Calorimeter Phase-I upgrade TDR: https://cds.cern.ch/record/1602230?ln=en

[2] Preliminary Design Review of AMC / Carrier Board I/O Interfaces: https://indico.cern.ch/event/376479/

[3] Xilinx Virtex 7 documentation: http://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html

[4] Georges Aad, Murrough Landon and Michael Begel. Presentation on the FE-BE-L1Calo mapping during the ATLAS upgrade week 22.04.2015: https://indico.cern.ch/event/387385/

[5] Avalon Interface: http://www.altera.com/literature/manual/mnl_avalon_spec.pdf?GSA_pos=4&WT.oss_r=1&WT.oss=avalon%20interface

[6] AVAGO uPOD receiver: http://www.avagotech.com/pages/en/fiber_optics/parallel_optics/12-channel_parallel_optics/afbr-78d2sz/

[7] AVAGO uPOD transmitter: http://www.avagotech.com/pages/en/fiber_optics/parallel_optics/12-channel_parallel_optics/afbr-77d2sz/

[8] FPGA ARRIA10 documentation: http://www.altera.com/literature/lit-arria-10.jsp?ln=devices_fpga&l3=Midrange%20FPGAs-Arria%2010&l4=Documentation

[9] ARRIA10 DDR3 EMIF: http://www.altera.com/literature/hb/external-memory/emi_fd_a10_emif.pdf

[10] Triple speed Ethernet ALTERA IP: http://www.altera.com/support/ip/interface-protocols/ips-inp-tse.html?GSA_pos=3&WT.oss_r=1&WT.oss=triple%20speed%20ethernet

[11] 10GbE MAC ALTERA IP: http://www.altera.com/support/ip/interface-protocols/ips-inp-10gbe.html?GSA_pos=2&WT.oss_r=1&WT.oss=10gbe

[12] GBT FPGA project: https://espace.cern.ch/GBT-Project/GBT-FPGA/default.aspx

[13] A line code with quick-resynchronization capability and low latency for the optical data links of LHC experiments": 2014 JINST 9 P07020

[14] W.E. Cleland and E.G. Stern, Signal Processing Considerations for Liquid Ionization Calorimeters in a High Rate Environment, Nucl. Instr. and Meth. A338 (1994) 467.

[15] S.V. Vaseghi, Advanced Signal Processing and Digital Noise Reduction, Wiley, New York, 1996.

[16] S. Hisajima et al. Presentation at LAr Phase-I upgrade BE meeting, 4th Nov, 2014. https://indico.cern.ch/event/349580/contribution/1/material/slides/0.pdf

[17] J. Philipp Grohs, Presentation at LAr sLHC simulation Meeting, 11th April, 2014. https://indico.cern.ch/event/350451/contribution/1/material/slides/0.pdf

[18] ALTERA corporation, AN639: Inferring Stratix V DSP Blocks for FIR Filtering Applications, 2011. https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/an/an639.pdf

[19] J. P. Grohs, S. Startz, AREUS: ATLAS Readout Electronics Upgrade Simulation, ATL-LARG-INT-2014-003.

[20] J. Philipp Grohs, Presentation at LAr upgrade simulation Meeting - ATLAS upgrade week, 21th April, 2015. https://indico.cern.ch/event/387375/

[21] Wishbone specifications: http://cdn.opencores.org/downloads/wbspec_b4.pdf

[22] A. Barriuso Poy, *et al.*, The detector control system of the ATLAS experiment, JINST **3** (2008) P05006. http://iopscience.iop.org/1748-0221/3/05/P05006/pdf/1748-0221_3_05_P05006.pdf

[23] The IPbus Protocol, https://svnweb.cern.ch/trac/cactus/browser/trunk/doc/ipbus_protocol_v2_0.pdf

[24] Link to the CACTUS IP bus controller https://svnweb.cern.ch/trac/cactus/browser/tags/ipbus_fw/ipbus_2_0_v1/firmware

[25] Notes on Firmware Implementation of an IPbus SoC Bus, https://svnweb.cern.ch/trac/cactus/browser/trunk/doc/IPbus_firmware_notes.pdf

[26] Link to the CACTUS wiki https://svnweb.cern.ch/trac/cactus

[27] HDL Make project: http://www.ohwr.org/projects/hdl-make/

[28] GIT Repository working environment documentation: https://git.cern.ch/web/atlas-lar-ldpb-firmware.git/blob_plain/HEAD:/doc/atlas-lar-ldpb-firmware-env.pdf