Argonne NATIONAL LABORATORY

FTK to Level-2 Interface Card User's Guide

-- PRELIMINARY --

December 15, 2016

Version 2.4

Originators: John T. Anderson, Michael Oberling

Table of Contents

1 GENERAL INFORMATION

The FTK to Level-2 Interface Card (FLIC) is an Advanced Telecommunications Computing Architecture (ATCA) module designed as part of the ATLAS Fast TracKer (FTK) to receive data records from Second Stage Boards (SSBs) via optical fiber, decode the packet structure of those records, perform on-the-fly lookup of detector physical geometry information from local RAM buffers, merge the physical geometry data into the SSB tracks, package the merged data into S-Link packets and transmit them to the ReadOut Subsystem (ROS) using SFP optical transceivers on a rear transition module. Each FLIC has eight front-panel SFP fiber optic inputs, one for each of eight "half-SSB" boards, such that two FLICs collect all the data from the FTK system. Each FLIC implements DDR ram buffers and four 10GbE links so that selected records of interest may be copied to ATCA processor blades resident in the same ATCA shelf as the FLIC. A mesh of internal FPGA-to-FPGA serial links allow collection of data from all "half-SSBs" into a single assembled record fragment for transmission to a processor blade over any of the four 10GbE links.

The system specification document for the FLIC defines the specifications of the module. This document contains specific information regarding details of the hardware and firmware design to assist the reader in the use of the module. Statements herein are intended to convey *how* the design achieves the specifications as stated, *not* the specifications themselves. This revision of the document applies only to the **production** FLIC boards. An earlier version of this document provides information about the prototypes.

1.1 ATCA shelf implementation

The ATCA shelf used for the FLIC system contains 6 slots, connected together by a 3x mesh backplane. Two FLIC boards are installed into hub slots 1 and 2. This allows each FLIC, which uses only one mesh of the 3X fabric, to have a private ATCA channel to each of slots 3, 4, 5 and 6. It is envisioned that these slots of the ATCA shelf will be populated by processor boards to enable monitoring of the data flowing through each FLIC.

The FLIC implements a DIMM socket intended to house an Intelligent Platform Management Controller (IPMC) module designed by the Laboratoire d'Annecy-le-Vieux de Physique des Particules (LAPP) for the ATLAS collaboration. This IPMC implements all communication with the Shelf Manager and is the only part of the FLIC that connects to the Base Interface of the ATCA shelf.

1.2 Front Panel

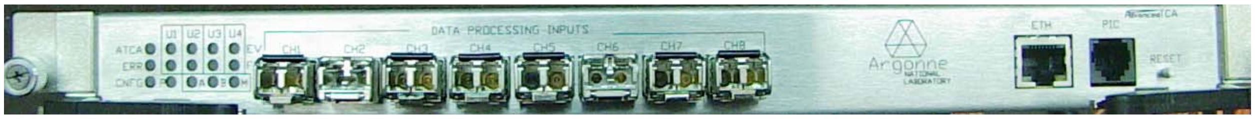


Figure 1 – Front panel of FLIC

The front panel of the FLIC is shown in Figure 1. All LEDs are grouped together in one place left of the SFPs. In the middle, eight SFP optical modulators provide connections to the rest of FTK. To the right of the SFPs, is the Ethernet cable connection that provides a direct interface to the FLIC's microprocessor for test bench operation. At the far right, is the RJ-11 connector used to program the on-board microprocessor.

1.3 Board Layout Overview

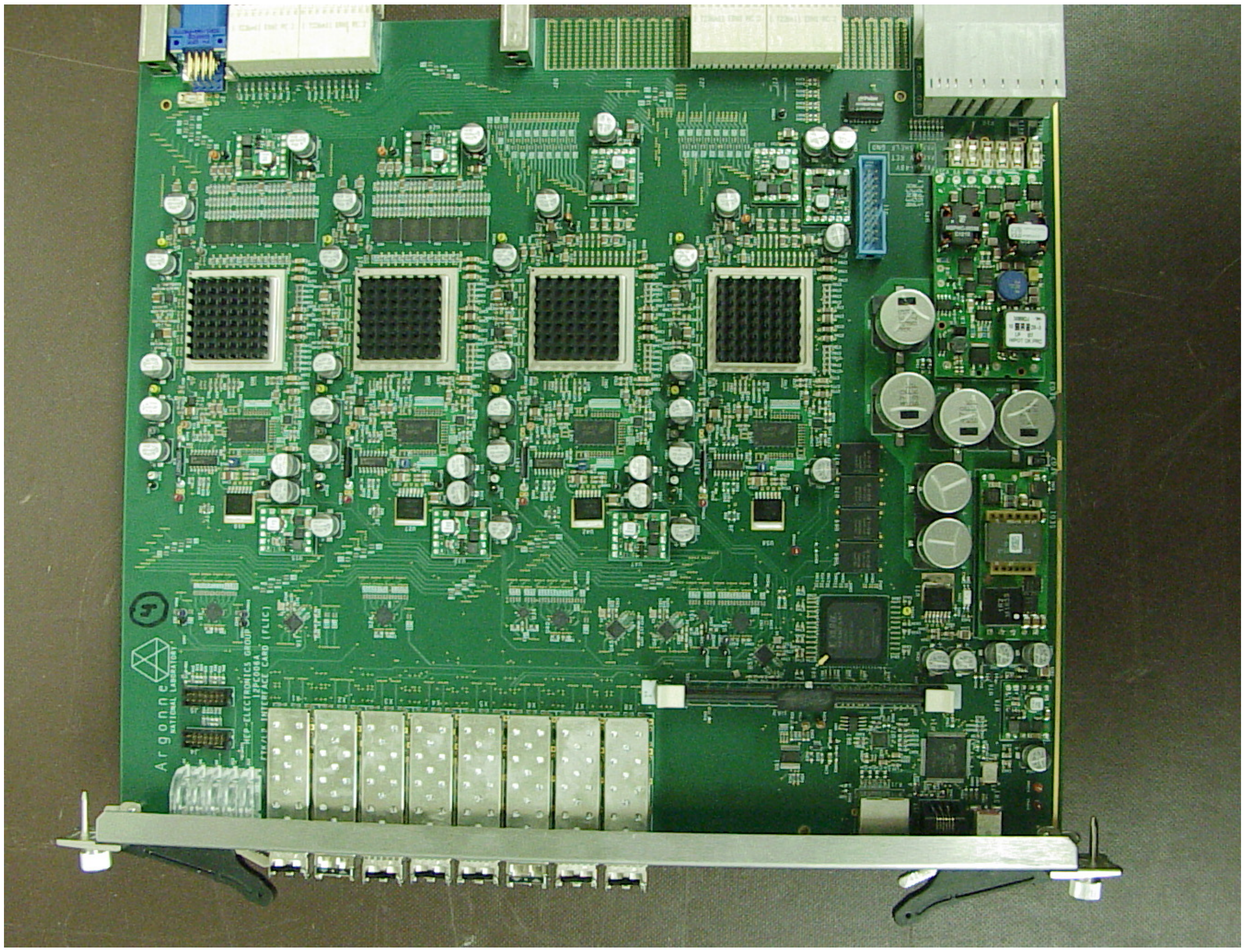


Figure 2 – View of component side of FLIC

Figure 2 shows the component side of the board. The FLIC holds five FPGAs; four Virtex-6 main FPGAs are easily identified by the large heatsinks. These are commonly referred to as "U1" through "U4", respectively from left to right. "U1" and "U2" are connected to the eight SFP optical modulators at the front of the module and also connect to eight more SFPs through the rear transition module (RTM) connector. "U3" and "U4" connect to "U1" and "U2" via an internal mesh of high speed serial links and also connect to four 10GbE channels of the ATCA main fabric.

At the lower right of Figure 2, near the power supply, a smaller Spartan-3 FPGA may be seen; this is referred to as the "Management FPGA". A Microchip PIC microcontroller is also located in this corner of the board that interfaces to the front panel Ethernet connector and the Management FPGA to provide register access and slow control. The large DIMM socket at lower right is for the LAPP IPMC module that connects to the ATCA Shelf Manager, the Management FPGA and the PIC micro.

JTAG connections for the FPGAs and LED light pipes are seen at bottom right. Two JTAG connectors are used to separate the Management FPGA and the main FPGA chains by voltage. A blue debug/programming connector for the IPMC module is located near the ATCA P1 connector.

1.4 Power Distribution

48V power from the ATCA backplane enters via connector P1 through a PIM300 power entry module in accordance with ATCA specifications. A PTEA420033N2AD 48V to 3.3V, 20A converter provides bulk power to the board. Test points on the board allow connection of a benchtop 48V supply to ease diagnosis & repair of the hardware. This bulk converter is enabled by processor control. Only the PIC microcontroller, the Spartan-3 FPGA and the IPMC interface module are powered from ATCA management power. Each of the four main FPGAs have point-of-load power converters to derive required FPGA and DDR memory voltages from the board-wide +3.3V, once the bulk converter is enabled. When the IPMC interface module is installed, the power on/power off commands sent by the Shelf Manager are decoded by the IPMC and passed through the on-board PIC microcontroller to the power modules.

1.5 Control Architecture

The PIC microcontroller is connected to a front panel Ethernet port, and communicates with a small Spartan-3 FPGA. The Spartan-3 implements a parallel address bus and a parallel data bus connected to each of the four main FPGAs for slow control purposes, as shown in Figure 3. The PIC performs register read or write operations in response to simple UDP commands packets with a target device number to select between FPGAs. This is described in detail in Section 4.1.

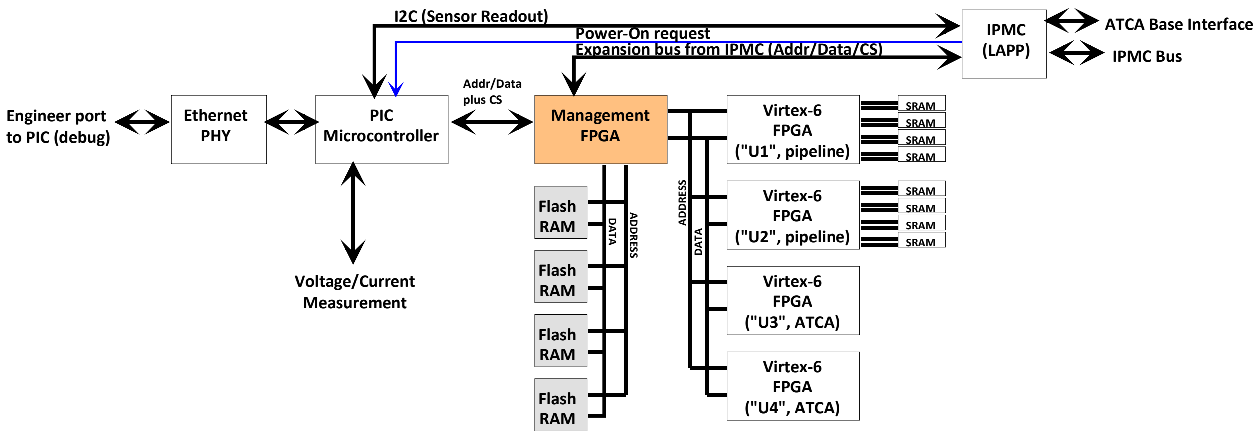

I2C (Sensor Readout)
Power-On request
Expansion bus from IPMC (Addr/Data/CS)
IPMC (LAPP)
ATCA Base Interface
IPMC Bus
Engineer port to PIC (debug)
Ethernet PHY
PIC Microcontroller
Addr/Data plus CS
Management FPGA
Virtex-6 FPGA ("U1", pipeline)
Virtex-6 FPGA ("U2", pipeline)
Virtex-6 FPGA ("U3", ATCA)
Virtex-6 FPGA ("U4", ATCA)
SRAM
Flash RAM
DATA
ADDRESS
Voltage/Current Measurement

Figure 3 – slow control architecture of FLIC

Firmware images for the main FPGAs are stored in local non-volatile Flash memory, managed by the Spartan-3 Management FPGA that may be loaded through the front panel Ethernet port. Each Flash memory chip is partitioned into areas to contain a main FPGA firmware image and module ID coordinate data storage for transfer to the SRAMs of the two Pipeline FPGAs. After the main board power is enabled, state machines in the Management FPGA are used to load the main FPGAs from the Flash memory. After FPGA initialization a separate command initiates automated transfer of the module ID data from the Flash memories into the SRAM module ID lookup tables associated with FPGAs U1 and

U2. After all initialization of FPGAs and memories is complete, operational setup is achieved by writing individual registers of the FPGAs. The 35-bit expansion bus of the LAPP IPMC module has been connected to the Spartan-3 to allow for future development of firmware to support slow control and firmware maintenance through the IPMC module.

1.6 System Monitoring

Analog multiplexers allow the PIC microcontroller to measure all voltages associated with each main FPGA using the A/D converter of the PIC. The point-of-load power distribution sub-system for each Virtex-6 FPGA is shown in Figure 4.

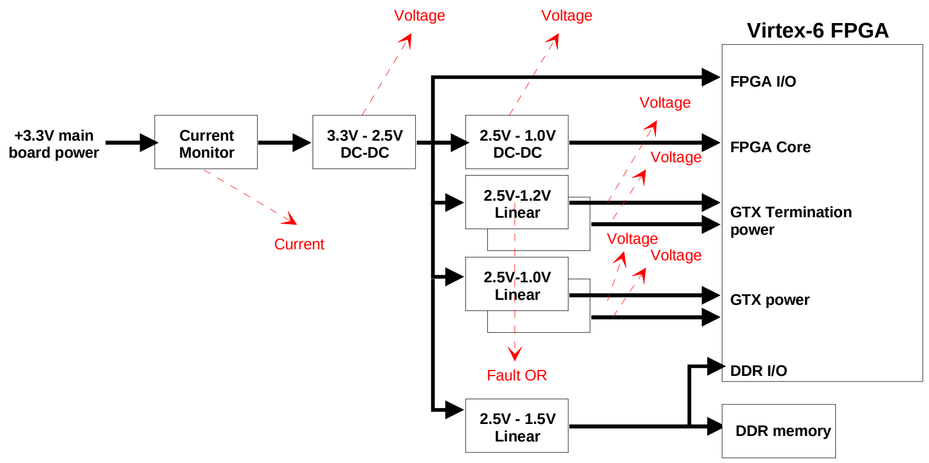


Figure 4 – Point of load distribution and monitoring

Eight analog data values are available per main FPGA; six voltages, one current and one fault signal. The ‘fault’ signal is tied by a resistive divider to the +3.3V and reads a known fraction of +3.3V if no fault is present, 0V if any of the GTX linear regulators are in a fault state. The PIC32MX695F512L implements a 16-channel, 10-bit multiplexed ADC as shown in Figure 5, taken from the Microchip documentation (section 17, figure 17-1).

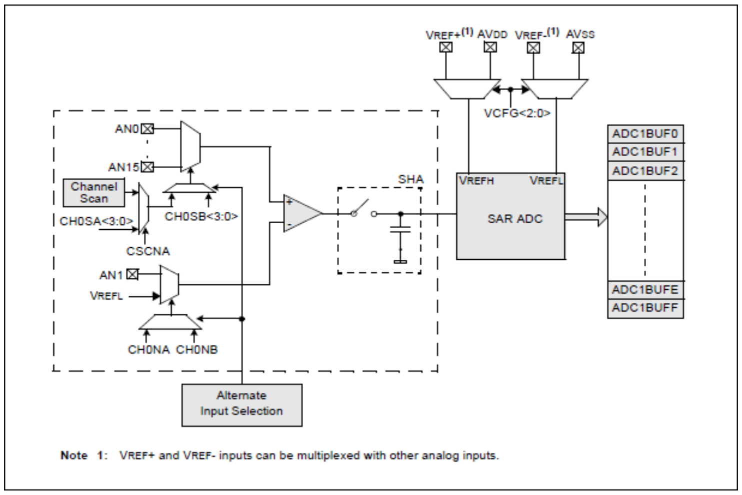


Figure 5 – PIC32MX695F512L ADC architecture

Each of the eight analog voltages is connected through an analog multiplexer to one of the multiplexed ADC inputs of the PIC (AN2, AN3, AN4, & AN5). Thus, with four main FPGAs the PIC can sample 32 voltage/current/fault parameters from the main FPGAs by first selecting which PIC ADC input is to be used (which main FPGA), then asserting a 3-bit analog mux selection code (which analog value of the selected FPGA block), and then performing a conversion. Through the front panel Ethernet interface the PIC may be commanded at any time to scan all channels and save a block of conversions to local memory. These values may then be read out through that same Ethernet interface to monitor all useful voltages & currents.

1.6.1 Temperature monitoring

The Virtex-6 FPGAs have temperature monitoring capability using the "System Monitor" block of the FPGA, as described in Xilinx document UG320. The AVDD pin of each Virtex-6 has been connected to the +2.5V power supply, and the AVSS, VREFP and VREFN pins are all tied to ground, setting up the "System Monitor" for its simplest configuration using the internal reference. The "System Monitor" temperature is regularly sampled and made available through the register interface of each FPGA build (Pipeline, ATCA Interface and SSB Emulator).

A secondary method of temperature measurement is supported by the PIC and the Management FPGA. This method uses the variable output voltage obtained from the Comparator Reference block of the PIC, and four digital outputs from the Management FPGA. Through these connections the temperature sensing diode of each main FPGA may be separately energized and the voltage dropped across the selected diode read using analog input channel AD11 of the PIC.

1.6.2 Monitoring through ATCA

The LAPP IPMC is connected to the PIC via an I2C bus so that the PIC is the single "sensor" for the entire board. Interrupt-driven software within the PIC based upon a timer may perform measurements of all voltage, current and temperature parameters in the background, leaving the most recent measurements in a block of memory. When accessed by the LAPP IPMC, this block of memory may be read out and made available to the Shelf Manager.

2 ELECTRICAL & MECHANICAL SPECIFICATIONS

2.1 PC board construction

The FLIC is standard FR-4 construction with 18 trace layers. The surface finish is electroless gold over nickel (ENIG) with white silkscreen over liquid photo-imageable soldermask on both sides. All differential traces are 100 ohm characteristic impedance.

2.2 Mechanical specifics

The FLIC is the standard size specified in PICMG 3.0 for an ATCA front board. A Schroff front panel with Southco handles (ATCA standard) is used.

2.3 Power and Cooling requirements

Nominal power dissipation of the FLIC is 57 Watts with all FPGAs fully programmed and active. Of this, approximately 8 Watts is management power.

2.4 Front Panel Connectors

Front panel connections to the FLIC include two JTAG connectors, a PIC programming connector and a front panel Ethernet port.

2.4.1 JTAG Connectors

Two JTAG connectors are provided at the front of the FLIC for debugging and analysis of FPGA operation. Due to the differing voltage requirements of the Spartan-3 vs. the Virtex-6, unique JTAG chains are required. Two standard 20-pin Xilinx JTAG connectors are provided, one for each chain, immediately behind the front panel, as shown in Figure 6. The Main FPGA JTAG chain uses +2.5V power as is required for Virtex-6 FPGAs. The Management FPGA JTAG chain uses +3.3V power as is required for the Spartan-3.

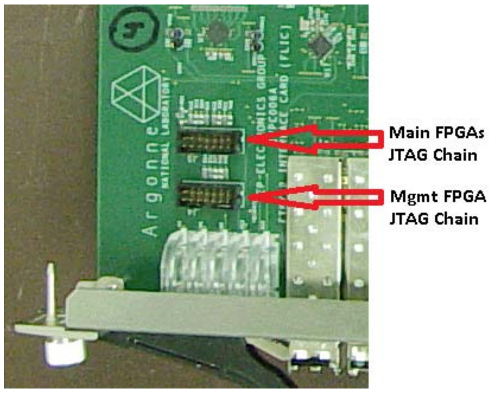


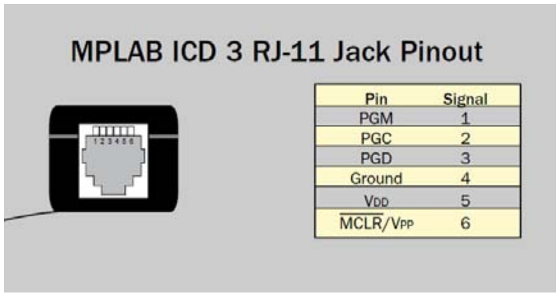
Figure 6 – JTAG connector location and identification

Internal zero ohm resistors are used to provide oscilloscope probing points for debugging of JTAG problems. These resistor sets, one per Virtex-6, may be installed in two different ways to either include or exclude the given FPGA from the JTAG chain.

2.4.2 PIC programming connector

A 6-pin telephone (RJ-11) jack is used for the PIC programming interface. Figure 7 shows the pinout, taken from the Microchip web site at

http://ww1.microchip.com/downloads/en/DeviceDoc/DS-51765C.pdf.



MPLAB ICD 3 RJ-11 Jack Pinout

Pin	Signal
PGM	1
PGC	2
PGD	3
Ground	4
VDD	5
MCLR/VPP	6

Figure 7 – Microchip PIC ICD connector pinout

2.4.3 Front Panel Ethernet connector

The front panel Ethernet connection of the FLIC is a standard RJ-45 connector (Amphenol J00-0065NL with internal magnetics). The FLIC uses a KSZ8051RNL PHY and supports communication at speeds up to 100BASE-T.

2.4.4 Front Panel LED indicators

Twelve front panel LED indicators provide general status. The LEDs are arranged in a 5x3 block of LEDs with 7 subsections as shown in Figure 8.

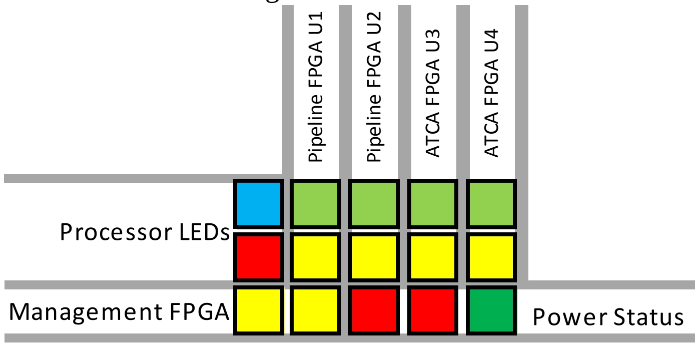


Figure 8 – LED indicator block

The two Processor LEDs are labeled ATCA (blue) and ERR (red). If an IPMC module is plugged into the FLIC a resistor stuffing option allows the “ATCA” LED to be driven by the IPMC in accordance with ATCA specifications. In the default assembly the “ATCA” LED is driven by the on-board PIC processor and is used for generic status. The ERR indicator is normally illuminated when the FLIC has no Ethernet connection on the front panel port and extinguishes when an Ethernet connection is present.

The two Management FPGA LEDs are the CNFG (configure) and the PROG (labeled “P”) indicators. Both LEDs are under the control of the Management FPGA and are generally used to indicate the status of firmware download to the four main FPGAs. However, these LEDs are under firmware control of the Management FPGA and the exact meaning depends upon the firmware version in place.

Three Power Status LEDs, labeled "A", "B" and "M", display the power state of the FLIC. The "A" LED is illuminated if the +3.3V management power to the FLIC is present and the internal voltage monitor for the Management FPGA indicates that all sub-voltages derived from management power are within tolerance. The "B" LED illuminates if the board-wide +3.3V power is enabled. The "M" LED illuminates if the +3.3V management power is present.

Each Virtex-6 main FPGA of the FLIC ("U1" through "U4") drives two LEDs. The upper green LED is labeled "EV" (for "event") for each and the lower yellow LED is labeled "FC" (for "flow control") as the nominal expected usage is to illuminate the LEDs to indicate that records are being processed and/or that flow control is being asserted. However, both LEDs are defined by the exact firmware loaded into the main FPGAs and thus the meaning of each may vary.

2.4.5 IPMC module LED signals

The IPMC module from LAPP has four LED output signals named ATCA, LED1, LED2 and LED3, all of which are defined by whatever firmware is running within the processors of the IPMC module. As previously noted a resistor stuffing option allows the blue ATCA LED of the FLIC to be driven by the ATCA output of the IPMC. The LED1 and LED2 outputs of the IPMC are connected to input pins of the PIC microcontroller to allow either of these signals to be monitored or optionally routed out to the 2nd processor LED (ERR) by firmware if deemed necessary.

2.5 Rear Connectors

The FLIC conforms to all rear connector pinouts for ATCA Zone 1 and ATCA Zone 2 as specified in PICMG 3.0. Additional connectors in Zone 3 are used for the RTM of the FLIC, with a custom pinout as defined in Table 1**Error! Reference source not found.**. Each pair of rows services one SFP of the RTM, as indicated by the bold lines.

Row	Pin A	Pin B	Pin AB	Pin C	Pin D	Pin CD	Pin E	Pin F	Pin EF	Pin G	Pin H	Pin GH
P1-1	TX+	TX-	GND	RX+	RX-	GND	RATESEL	LOS	GND	TFAULT[1]	TDIS	GND
P1-2	N/C	N/C	GND	N/C	MOD DET	GND	RSVD[2]	N/C	GND	SERCLK	SERDAT	GND
P1-3	TX+	TX-	GND	RX+	RX-	GND	RATESEL	LOS	GND	N/C	TDIS	GND
P1-4	N/C	N/C	GND	N/C	MOD DET	GND	N/C	N/C	GND	SERCLK	SERDAT	GND
P1-5	TX+	TX-	GND	RX+	RX-	GND	RATESEL	LOS	GND	N/C	TDIS	GND
P1-6	N/C	N/C	GND	N/C	MOD DET	GND	N/C	N/C	GND	SERCLK	SERDAT	GND
P1-7	TX+	TX-	GND	RX+	RX-	GND	RATESEL	LOS	GND	N/C	TDIS	GND
P1-8	N/C	N/C	GND	N/C	MOD DET	GND	N/C	N/C	GND	SERCLK	SERDAT	GND
P1-9	N/C	N/C	GND	N/C	N/C	GND	N/C	N/C	GND	N/C	N/C	GND
P1-10	N/C	N/C	GND	N/C	N/C	GND	N/C	N/C	GND	N/C	N/C	GND
P2-1	TX+	TX-	GND	RX+	RX-	GND	RATESEL	LOS	GND	N/C	TDIS	GND
P2-2	N/C	N/C	GND	N/C	MOD DET	GND	N/C	N/C	GND	SERCLK	SERDAT	GND
P2-3	TX+	TX-	GND	RX+	RX-	GND	RATESEL	LOS	GND	N/C	TDIS	GND
P2-4	N/C	N/C	GND	N/C	MOD DET	GND	N/C	N/C	GND	SERCLK	SERDAT	GND
P2-5	TX+	TX-	GND	RX+	RX-	GND	RATESEL	LOS	GND	N/C	TDIS	GND
P2-6	N/C	N/C	GND	N/C	MOD DET	GND	N/C	N/C	GND	SERCLK	SERDAT	GND
P2-7	TX+	TX-	GND	RX+	RX-	GND	RATESEL	LOS	GND	N/C	TDIS	GND
P2-8	N/C	N/C	GND	N/C	MOD DET	GND	N/C	N/C	GND	SERCLK	SERDAT	GND
P2-9	N/C	N/C	GND	N/C	N/C	GND	N/C	N/C	GND	N/C	N/C	GND
P2-10	N/C	N/C	GND	N/C	N/C	GND	N/C	N/C	GND	N/C	N/C	GND

Table 1 – Pinout of FLIC RTM connector

+3.3V power to the RTM of the production FLIC is provided by connector J30 as specified in PICMG 3.8.

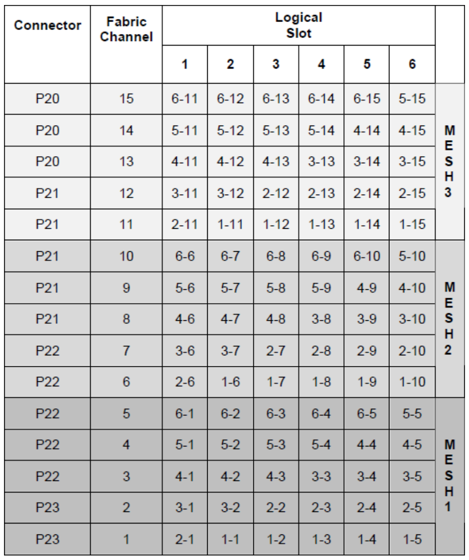
2.6 Physical Interface to ATCA backplane

The FLIC makes connection to the ATCA backplane for power and to the main fabric for communication with processor blades. A given FLIC connects to Fabric Channels 2, 3, 4 & 5 of the fabric, with each of the four channels connected to a 10Gbit Ethernet core implemented in one of the FPGAs of the FLIC. FPGA "U3" connects to two of the channels, while FPGA "U4" connects to the other two channels. This allows a FLIC in slot 1 or slot 2 of a six-slot ATCA backplane to communicate with processor blades in slots 3, 4, 5 & 6. The reader is reminded that a 1X mesh backplane is identically the same as a dual-star backplane at the six-slot size. Most six-slot backplanes are 3X mesh with

[1] TFAULT is normally not connected in most SFP adapters. In the FLIC RTM all TFAULT pins of all eight SFPs are connected via isolating resistors to the corresponding G-row pin. The FLIC connects FPGA "U1", pin K32, to pin G, row 1 to allow for testing of SFPs that do implement TFAULT.

[2] On the FLIC pin E, row 2 of the upper RTM connector is wired to I/O pin P30 of the "U1" FPGA but the FLIC's RTM does not connect. This connection is reserved and should not be used.

connectivity as shown in Figure 9**Error! Reference source not found.**; the FLIC only connects to Mesh #1.



Connector	Fabric Channel	Logical Slot						
		1	2	3	4	5	6	
P20	15	6-11	6-12	6-13	6-14	6-15	5-15	MESH 3
P20	14	5-11	5-12	5-13	5-14	4-14	4-15	
P20	13	4-11	4-12	4-13	3-13	3-14	3-15	
P21	12	3-11	3-12	2-12	2-13	2-14	2-15	
P21	11	2-11	1-11	1-12	1-13	1-14	1-15	
P21	10	6-6	6-7	6-8	6-9	6-10	5-10	MESH 2
P21	9	5-6	5-7	5-8	5-9	4-9	4-10	
P21	8	4-6	4-7	4-8	3-8	3-9	3-10	
P22	7	3-6	3-7	2-7	2-8	2-9	2-10	
P22	6	2-6	1-6	1-7	1-8	1-9	1-10	
P22	5	6-1	6-2	6-3	6-4	6-5	5-5	MESH 1
P22	4	5-1	5-2	5-3	5-4	4-4	4-5	
P22	3	4-1	4-2	4-3	3-3	3-4	3-5	
P23	2	3-1	3-2	2-2	2-3	2-4	2-5	
P23	1	2-1	1-1	1-2	1-3	1-4	1-5	

Figure 9 – six slot 3X mesh connectivity

With two FLICs mounted in the two Hub slots (slots 1 and 2) of the 1X mesh, the connectivity of the FLIC then provides two 10G channels of communication to each of three blades (in slots 3,4 & 5), one 10G channel from each FLIC. This allows for direct distribution of data across the blades by simple round-robin logic within the FLIC, as shown in **Error! Reference source not found.**. This is sufficient to allow copying the full bandwidth of all data being processed by each FLIC over the fabric at the original specifications as three 10Gbit Ethernet channels is sufficient to carry the 24Gbit aggregate bandwidth of the eight fiber optic inputs with each fiber running at 2Gbit/s. In this architecture no Hub or Switch board is used in the ATCA shelf as the direct connections of the backplane map three of the four FLIC output channels to the fabric channels used by commercial processor blades in more traditional hub/switch ATCA implementations, with one of the fabric channels (2-1 ⇔ 1-1) connecting the two FLICs to each other.

Channel	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	FLIC
Fabric CH5	6-1	6-2	6-3	6-4	6-5	5-5	No Connect
Fabric CH4	5-1	5-2	5-3	5-4	4-4	4-5	U3 Eth2
Fabric CH3	4-1	4-2	4-3	3-3	3-4	3-5	U3 Eth1
Fabric CH2	3-1	3-2	2-2	2-3	2-4	2-5	U4 Eth2
Fabric CH1	2-1	1-1	1-2	1-3	1-4	1-5	U4 Eth1
	FLIC1	FLIC2	BLADE1	BLADE2	BLADE3	BLADE4	

Figure 10 – distribution of 10G channels with two FLICs in Hub slots

Installation of the FLIC into payload slots is also possible. In this case the 10Gbit Ethernet interfaces of the "U4" FPGA on fabric channels 1 and 2 naturally map to the correct channels for connection to hub/switch boards in the hub/switch slots, but the "U3" FPGA now connects to channels that are only available in a full-mesh backplane. FLICs may be used in the more common dual-star ATCA backplanes. Figure 11 shows the mapping of a commercial 14-slot dual-star ATCA backplane.

FLICs inserted into any of the payload slots would be able to use the two 10G channels connected to FPGA "U4" to drive data through the hub slots to processor blades in any other slot.

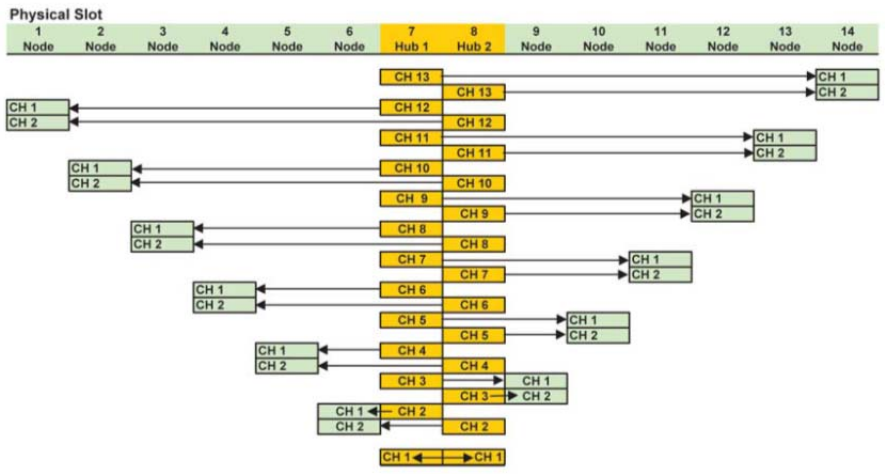


Figure 11 – 14 slot, dual-star ATCA backplane.

The FLIC provides connections to both the primary and secondary Base Interface channels using zero ohm resistors. The Ethernet interface of the LAPP IPMC module may be optionally connected to either Base Interface channel or alternatively the two Base Interface channels may simply be jumpered from one to the other.

3 FUNCTIONAL OVERVIEW

3.1 Basic Features

The FLIC receives data from eight fiber optic links, processes the data, re-formats the data to comply with the CERN S-Link protocol and sends the resulting information to the ATLAS readout system (ROS). Data is expected to be formatted as ***records***, where each record consists of a *record header*, some number of *track data blocks*, and a *record trailer*, as defined in the ATLAS Fast Tracker (FTK) documentation and copied here as Figure 12.

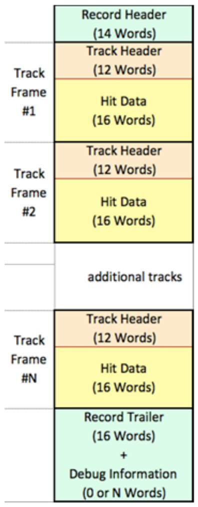


Figure 12 – Overall format of input data in the FTK system.

Within FTK, the output of the Second Stage Boards (or SSBs) is the input to the FLIC. The board design, however, is sufficiently generic to allow use of the FLIC as a general fiber-ATCA interface block. All track data received from the SSB boards initiates a per-layer coordinate lookup from SRAM data tables in the FLIC. The coordinate data is merged with the SSB data to form a complete record. That record is then reformatted into S-Link format for compatibility with ATLAS requirements and transmitted over the output fiber optic links to the ROS buffers of the ATLAS DAQ. Two Pipeline FPGAs ("U1" & "U2") perform this function.

An internal mesh of high speed serial connections connects the two Pipeline FPGAs to two ATCA Interface FPGAs ("U3" and "U4"). As records are processed in the Pipeline FPGAs, a selected subset of the records is copied over the internal mesh to the ATCA Interface FPGAs. Each ATCA Interface FPGA assembles the eight record fragments (one from each fiber input, one per pipeline, four per Pipeline FPGA) into the fully assembled record. DDR memories are used for the assembly process. When each assembled record is complete, it is transferred over the ATCA backplane to a processor blade or hub using 10Gb Ethernet (XAUI).

Management of all board functions is accomplished through a slow control interface between the Management FPGA and the four main FPGAs. Flash memory connected to the Management FPGA is used to hold main FPGA firmware images plus module ID data that is merged by the FLIC with the SSB data during the pipeline process.

3.2 Block Diagram

The overall architecture of the FLIC is given in Figure 13. The control section is at the left, with the two pipeline FPGAs in the middle and the two ATCA Interface FPGAs at the right. Connectivity is noted as orange text.

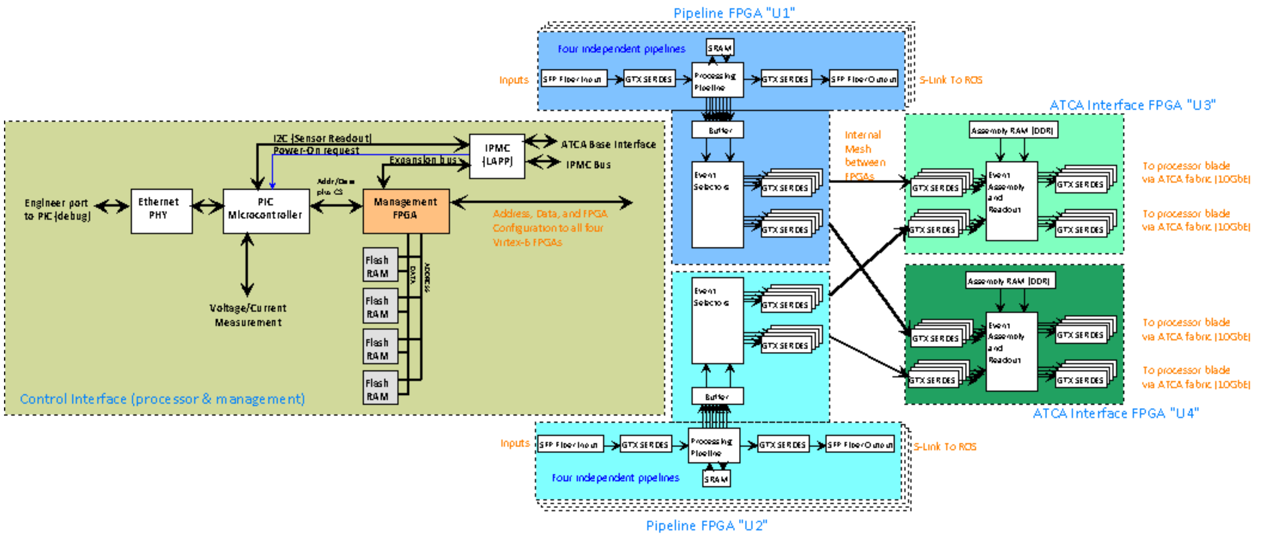


Figure 13 – Overall block diagram of FLIC

An internal mesh of SERDES connections, equal in bandwidth and number to the input and output SERDES connections, connect the two Pipeline FPGAs to the ATCA Interface FPGAs. To complete the internal mesh, Pipeline FPGAs U1 and U2 connect a GTX bank to each other, plus ATCA Interface FPGAs U3 and U4 connect a GTX bank to each other. At the present time this additional connectivity is unused. The full internal mesh of FPGA serial connectivity is shown, along with which clock references are associated with each GTX bank, in Figure 14. The specifics of clock generation are detailed later in this document.

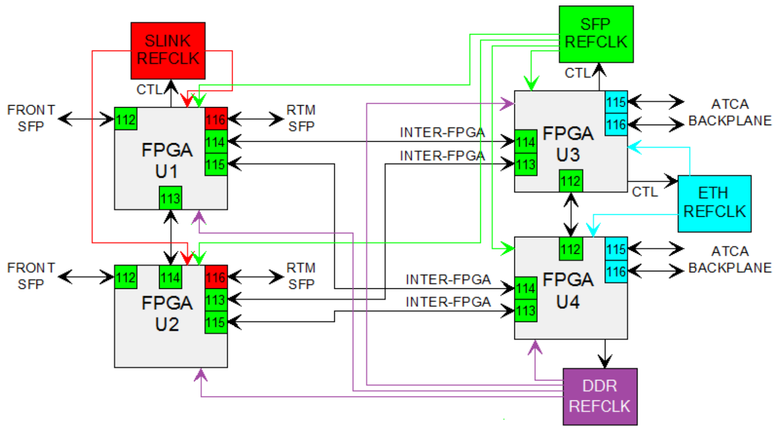


Figure 14 - Internal SERDES mesh connectivity within the FLIC

3.2.1 Data object terminology

It is at this juncture necessary to define a few terms that will be used throughout this document to describe various data structures that flow throughout the FLIC.

- The term ***"event"*** shall be avoided throughout this document as it is understood that this term is reserved for defining experiment-wide data objects that contain data from all the detector systems within ATLAS.
- A data object entering the FLIC Pipeline FPGAs from the SSB shall always be referred to as a ***record.*** As previously stated in Section 3.1, above, each ***record*** consists of a ***record header***, some number of ***tracks***, and a ***record trailer***.
    - These terms will be used throughout the description of the Pipeline FPGAs as shown in Figure 13, Figure 15 and throughout chapter 5, DATA PROCESSING PIPELINE FIRMWARE.
- As the ***records*** flow through the Pipeline FPGAs, some subset of ***records*** is selected for copying to either or both ATCA Interface FPGAs. Such copied ***records*** are referred to as ***tagged records***.
- Each ATCA Interface FPGA refers to the received tagged records as ***fragments.***
- Sets of ***fragments*** with the same Level 1 ID are collected together by the ATCA Interface FPGA into ***assembled fragments***.
    - The terms ***fragment*** and ***assembled fragment*** are used throughout this document as seen in Figure 16, Figure 27 and all of chapter 6, ATCA Interface Firmware.
- ***Assembled fragments***, when complete, are then transmitted as a series of *UDP Packets* over the backplane to processor blade(s).
- As each FLIC receives information from only half of FTK, a processor blade must merge together the ***assembled fragments*** from two different FLICs using the dual-star nature of the ATCA backplane as shown in Figure 10 and Figure 11.
    - Once the two ***assembled fragments*** with the same Level 1 ID index have been merged within the processor blade, only at this point should the data be referred to as an ***event fragment***, as it is at this point that the data block contains all the information from the FTK detector, but not any data from the rest of ATLAS.

3.3 Clock Domains within the FLIC

Multiple clock domains exist within the FLIC. Three of those, used in the main data flow, are shown in Figure 14. The generation and distribution of the clock domains is accomplished using a series of CDCM61004 reference clock generator chips with fanout logic. Each CDCM61004 uses its own local 25MHz reference crystal to generate a variety of frequencies based upon the state of various control lines driven from the main FPGAs. The list of clock generators is below. The details of which frequencies each can generate is found in Section 1.

- Clock generator #1, controlled by FPGA "U3", provides the reference for all front panel and inter-FPGA serial links, the "SFP REFCLK" (typically 125.0MHz for 2Gb/sec operation or 187.5MHz for 3Gb/sec).
- Clock generator #2, controlled by FPGA "U1", provides the reference for all S-Link (RTM) serial links (typically 125.0MHz, corresponding to 2Gb/sec serial rate).
- Clock generator #3, controlled by FPGA "U4", provides the reference for the 10Gb Ethernet serial links to the ATCA backplane (typically 156.25MHz).

- Clock generator #4, also controlled by FPGA “U4”, provides the reference for all DDR memories, typically 200MHz.
- The PIC processor is connected to a 25MHz oscillator that it internally multiplies with a PLL to 80MHz. The 25MHz is required by the Ethernet PHY. The Management FPGA also generates, and forwards to all four Virtex-6 FPGAs, an 80MHz clock that is used for the slow control bus.
    - An internal clock multiplier is used in the Pipeline FPGAs to generate the internal pipeline processing clock of 200MHz. Generating the pipeline clock from the 80MHz decouples that clock from the DDR clock that may change dependent upon how the DDR interface core is designed.

3.4 DDR Memory Buffers

Each of the four main FPGAs has a single 1Gbit MT41J128M16JT-125 DDR RAM connected using a soft core. The DDR connected to each of FPGAs U1 and U2 is reserved for future use. The DDR bandwidth (12.8 Gb/s) is not sufficient to buffer all data from all SSB streams simultaneously. The DDR memories attached to FPGAs U3 and U4 are used for record assembly and buffering from the internal mesh prior to transmission of records to the ATCA backplane.

3.5 Data Processing Pipeline Overview

The internal gigabit transceiver, or GTX, blocks of bank 112 in both U1 and U2 are connected to the front panel SFPs and are driven by the same clock reference. The serial rate can be changed by a combination of GTX core settings and reprogramming of the clock reference, although reloading of firmware is typically required to change speeds. The FLIC has been tested at rates of 2Gbit/s and 3Gbit/s. A pipelined series of state machines process the data records received in stepwise fashion and present the final data to CERN S-Link cores in the GTX SERDES blocks of bank 116 of U1 and U2. The eight serial streams from GTX116 of both U1 and U2 are connected to a second bank of SFP modules mounted on a rear transition module (RTM). The RTM SERDES links have a separately programmable clock reference so that the input and output links may be run at different line rates. The overall flow of the data processing pipeline is described by Figure 15.

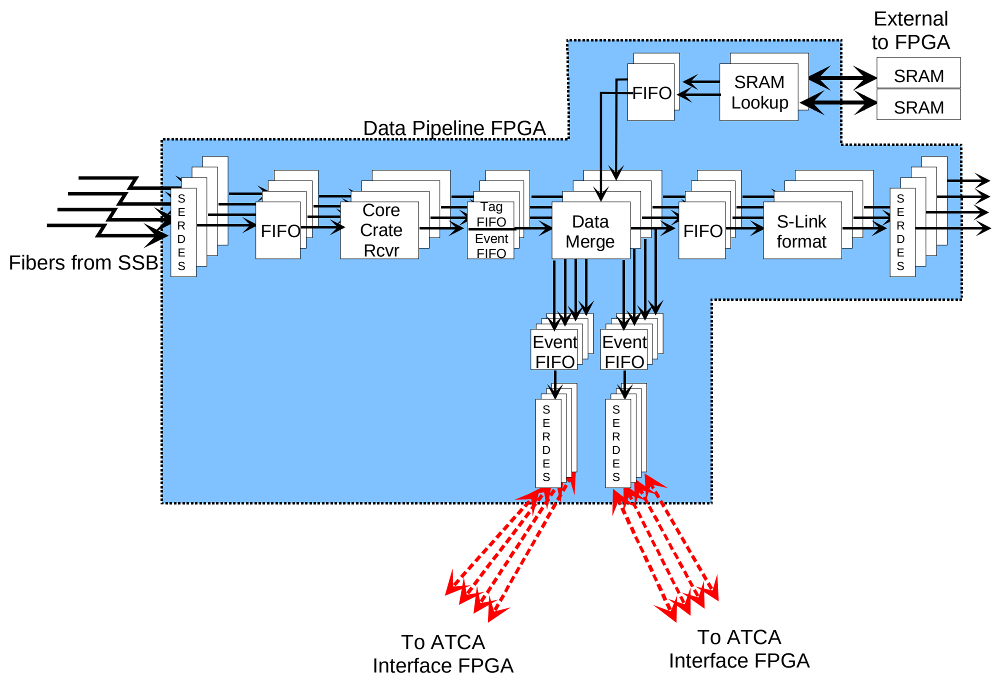


Figure 15 - General data pipeline architecture used in U1 and U2

Data enters at the left. FIFO buffers are placed between each state machine to allow for clock domain crossing and record management. The Core Crate Receiver section locks onto the data format as provided by the SSB and checks data consistency. The Data Merge machine looks up module ID data from the SRAM and merges that with the original SSB data to make the final record. The S-Link format machine wraps the SSB data into correct S-Link format, after which the records are transmitted to the DAQ. At the Data Merge level, selected records may be transferred over the internal SERDES mesh to the ATCA Interface FPGAs.

3.6 ATCA Interface Overview

The two ATCA Interface FPGAs are tasked with assembling record fragments from the eight independent pipelines into FLIC-wide records and then transmitting those assembled records over the ATCA backplane to commercial processor blades. It is understood that a FLIC-wide record is, itself, only a fragment of a full FTK event as there are multiple FLICs in FTK. The assembly of FLIC-wide events into full FTK event fragments is presumed to be accomplished in software.

Each SSB record is tagged with an identifier called the L1 ID. As the record is processed in the Pipeline FPGA, the L1 ID of the record is compared against a selection mask to determine whether the event should be copied to the ATCA Interface FPGA. The selection mask is simple and selection of all records is allowed. Records so selected are marked with bits in the Tag FIFOs of Figure 7. The Tag FIFO bits indicate whether the record is to be copied to ATCA Interface FPGA "U3", ATCA Interface FPGA "U4", or both. This tagging is done by the Core Crate Receiver machine. When each record fragment has been completely received and is being read out by the Data Merge machine, any tagged record is copied to Event FIFOs and transmitted across the internal mesh.

Each ATCA Interface FIFO receives eight SERDES links from the internal mesh, matching the number of fiber optic inputs to the FLIC board. The internal mesh supports double the bandwidth of the inputs such that all data could be copied to either "U3" or "U4" alone without loss, but the bandwidth of the DDR memory, which is required to a single ATCA Interface FPGA is insufficient to carry the total input bandwidth of the FLIC. The general design of ATCA Interface FPGA is shown in Figure 16.

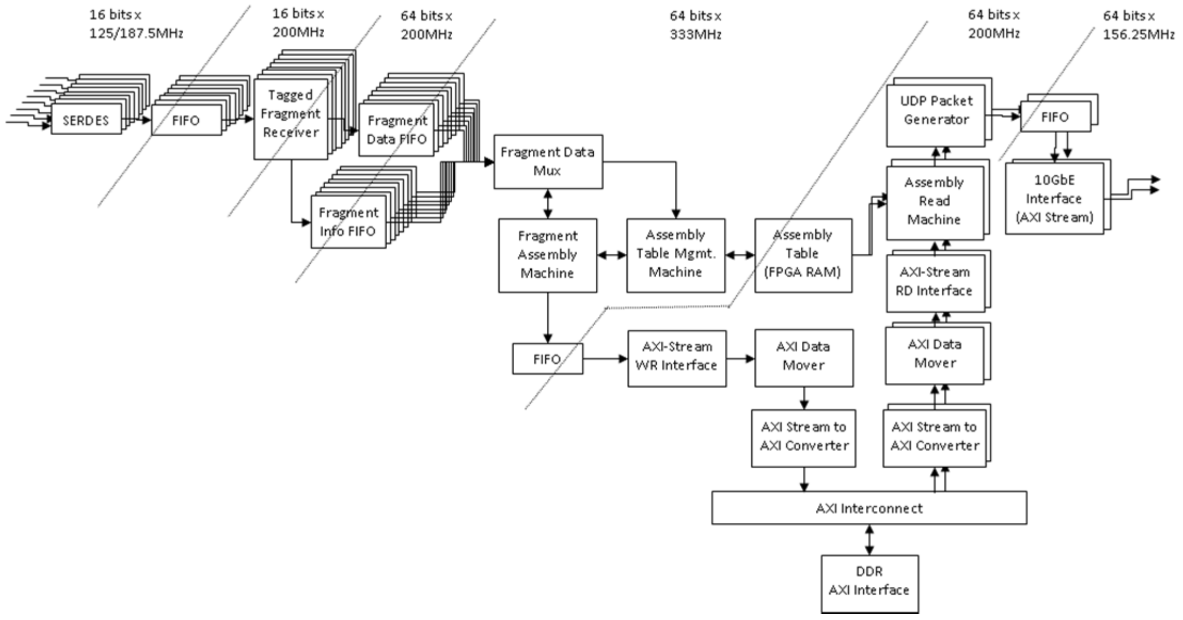


Figure 16 – Block diagram of ATCA Interface FPGA logic

As the fragments are received they are sorted by L1 ID number and copied into *assembly regions* in the external DDR memory buffer. An Assembly Data Table, implemented as a quad-port RAM internal to the FPGA, keeps track of the state of each assembly region. The table also stores data that will form the Assembly Header. When a given assembly region is fully assembled, one of the two Assembly Read Machines are selected to retrieve the assembled record. The Assembly Read Machine collects the record from the DDR using as many reads as required and passes the completed package to the UDP Packet Generator. The UDP Packet Generator receives the completed assembly data package and parses it into UDP packets. If the assembly data is larger than a single packet, the UDP Packet Generator will used IP packet fragmentation to ensure that each assembly is properly received by the processing blade. Packets are handed off to the 10GbE interface via an intermediate domain crossing FIFO. It then sends the UDP packets to the processing blades over the ATCA backplane. Two 10GbE Interfaces are implemented in each ATCA Interface FPGA, allowing the FPGA to send data to two different processors.

Within the FPGA firmware design, the data bus bandwidth is stepped up in stages from 2Gbps per channel at the SERDES input to 21.3Gbps by the time it reaches the Fragment Assembly Machine. There are two reasons for this:

1) Unlike in the Data Pipeline FPGAs, the ATCA Interface FPGAs have a variable number of parallel data pipelines depending at the different processing stages of the design. At the SERDES interface, there are 8 receiver pipelines. However, the sorting process permits only a single data pipeline, requiring at minimum a data handling rate of 16 Gbps. While the DDR cannot sustain this speed, the logic can still process a burst of data at this rate as a result of the large DDR write buffer implemented in and the Fragment Receive FIFOs provided on each input link. These buffers and FIFOs are implemented FPGA RAM and run at the full processing bandwidth.
2) The Fragment Assembly Machine cannot immediately commit to writing a fragment to the DDR. It must wait a few clocks for a decision from the Assembly Table Management Machine to begin processing the fragment, prior to both state-machines proceeding in parallel with their respective tasks. The amount of time it takes to process a fragment depends on a few factors discussed in Section 6. The high data bandwidth allows it to compensate for this processing overhead.

3.6.1 Spy Buffer

The design of the ATCA Interface FPGAs, combined with the primitive record selection logic of the Pipeline FPGAs, implements the "Spy Buffer" requirement of FTK. Various subsets or all of the input data as received by the FLIC may be collected and transmitted to the processor blades to allow for analysis without affecting the bandwidth of the path from SSB to ROS. There is no facility within the current firmware to select or change the point in the pipeline at which the data is picked off. Verification of the data transformations that occur within the FLIC by each state machine may be separately accomplished by use of the slow control Monitoring FIFOs and counters.

3.7 Flow Control

Use of a 200MHz clock for the internal processing pipeline and the 16-bit width of the internal buses defines that each pipeline of the FLIC is able to process data at 3.2Gb/sec. Thus, for the expected data input rate of 2Gbits/sec s the pipeline should never be overrun by the input data stream. On the back end, however, no such guarantee exists. Flow control from the S-Link interface is monitored and propagated backwards through the FLIC. Whenever the S-Link signals "XOFF", or if any internal FIFO buffer fills to greater than 3/4th full, an "XOFF" message is sent out the front panel port of the FLIC. The "XON" message to clear the "XOFF" is not sent until none of the FIFOs within that pipeline are asserting the programmable full flag and the back-side S-Link is also no longer asserting "XOFF".

The size of a record as sent to the ROS is different from that received from the SSB due to the multiple modifications to the data that occur within the FLIC processing pipeline. Some header/trailer information used by the FLIC to synchronize to the SSB data stream is stripped, whereas the FLIC also adds Module ID information in the Data Merge process and S-Link header/trailer information in the S-Link formatting process. Because of this the data expansion percentage is not fixed but is due to the number of tracks within the record.

Breaking the record into header, track and trailer components the net effect of the entire pipeline, including S-Link, is as follows:

- Header information per record increases from 14 sixteen-bit words to 18 sixteen-bit words per record, or 28.6%.
- Track information ***per track*** increases from 20 sixteen-bit words to 44 sixteen-bit words per track, or 120%.
- Trailer information per record increases from 16 sixteen-bit words to 18 sixteen-bit words per record, or 12.5%.

Knowing the above a graph can be generated showing the maximum achievable *input* record rate as a function of the number of tracks in the record versus the *output* record rate as a function of the number of tracks in the record. In order to generate such a graph the differentiation between ***line rate*** and ***payload rate*** must be understood. When a rate of 2Gb/sec is specified, this is the ***line rate***, or the actual bit rate on the fiber. However, the SERDES blocks use 8b/10b encoding so the ***payload rate*** is 8/10 of the ***line rate***, or 1.6Gb/sec. Similarly, 10Gb Ethernet links on the ATCA backplane consist of four differential pairs, each with a ***line rate*** of 3.125Gb/sec but a ***payload rate*** of 2.5Gb/sec.

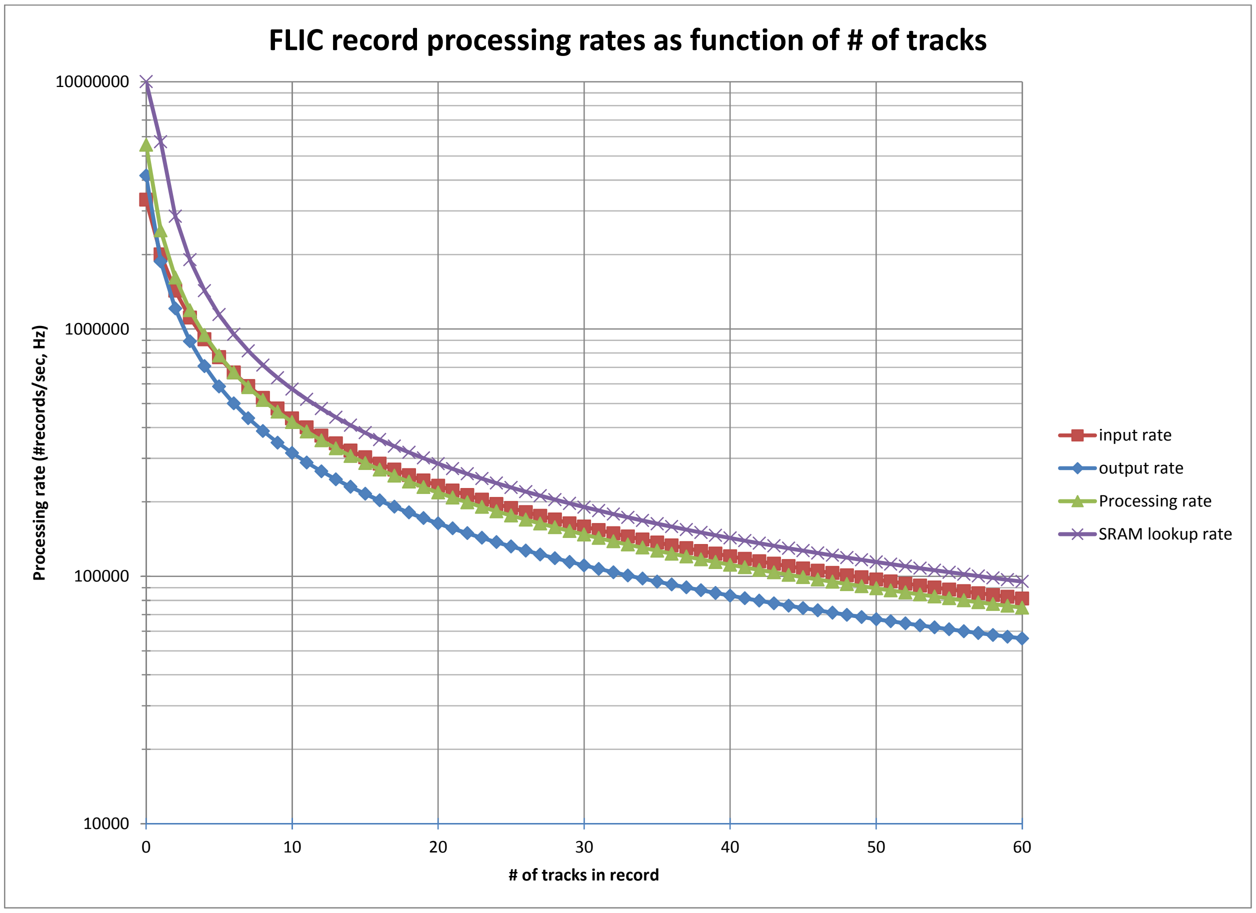


Figure 17 – Estimated FLIC record processing rates with 2Gb/sec input and output rates

Figure 17 shows the estimated record processing rate of the FLIC at the input side, through the pipeline, at the output and at the Module ID (SRAM) for various numbers of tracks in each record. As can be seen the dominant feature is the expansion of the record size as it goes through the FLIC from the addition of the Module IDs and addition of S-Link header/trailer.

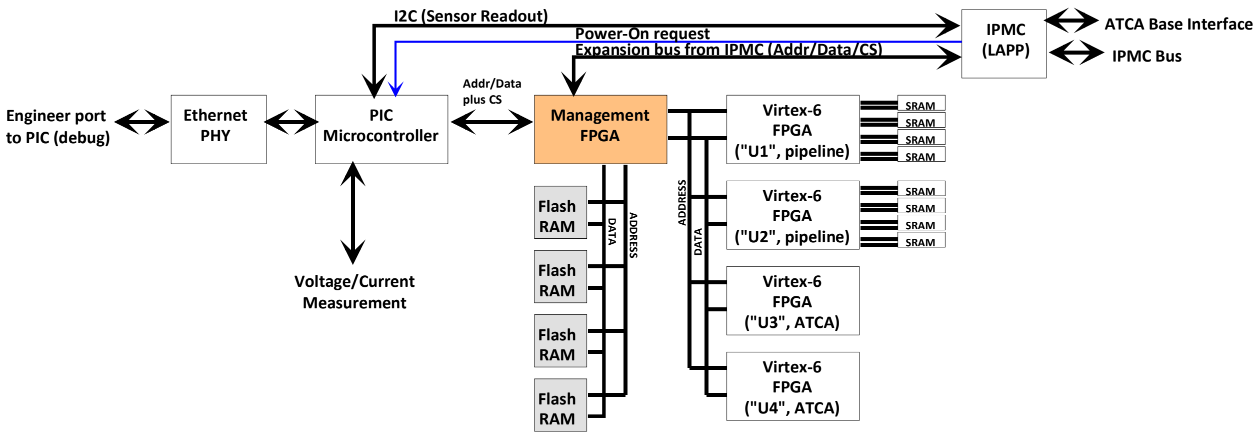
The SERDES links of the FLIC are capable of running faster than 2Gbit/sec. It is understood that the FPGAs of the SSB are of similar nature to that of the FLIC and likely could also run at a faster rate. If we hypothesize that future developments in S-Link may at some point allow the use of a 3Gbit/sec line rate, the output rate is still the dominant factor in total system speed due to the percentage-wise expansion of the data. If 3Gbit/sec line rate is assumed the output rate curve of Figure 17 rises much closer to the other rate curves, crossing the 100KHz line at 33 tracks/record instead of the current 22 tracks/record, but is still slower than the other three curves.

3.8 Secondary Parallel Mesh

In addition to the high speed serial mesh described in section 3.2, a secondary mesh of FPGA-to-FPGA connections is provided. Each main FPGA has an 8-bit unallocated bus of connections to every other main FPGA. In many cases the individual wires can be considered low quality differential pairs but no specific effort was made in board layout to ensure impedance or matched delay. The expected use of these lines is as excess connections that may be of value at some future point, either to implement a new function or to compensate for a broken solder joint. This secondary mesh has no termination other than that which can be set by the FPGAs themselves.

4 CONTROLLING THE FLIC

The FLIC implements five FPGAs on a shared slow control bus as shown previously in Figure 3 (repeated here for convenience). Access to the board for control or monitoring purposes is controlled by a small Management FPGA that processes read/write cycles from the PIC microcontroller. For compatibility with the ATCA standard an IPMC module connects to the PIC via a serial bus for sensor and monitoring requirements. At some future point the IPMC module may evolve to the point where the microprocessor of the IPMC may be the preferred control point, so the expansion bus of the IPMC is also connected to the Management FPGA.


I2C (Sensor Readout)
Power-On request
Expansion bus from IPMC (Addr/Data/CS)
IPMC (LAPP)
ATCA Base Interface
IPMC Bus
Engineer port to PIC (debug)
Ethernet PHY
PIC Microcontroller
Addr/Data plus CS
Management FPGA
Virtex-6 FPGA ("U1", pipeline)
Virtex-6 FPGA ("U2", pipeline)
Virtex-6 FPGA ("U3", ATCA)
Virtex-6 FPGA ("U4", ATCA)
SRAM
Flash RAM
DATA
ADDRESS
Voltage/Current Measurement

Copy of Figure 3

The Management FPGA firmware expects to receive single read/write transactions containing both address and data. The input side of the Management FPGA is a time-multiplexed address/data bus that is 16 bits wide.

4.1 UDP Slow Control Packet Structure

The FLIC uses a simple packet structure for the slow control communication path between the FLIC's front panel Ethernet port and a host computer. The current implementation sends UDP/IP packets but may be modified to support TCP/IP. Irrespective of the Ethernet protocol, the data payload sent by the host or the FLIC is as shown in Table 2. The orange region is a fixed size header that is included in all transactions. The green region is a variable size data portion that is only included when necessary based on the Command word in the header.

BIT =>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEST	Target Destination(15:0)															
CMD	Command(15:0)															
ADD1	Address(31:16) - Upper Bits															

ADD0	Address(15:0) - Lower Bits
SIZE	Size(15:0)
STAT	Status(15:0)
DATA0	Data Word 0 (15:0)
…	…
DATA255	Data Word 255 (15:0)

Table 2 – UDP packet structure used by the FLIC

The slow control protocol is a master/slave interface controlled by the host computer. The host sends a command to the FLIC and waits for a response. The FLIC never initiates a slow control transaction but it always sends something in response to every command. When the FLIC receives a packet, it inspects the header for a valid combination of Target, Command, Address, and Size. (The host always sends a Status word of zero.) If the header is invalid, the FLIC will respond with a packet consisting only of a copy of the received header with the Status word changed to one of the error values defined in the Appendixes of this document. The FLIC does not attempt any further processing of the command if the header is invalid.

If the FLIC receives a valid header, firmware within the PIC microcontroller performs the requested operation and returns a response packet. The response will contain a copy of received header and will append a data portion if necessary.

4.1.1 Physical vs. Logical Addressing

When software forms a command packet to the FLIC, the *target destination* field is defined as the *logical device number* that is being accessed. Within the PIC firmware the *logical device number* indexes to a table selecting a 4-bit *physical device number* (or Chip Select) that is asserted by the PIC to the Management FPGA, and also the *access method.*

The *physical address* selects which device within the FLIC that the transaction is aimed at, as summarized in Figure 18. There are five codes for registers within FPGAs, four codes for the four flash RAM chips, and a few codes for accessing the memories (SRAM and DDR) directly connected to the main FPGAs.

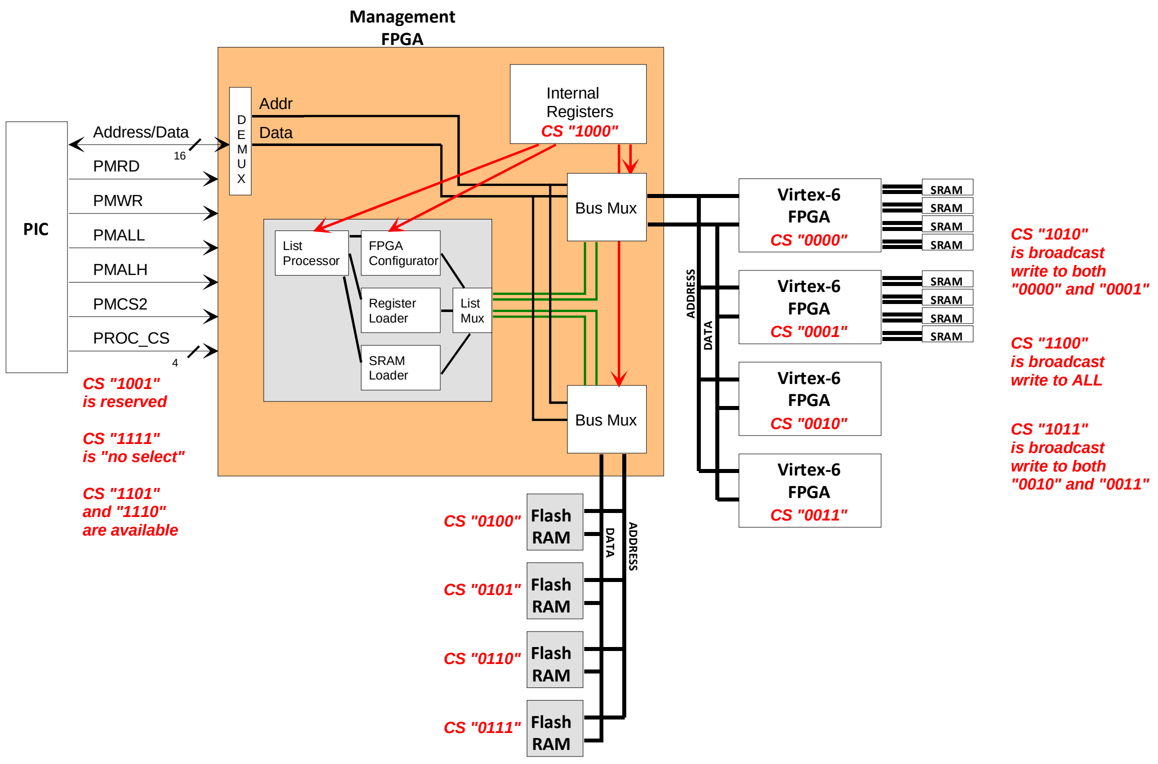


Figure 18 – Physical Device (chip select) map of the FLIC

4.1.2 Address Extension and access methods

Different physical devices on the FLIC require different size address buses, but the communication between the PIC and the Management FPGA is limited by the PIC's hardware design to a 16-bit multiplexed address/data bus. Access to the full address range of the various memory structures is accomplished through the use of address extension registers in both the Management FPGA and the main FPGAs. Each *logical address* entry in the lookup table of the PIC program contains an *access method* value that defines how the PIC will utilize the address extension registers. This is all done in PIC firmware and is invisible to user software; none the less the addressing scheme must be documented to allow future developers to understand the underlying architecture.

As shown in Figure 19, the Management FPGA implements two address extension registers, one for the FPGA address bus and one for the Flash RAM address bus. Each of these registers is written to by the PIC firmware when the Chip Select code from the PIC indicates that the cycle is to one of these physical objects.

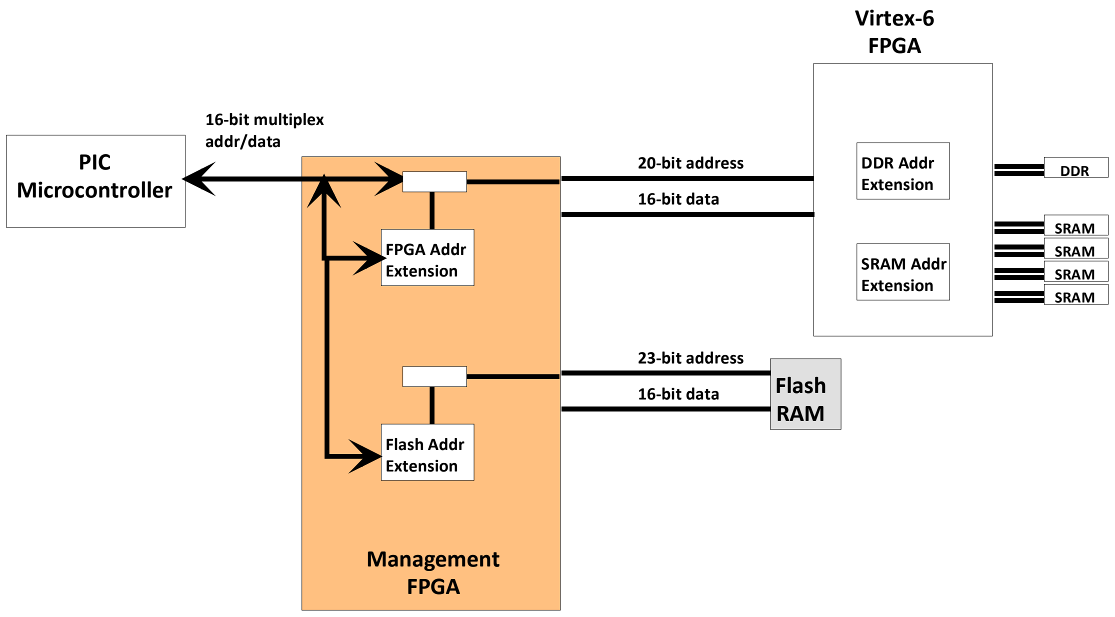


Figure 19 – Address Extension implementations

A second level of address extension is implemented within the main FPGAs. The 20-bit address asserted by the Management FPGA to the main FPGA is interpreted as containing two fields. The upper 4 bits are the *secondary device select* and the lower 16 bits are the actual address. The *secondary device select* field is used to determine whether the transaction is directed towards the internal registers of the main FPGA, or to one of the devices “behind” the main FPGA (an SRAM or the DDR). The interpretation of the *secondary device select* is shown in Table 3.

Secondary device select code (binary)	**Transaction destination**
0000	Internal register
0001	External SRAM 1
0010	External SRAM 2
0011	External SRAM 3
0100	External SRAM 4
1010	Broadcast to all four SRAM (writes only)
1101	External DDR memory

Table 3 – main FPGA sub-selection coding

4.1.3 Pipeline vs. processor access to SRAMs of Pipeline FPGAs

The two main FPGAs with external SRAM are intended to run as fast pipeline processors that continuously read data from the SRAMs as data is processed. An internal register controls access to the SRAMs by a control bit that selects whether the SRAM address and data buses are made available to the slow control (default at power up) or whether they are available to the pipeline logic. The normal method of usage is to load the SRAMs with lookup data over the slow control interface, then hand over control of the SRAMs to the pipelines when initialization is complete.

4.1.4 Details of access methods

As noted above the PIC firmware internally maps each of the various *logical device numbers* to an *access method* using an internal table. The defined *access methods* align with the secondary addressing requirements:

- Access method #0 simply uses the lower 16 bits of the address field of the packet as the address to be asserted throughout the transaction. This method is used for accessing registers within the Management FPGA.
- Access methods #1 and #2 are intended for use with the Flash RAMs connected to the Management FPGA. These RAMs are each 128Mbits in size, accessed in 16-bit words. Thus a 23-bit address is required. In these modes the PIC performs a dual-cycle transaction to first set the value of the Flash Memory Address Extension register within the Management FPGA, and then to access the selected Flash RAM where the lower 16 bits of the RAM address comes from the lower 16 bits of the address within the transaction and bits 22:16 come from the Flash Memory Address Extension register.
    - In access method #1 the upper 16 bits of the address within the transaction are ignored and the Flash Memory Address Extension register is set explicitly to zero; the lower 16 bits come from the UDP packet.
    - In access method #2 bits 22:16 of the 32-bit address in the packet are stored to the Flash Memory Address Extension register prior to accessing the Flash RAM.
- Access methods #3 and #4 are designed for register-level access within the main FPGAs. The Chip Select code, derived from the *logical address*, selects which of the FPGAs will be targeted. The address bus between the Management FPGA and the main FPGAs is a 20-bit object, and thus the Management FPGA also implements an FPGA Address Extension Register akin to the Flash Memory Address Extension register used in methods #1 and #2.
    - In access method #3 the upper 16 bits of the address within the transaction are ignored and the FPGA Memory Address Extension register is set explicitly to zero; the lower 16 bits come from the UDP packet. This mode is intended to access registers within the main FPGAs.
    - In access method #4 bits 19:16 of the address within the transaction are stored to the FPGA Memory Address Extension register prior to accessing the main FPGA; bits 15:0 of the address come from the UDP packet. This mode is intended to support single read/write transactions of a diagnostic nature to the various memory elements connected to the main FPGAs.
- Access method #5 is intended for general purpose use to communicate with any of the memory elements connected to the main FPGAs. In this method the FPGA Address Extension register of the Management FPGA is set to the correct values based upon the lookup from the *logical device number* so that the correct memory is selected. The upper 16 bits of the address in the packet are written to the FPGA Memory Address Extension register of the selected main FPGA, and then the transaction occurs with the lower 16 bits of the address in the packet asserted as the address. In short, this provides a facility to use the full 32-bit address within the packet in each main FPGA.

4.2 Slow Control Commands

Table 4 summarizes the legal combinations of commands and targets within the FLIC. It also lists the size of the header and data sections of packets sent between the host computer and FLIC for each packet type. All FLIC operations use 16-bit data. Therefore, the number of words means the number of 16-bit integers.

Host Packet				**FLIC Response**		
Command	Cmd #	Header Words	Data Words	Header Words	Data Words	Valid Target Destinations within FLIC
cmdRead	0	6	0	6	1	PIC, FLASH, FPGA, SRAM
cmdWrite	1	6	1	6	0	PIC, FPGA, SRAM
cmdSet	2	6	1	6	0	PIC, FPGA
cmdClear	3	6	1	6	0	PIC, FPGA
cmdErase	4	6	0	6	1	FLASH
cmdArrayRead	5	6	0	6	SIZE	PIC, FLASH, FPGA, SRAM
cmdArrayWrite	6	6	SIZE	6	0	PIC, FLASH, FPGA, SRAM
cmdArraySet	7	6	SIZE	6	0	PIC, FPGA
cmdArrayClear	8	6	SIZE	6	0	PIC, FPGA
cmdArrayErase	9	6	0	6	SIZE	FLASH
cmdFifoRead	10	6	0	6	SIZE	FPGA
cmdFifoWrite	11	6	SIZE	6	0	FPGA
cmdAdcScan	12	6	0	6	0	PIC
cmdAdcSingle	13	6	0	6	0	PIC

Table 4 – Supported command/target combinations in the FLIC

The individual commands are described next:

- cmdRead - This operation reads a single 16-bit value from the specified address. The header SIZE field should always equal 1. The header DEST field can be any legal value.
- cmdWrite - This operation writes a single 16-bit value to the specified address. The header SIZE field should always equal 1. The header DEST field can be the PIC, any SRAM, or any FPGA in the FLIC.
- cmdSet - This operation modifies the 16-bit value at the specified address by setting all bits that are high in the packet's single 16-bit data value. The other bits are not modified. The header SIZE field should always equal 1. The header DEST field can be the PIC or any FPGA in the FLIC.
- cmdClear - This operation modifies the 16-bit value at the specified address by clearing all bits that are high in the packet's single 16-bit data value. The other bits are not modified. The header SIZE field should always equal 1. The header DEST field can be the PIC or any FPGA in the FLIC.
- cmdErase - This operation erases the flash block containing the specified address. The header SIZE field should always equal 1. The header DEST field can be any Flash in the FLIC. A flash block is 65,536 (0x10000) words. The returned value is the contents of the status register of the flash chip following the block erase attempt.
- cmdArrayRead - This operation reads a series of 16-bit values starting from the specified address. The header SIZE field must be less than or equal to MAX_DATA_WORDS. The header DEST field can be any legal value.
- cmdArrayWrite - This operation writes a series of 16-bit values starting at the specified address. The header SIZE field must be less than or equal to MAX_DATA_WORDS. The header DEST field can be any legal value.
- cmdArraySet - This operation modifies a series of 16-bit values starting at the specified address by setting all bits that are high in the packet's corresponding 16-bit data value. The other bits are not modified. The header SIZE field must be less than or equal to MAX_DATA_WORDS. The header DEST field can be the PIC or any FPGA in the FLIC.

- cmdArrayClear - This operation modifies a series of 16-bit values starting at the specified address by clearing all bits that are high in the packet's corresponding 16-bit data value. The other bits are not modified. The header SIZE field must be less than or equal to MAX_DATA_WORDS. The header DEST field can be the PIC or any FPGA in the FLIC.
- cmdArrayErase - This operation erases a series of flash blocks starting at the specified address. The header SIZE field must be less than or equal both FLASH_BLOCKS and MAX_DATA_WORDS. The header DEST field can be any Flash in the FLIC. A flash block is 65,536 (0x10000) words. The returned values are the contents of the status register of the flash chip following each block erase attempt.
- cmdFifoRead - This operation repeatedly reads a series of 16-bit values from the specified address. The header SIZE field must be less than or equal to MAX_DATA_WORDS. The header DEST field can be any FPGA in the FLIC. However, the address must correspond to a FIFO access register or the return packet will simply contain SIZE copies of the same value.
- cmdFifoWrite - This operation repeatedly writes a series of 16-bit values to the specified address. The header SIZE field must be less than or equal to MAX_DATA_WORDS. The header DEST field can be any FPGA in the FLIC. However, the address must correspond to a FIFO access register or the result is no different than a cmdWrite to the same address with the final value.

4.3 Physical Interface between PIC and Management FPGA

The PIC32 processor communicates using a 16-bit multiplexed address/data bus, plus a few direction/strobe lines, as defined in section 13 of the PIC32 manual, "Parallel Master Port". The PIC is configured in Master mode, using 16-bit data width (PMMODE<10> = 1). The rest of the assumed settings are as follows:

- The Chip Select control (PMCON<7:6>) is set to 01, enabling PMCS2 as an actual chip select.
- Master Mode 2 (MODE<1:0> bits (PMMODE<9:8>) = 10) is assumed.
- Pin polarities are set in the PMCON register such that all control signals (PMRD, PMWR, PMALL, PMALH, PMCS1 and PMCS2) are *active high.*
- The WAITB control is set so that PMRD and/or PMWR are asserted coincident with PMCSx.
- The WAITM control is set to that PMRD is asserted for no less than 3 clock cycles.
- The WAITE control may be set to zero.
- ADRMUX<1:0> are set to 11 so that the address and data bus are fully multiplexed.
- No interrupts associated with the PMP are enabled.

4.4 Secondary methods of slow control

When the PIC performs a cycle to any device on the FLIC, the four CS lines are used by the Management FPGA to decode how to route the transaction. The PIC is always the controller of the bus multiplexers through control registers within the Management FPGA. The default case is that the PIC transaction will be forwarded to the appropriate external buses based upon the Chip Select code asserted by the PIC. There are, however, other bus control scenarios that are supported.

4.4.1 DMA transfers from flash RAM to main FPGAs

The PIC may yield control of the FPGA slow control and Flash RAM address/data buses to the internal state machines of the Management FPGA and then perform a register access to the Management FPGA to set up and then initiate a data transfer orchestrated by the state machines of the Management FPGA. In this mode a set of registers within the Management FPGA act as a "command list". The PIC first loads the desired "command list" and then initiates execution of the list by the internal list processor state machine of the Management FPGA. Each list entry can do one of three things:

- Read data from a range of flash RAM addresses and serially load the data into any or all of the main Virtex-6 FPGAs as a configuration (programming) action.
- Read data from a range of flash RAM addresses and copy that data over the parallel interface to the SRAMs connected to either FPGA "U1" or FPGA "U2", as a series of block transfers ("SRAM loader").
- Read data from a range of flash RAM addresses, interpreting that data as a set of (address, data) values, to load registers of a Virtex-6 in a controlled order ("register loader").

When using the DMA transfer capabilities of the Management FPGA, the PIC can monitor the operation by reading status registers of the Management FPGA. These status registers indicate whether a given DMA machine is active or not, and whether the DMA transaction ended successfully or not.

4.4.2 Use of the DMA engines at board startup

In normal operation within an ATCA shelf the FLIC only powers the Management FPGA, IPMC module and PIC processor until such time as the Shelf Manager of the ATCA shelf sends commands to the IPMC to tell the FLIC that it may turn on the rest of the board. After power is applied to the main FPGAs they must be configured. This is accomplished by using the "configurator" state machine to load firmware into the FPGAs. After each main FPGA is configured and running, the SRAM lookup tables associated with the two pipeline FPGAs ("U1" and "U2") must also be initialized. This is accomplished by using the "SRAM Loader" state machine. The "register loader" machine may similarly be used to initialize the firmware of the main FPGAs after the FPGAs have been configured.

Slow control software may then perform partial, selective or complete initialization of the board at any time by appropriate use of the "command list" registers and the DMA engines. A separate document that details the firmware of the Management FPGA provides full details of this operation.

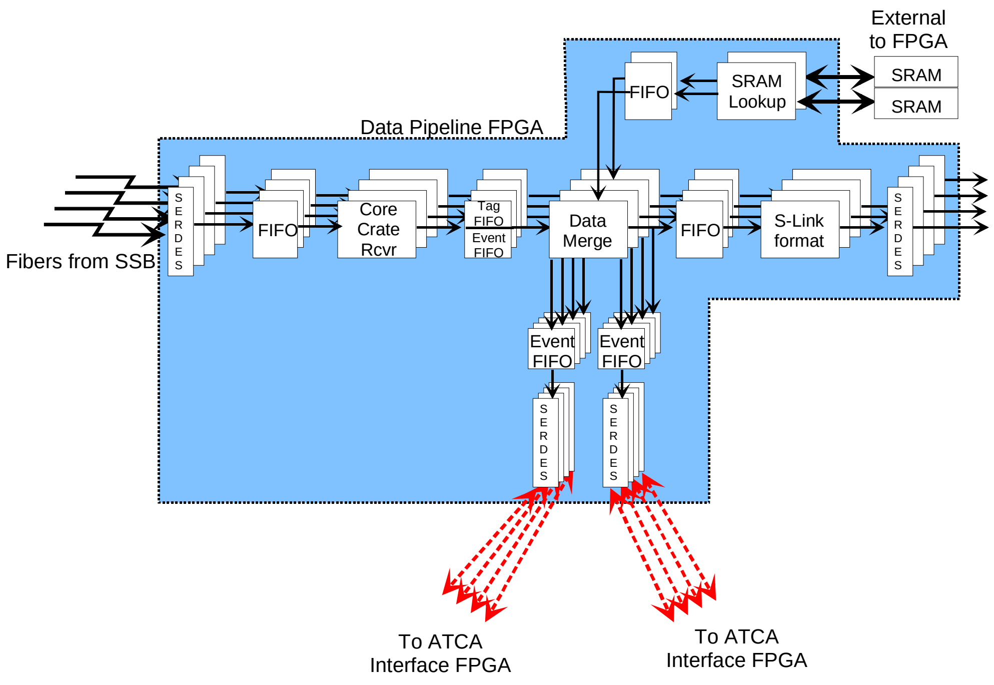
4.4.3 Support for future expansion

Two additional bus control architectures are supported by the Management FPGA to allow for potential needs in the future.

1) The PIC may command the Management FPGA to completely tri-state all FPGA address and data lines, during which time the Virtex-6 FPGAs may communicate with each other over the parallel bus. This mode is foreseen as providing a path for potential feature expansions in which processor blades within the ATCA shelf might communicate with the ATCA interface FPGAs "U3" and/or "U4", and then through those FPGAs perform register I/O across the yielded slow control bus to the other FPGAs of the FLIC.
2) The PIC may command the Management FPGA to interpret the expansion bus of the IPMC controller as address/data/CS lines, allowing for the future development of firmware within the IPMC to access registers of the Virtex-6 FPGAs and/or the flash RAMs. The exact format of the IPMC expansion bus for this scenario has not been defined.

5 DATA PROCESSING PIPELINE FIRMWARE

This section will expand upon the overview provided in section 3.5 and specifically explain what each state machine in the pipeline does. Flow control and error detection will also be detailed herein. For convenience a copy of Figure 15 from section 3.5 is provided below.



Copy of Figure 9

5.1 Serial interface to/from the SSB

The four SERDES GTX blocks of FPGA group 112 are used to both receive data from and send data to the SSB. The GTX core is configured to use standard 8b/10b encoding with the K28.5 comma character (0xBCBC). On the parallel side the data received is in 16-bit words; for a payload rate of 2Gbit/sec, this means that 16-bit words arrive at 125MHz. The GTX operates in clock resynchronization mode, meaning that the RX data will have randomly inserted comma bytes added as needed to continuously re-sync the receiver with the clock of the transmitter (SSB). The four transmit sections of the FLIC are run using a common clock but a unique RX clock is extracted for each link of the GTX quad. The raw data received by each link is stripped of commas and kept in proper byte order by a receiver state machine. The TX side of the GTX is used to transmit flow control and status information back to the SSB. A pseudo-random bit stream (PRBS) mode is available for link self-test.

5.1.1 Reception of data from the SSB

The comma-stripped data from each link is written to a 1K deep, 18-bit wide FIFO. The FIFO is read by the next state machine in the pipeline using the 200MHz clock common to all four pipelines. Data is written to the FIFO so long as the data is valid and the MOD_PRESENT signal from the SFP receiver is present. Sixteen of the 18 bits of the FIFO are used for SSB data. The extra two bits of the FIFO are used to hold "timing tag" bits that, if enabled, are set when the received data word is either

0x0000 or 0x5A5A respectively. These "timing tag" bits are carried in the extra bits of every FIFO buffer as the record propagates down the pipeline and are used in concert with Chipscope logic analyzer cores to provide a data-specific trigger.

It is expected that the "SFP Input FIFO" should normally be empty or nearly empty, as the Core Crate Receiver state machine that reads it runs at a faster clock rate. If, however, the FLIC pauses data processing in response to flow control from the ROS, this "SFP Input FIFO" will fill. While it is expected that the FLIC will pass the flow control message to the SSB, and the SSB will respond appropriately, a failure may result in the SFP Input FIFO becoming full. At that point data is lost. This condition sets a latch to inform the Core Crate Receiver machine that the incoming record is corrupted so that the Core Crate Receiver may flush the bad record and attempt to resynchronize after the SFP Input FIFO has been first reset by slow control and afterwards new data is received.

5.1.2 Transmission of status and flow control to the SSB

A control multiplexer is formed by the combination of two bits from the internal SFP_CTRL register of the FLIC and logic that monitors internal pipeline status. This control multiplexer allows the front panel fiber optic links of the FLIC to send either a PRBS link test pattern or SSB flow control information. At power-up the FLIC initializes to send SSB flow control data. In this state the FLIC normally transmits the K28.5 comma character all the time unless there is a status change.

5.1.2.1 Power-on handshaking

At power-up or recovery from a system reset the SSB should assume that the FLIC data reception pipelines is in an initialization state requiring a fake "priming" record to synchronize the data reception state machines of the FLIC. This FLIC requests this by sending a status word with bit 4 set. When in this initialization state the FLIC will send the data value 0x0000 when not requesting a fake record.

After initialization is complete software sets the FLIC into the 'run' state by changing control registers. In the 'run' state the FLIC sends comma characters at all times until a status change occurs. A single non-comma 16-bit word is transmitted whenever the state of the FLIC has changed. Each bit of the 16-bit status word is intended to be processed separately by the SSB, but the SSB is expected to process every bit each time a non-comma word is received. Each time a non-comma word is received the new status of each bit should be recorded as a word will be sent each time the collected status vector changes. A single word may contain multiple simultaneous state changes in both directions. Table 5 shows the currently implemented bits and what meaning should be imputed to receiving a word with the bit set as opposed to receiving a word with the bit clear. Bit 4 is shaded to highlight that this bit should normally only be used during setup.

Data bit in message to SSB	Meaning to SSB if new word sent with bit set	Meaning to SSB if new word sent with bit clear
15:5	No meaning associated with these bits (reserved)	No meaning associated with these bits (reserved)
4	FLIC requests that SSB send fake "priming" record	FLIC does not request fake "priming" record
3	FLIC reports that there is a problem with the S-Link interface to the ROS and thus the SSB should stop sending data soon as continued data will cause FIFOs to assert PROG_FULL.	No S-Link interface problems at back end of FLIC.
2	FLIC has detected spy buffer status requiring SSB	FLIC no longer detects any spy buffer status

	to cease transmission of data immediately.	requiring SSB to cease transmission of data.
1	FLIC has detected fatal overflow of SFP input FIFO. Data has been lost.	SFP input FIFO has been reset and is ready to receive data.
0	FLIC has detected pipeline FIFO prog-full condition. SSB should cease sending data immediately.	All FIFO buffers within FLIC no longer prog-full.

Table 5 – Summary of FLIC status bits to SSB

It is common for the FLIC to send a status word of 0x0008 followed relatively quickly by a status word of 0x0009 when records are flowing at a high rate. The 0x0008 has bit 3 set, indicating that XOFF has been received from the S-Link. The following 0x0009 has bits 3 and 0 set, indicating that the FLIC has responded internally to the XOFF from the ROS, but was still processing records in the pipeline, so one or more internal pipeline FIFOs are now asserting PROG_FULL. When the XON is received from the ROS a new status word of 0x0001 would indicate that the XOFF condition has cleared but one or more FIFOs is still PROG_FULL. When the pipeline flushes in response to XON, a later word of 0x0000 would follow the 0x0001 once all FIFOs hit PROG_EMPTY.

If the SSB responded to the 0x0008 or 0x0009 by stopping in the middle of a long record, it may be necessary for the SSB to finish the record in progress in response to the status of 0x0001 in order for the FLIC to finish flushing all the records in the pipeline. This is because each stage of the pipeline will not process until the FIFO buffer indicates that at least one full record is available for processing. If the SSB does not respond to the status of 0x0009 quickly enough, the status may change to 0x000B because the SSB has overflowed the relatively small SFP input FIFO. In general, the SSB should respond immediately to the value of 0x0009 and not wait until the end of the current record.

5.1.3 PRBS test mode

Multiplexer logic in each SERDES link, controlled by register bits, allows user to switch the TX side between the status/flow control data and a PRBS test data pattern. Similarly, the RX logic contains a PRBS-based data checker that may be enabled to run in parallel with the usual FIFO logic. The PRBS testing logic sends a test sequence of programmable length, broken into ‘chunks’ with a programmable number of comma characters sent between ‘chunks’. When the total length is complete, the PRBS logic sends the value of 0x0000 for a few clocks (as 0x0000 is not a legal PRBS value) to re-seed the PRBS checking logic in the receiver. Upon receipt of the 0x0000 characters, the next non-comma data value is used to seed a PRBS generator in the receiver. After re-seeding, the PRBS data value generated by the receiver is compared against the values received from the fiber and any errors noted. The PRBS test mode was used to test the bit error rate of the FLIC’s fiber interfaces to better than 1E-14 BER.

5.2 FIFO Buffers in the pipeline

The SFP Input FIFO connecting the SERDES RX logic to the Core Crate Receiver state machine is a simple first-word-fall-through FIFO buffer, as there is no understanding of data structure at the point of reception. However, all other FIFOs connecting the state machines together after the Core Crate Receiver are *Event-based FIFOs*. In this context, “event” is a generic term, identical to a *record* in FTK, used to describe a delineated block of data of unknown length. The Event-based FIFO structure takes advantage of the 18-bit wide parity structure of the Xilinx FPGA Block RAM primitive to store boundary information along with the 16-bit raw data. The Data Merge and SLINK formatter state machines wait for full records to be available in the FIFOs before beginning to process the information. Thus, at any given point, up to three records may be simultaneously processed by the pipeline (record "n" in the Core Crate Receiver, record "n-1" in the Data Merge and record "n-2" in the SLINK formatter).

The Event-based FIFO sub-design adds a small counter to the standard FIFO buffer, plus requires that the write-side state machine issue signals on the next-to-last (*event tag)* and last (*event end)* data words of the event to control the counter. Similarly the read-side machine is required to assert an *event read* signal when it begins reading the event. The Event-based FIFO asserts a signal *event available* if there are a non-zero number of complete events in the buffer. Similarly, the *event tag* signal is saved in one of the excess bits of the block RAM, such that the reading machine receives the *event tag* bit on the next-to-last word of the event. This simplifies the handling of events of variable size. A generic control is passed in to each instance of the Event FIFO throughout the pipeline selecting the appropriate counter handling based upon whether the input clock is faster, the output clock is faster or both run at the same speed.

5.3 The Core Crate Receiver

The Core Crate Receiver state machine verifies that the data being received is consistent with the FTK record format definition checks various fixed patterns in the data for correctness and performs simplistic comparisons upon the L1 ID values in the data to determine if records are to be tagged for copying to the ATCA Interface FPGAs. As noted in Section 3.1, data is expected to be formatted as ***records***, where each record consists of a *record header*, some number of *track data blocks*, and a *record trailer*, as defined in the ATLAS Fast Tracker (FTK) documentation. The Core Crate Receiver (CCR) runs at 200MHz. Data is read from the SFP input FIFO whenever it is not empty; every state of the CCR that processes a data word waits for the FIFO to become non-empty before proceeding.

The CCR machine copies the data from records that are correctly formatted to an output buffer, the CCMUX FIFO. A variety of status flags and error flags are generated by the CCR machine that connect to various counters, registers and also to the FLIC Status Word sent to the SSB. All of these bits are accumulated throughout the record and written as one word to the STATUS FIFO. A third FIFO, the TAG FIFO, is written with one word indicating whether the L1 ID of the record matched any of the match patterns. A 4th FIFO, the Address FIFO, is connected to the SRAM Lookup state machine. See Figure 20.

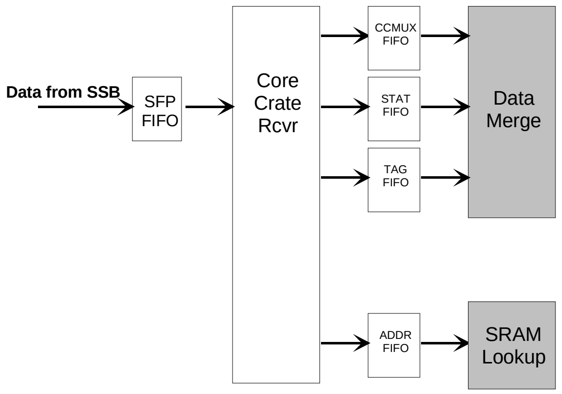


Figure 20 – details of Core Crate Receiver (one of four in pipeline FPGA)

5.3.1 Initial Synchronization

The Core Crate Receiver achieves synchronization by looking for a fixed sequence of four data words that are found at the end of the record, in the Record Trailer. The required sequence is 0xE0F0, 0xA5A5, 0x5A5A then 0x0E0F. All four words must match and be in the proper order to synchronize the machine. A status signal NOT_IN_SYNC is asserted until the CCR has synchronized to the incoming data. NOT_IN_SYNC is one of the conditions that is reported to the SSB through the status bits. Because the position of the fixed words has been relegated to the end of the record fragment, and not the beginning, the FLIC requires an initial "dummy" or "priming" record at initialization time. Once the CCR has initially synchronized it proceeds to processing the Record Header.

5.3.2 Error Detection in the Core Crate Receiver

The Core Crate Receiver provides two sets of error signals to the rest of the pipeline logic. A set of 'live' error bits persisting for only a single clock cycle are generated whenever the Core Crate Receiver has detected an error condition, for use in counters or with a Chipscope internal logic analyzer. In parallel, an eight-bit *Accumulated Status Vector* is collected throughout the processing of a record. At the end of each record the accumulated status vector is written into the Status FIFO such that there is one status byte for each record in the CCMUX data FIFO. The Data Merge machine is expected to read the status byte, add its own status, and write a wider accumulated status value into its own Status FIFO when the Data Merge process is complete.

The 'live' status bits asserted by the Core Crate Receiver are

- NOT_IN_SYNC is asserted whenever the state machine isn't synchronized to the data. This signal may persist for many clock cycles.
- HEADER_ERROR is asserted when the Record Header is malformed.
- TRACK_ERROR is asserted when the Track Header/Record Trailer word fails to match either pattern.
- TRAILER_ERROR is asserted if one or more of the synchronization words in the Record Trailer are malformed.
- TRUNCATION_ERROR is asserted if the record is too long and has been truncated.

The Accumulated Status Vector is similar to the live status bits. The exact format is shown in Table 6.

Bit	Meaning
0	HEADER_ERROR was asserted.
1	TRUNCATION_ERROR was asserted.
2	The SFP FIFO was overrun (went full).
3	TRACK_ERROR was asserted.
4	Error in the diagnostic block of the Record Trailer.
5	The record was tagged for copy to the Spy Buffer.
6	Fixed word error in Record Trailer.
7	The record contains a manufactured Record Trailer.

Table 6 – Format of the Core Crate Receiver Status Byte

5.3.3 Processing the Record Header

The FLIC checks the first word of the Record Header against the fixed pattern 0xB0F0. If the comparison fails, the CCR asserts the *HEADER_ERROR* bit and jumps back to initial synchronization. *A failure in the Record Header comparison will jettison the entire record.*

If the comparison is successful, then a fixed number of words equal to the expected length of the Record Header are copied from the SFP Input FIFO to the CCMUX FIFO[3]. During the copy function, the words in the Record Header that are assumed to contain the L1 ID value are compared against two L1 ID Match registers in a simple bitwise AND as a form of simplistic pre-selection of which records are to be marked in the Tag FIFO for copying to either or both of the ATCA Interface FPGAs. The expected format of the Record Header is shown in Table 7. Data within the FLIC is always processed at 16-bit words.

Bit=> Word	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RH01	B				0				F				0			
RH02	C				A				F				x			
RH03	F				F				1				2			
RH04	3				4				F				F			
RH05	0	Run Number [30:16]														
RH06	Run Number [15:0]															
RH07	Extended Level 1 ID [31:16]															
RH08	Extended Level 1 ID [15:0]															
RH09	Reserved															
RH10	Reserved				BCID [11:0]											
RH11	Reserved															
RH12	Reserved								Level 1 Trigger Type [7:0]							
RH13	Reserved								Detector Event Type [7:0]							
RH14	Reserved												TIM [3:0]			

Table 7 – Expected format of Record Header from SSB

As specified in the overall FTK data format document, the FLIC strips and throws away the first four words of the Record Header. This could be done at the Core Crate Receiver level, but is deferred to the next step of the pipeline for purposes of supporting the copy of raw input data through the Spy Buffer path.

5.3.4 Processing Track Data

After the copy of the Record Header is complete the CCR checks to see whether the fixed bits of the next word are equal to those expected for a Track Header or for the first word of the Record Trailer; this is required because records with zero tracks are defined as acceptable. If the pattern of the word

[3] As of 20160826 the Core Crate Receiver does not copy the first four words of the Record Header (the 0xB0F0, 0xCAFx, 0xFF12, 0x34FF) but does copy the rest. This is because the words of the Record Header are interpreted by the higher level DAQ as part of the S-Link header and the header size the FLIC is required to report does not include these four values. However, there is a competing specification that states that the Spy Buffer must save the unmodified raw input data, so at some future point the stripping of these four words may move from the Core Crate Receiver to the Data Merge machine.

matches that of the Track Header, then the data following is assumed to be track data; if it matches the fixed pattern of the Record Trailer the machine jumps to a different set of states to process that information. If the word fails to match either pattern, the *TRACK_ERROR* bit is set and the Core Crate Receiver terminates the record, writing Record Trailer of its own (see section 5.3.6). The Track Header and following data is expected to follow the format shown in Table 8. Track data consists of the Track Header (orange), the Pixel layer data (bright yellow) and the Silicon layer data (pale yellow).

Bit=> Word	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TH1	Res [3:0]			L	B				D				A			
TH2	SECTOR NUMBER															
TH3	Reserved				TF#				Res	Tower Number						
TH4	Reserved				Layer Map [11:0]											
TH5	Reserved								ROAD ID [23:16]							
TH6	ROAD ID [15:0]															
TH7	TRACK CHISQ															
TH8	TRACK D0															
TH9	TRACK Z0															
TH10	TRACK COTTH															
TH11	TRACK PHI0															
TH12	TRACK CURV															
IBLa	0	COL WIDTH			COL COORDINATE [11:0]											
IBLb	s	ROW_WIDTH			ROW COORDINATE [11:0]											
PL0a	0	COL WIDTH			COL COORDINATE [11:0]											
PL0b	s	ROW_WIDTH			ROW COORDINATE [11:0]											
PL1a	0	COL WIDTH			COL COORDINATE [11:0]											
PL1b	s	ROW_WIDTH			ROW COORDINATE [11:0]											
PL2a	0	COL WIDTH			COL COORDINATE [11:0]											
PL2b	s	ROW_WIDTH			ROW COORDINATE [11:0]											
SAx0	0	HIT2 WIDTH			ReV	HIT2 COORDINATE [10:0]										
SSt0	0	HIT1 WIDTH			ReV	HIT1 COORDINATE [10:0]										
SAx1	0	HIT2 WIDTH			ReV	HIT2 COORDINATE [10:0]										
SSt1	0	HIT1 WIDTH			ReV	HIT1 COORDINATE [10:0]										
SAx2	0	HIT2 WIDTH			ReV	HIT2 COORDINATE [10:0]										
SSt2	0	HIT1 WIDTH			ReV	HIT1 COORDINATE [10:0]										
SAx3	0	HIT2 WIDTH			ReV	HIT2 COORDINATE [10:0]										
SSt3	0	HIT1 WIDTH			ReV	HIT1 COORDINATE [10:0]										

Table 8 – Expected format of Track data (header, pixel and silicon)

The Core Crate Receiver machine uses the "L" bit from the first word of the Track Header plus all 16 bits of the 2nd word of the Track Header (*sector number)* to develop a 17-bit SRAM Base Address. This SRAM Base Address is loaded into the Address FIFO (see Figure 20) and is used by the SRAM Lookup state machine to fetch Module ID data from the external SRAMs. This is discussed further in Section 5.4.

As each track is processed the data is copied to the CCMUX FIFO without modification. This process goes on ad infinitum until a Record Trailer is seen or a *TRACK_ERROR* occurs. Because the size of the CCMUX FIFO is not infinite, the Core Crate Receiver implements a maximum number of tracks allowed per record (presently set via a hard constant to 144). If the number of tracks in the record from the SSB exceeds this amount, the CCR machine asserts the *TRUNCATION_ERROR* flag and ceases writing to both the CCMUX and Address FIFOs, truncating the record at the end of the track. Event processing resumes when the Record Trailer of the excessively long record is seen.

5.3.5 Processing Record Trailers

The Record Trailer is again identified by a word with the fixed value 0xE0DA. As the Core Crate Receiver loops through tracks, each expected Track Header word is, if not identified as a Track Header, checked against 0xE0DA to identify the Record Trailer. The Record Trailer has a "debug block" of indeterminate length, starting with the 0xE0DA key word and ending with a value of 0xE0DF. Immediately following the 0xE0DA and 0xE0DF words are words containing the expected length of the debug block, as seen in Table 9. The Core Crate Receiver uses the first instance of the debug length to set a counter and copies that many words without processing as the debug data is indeterminate. When the count elapses the data is checked for the 0xE0DF value and the 2nd copy of the debug length is compared against the first copy. If the received debug block is of the incorrect length, the *TRAILER_ERROR* flag is set. The Core Crate Receiver then stops reading data from the SFP FIFO and attempts to generate a correctly formatted trailer as best it can.

Bit=> Word	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RTF1	E				0				D				A			
RTF2	Length of Debug Block												🟥			
RTFD(1)	debug information															
RTFD(N)	debug information															
RTF3	E				0				D				F			
RTF4	Length of Debug Block												🟥			
RTF5	L1ID [31:16]															
RTF6	L1ID [15:0]															
RTF7	Error Flag [31:16]															
RTF8	Error Flag [15:0]															
RTF9	Reserved															
RTF10	Reserved															
RTF11	Reserved															
RTF12	Reserved															
RTF13	E				0				F				0			
RTF14	A				5				A				5			
RTF15	5				A				5				A			
RTF16	0				E				0				F			

Table 9 – Expected format of Record Trailer from SSB

If the "debug block" is the correct length, then words 5-12 of the trailer are copied to the CCMUX FIFO without processing.

Words 13-16 of the Record Trailer are the record-level synchronization data, the same that is used at initialization. The Core Crate Receiver checks all four of these words at the end of the Record Trailer, asserting the *TRAILER_ERROR* condition if any mismatch is found. If a mismatch of any of these words is found, the Core Crate Receiver stops reading data from the SFP FIFO and forcibly generates the rest of the Record Trailer correctly from this point on to make the CCMUX data as usable as possible.

5.3.6 "Manufactured" Record Trailers in response to errors

The Core Crate Receiver state machine responds to certain classes of errors by jumping into a set of states collectively referred to as *Push_Trailer.* This set of states stops reading the SFP input FIFO, generates a pre-defined Record Trailer to terminate the record, and then transitions back to the initial synchronization states. This has the effect of flushing data until a properly formatted Record Trailer is found, after which normal processing resumes. Typically, this results in the remainder of the erroneous record being discarded *plus the possibility of also completely flushing the next record to resynchronize.*

Whether the next record is lost to resynchronization or not depends upon which of the error conditions has occurred.

- If a word expected to be either a Track Header or the first word of a Record Trailer fails to match either pattern, the FLIC will manufacture a complete Record Trailer and then attempt to resynchronize. Thus, if the error was in a Track Header but the Record Trailer of the errant record is ok, no records will be lost. Similarly, if the error was in first word of the Record Trailer, resynchronization will likely occur on the last four words of the Record Trailer of the errant record.
    - Events that have the wrong number of tracks (too few or too many) will flag the error on the next expected Track Header word.
- Errors in the debug block of the Record Trailer will typically not lose a record as the necessary fixed values for resynchronization are later in the Record Trailer.
- Errors in the fixed values of the Record Trailer that are used for synchronization will cause the loss of the next record.

The FLIC attempts to mark any manufactured Record Trailer by setting bit 31 of the Error Flags word of the Record Trailer. This bit is reserved by the firmware solely for the indication of a manufactured trailer and may not be used to indicate any other type of error. The full map of error bits generated by the FLIC is found in a later section of this document.

5.4 SRAM Lookup Machine

The SRAM Lookup machine performs the lookup of FTK *Module ID* values from an external SRAM buffer for use in the Data Merge process as shown in Figure 21. Each record processed by the Core Crate Receiver generates a single 17-bit base address for every track within the record. These base addresses are extracted from the Track Header information and stored as they are extracted in the Address FIFO. The SRAM Lookup machine continuously polls the Address FIFO and when a base address is available a lookup occurs.

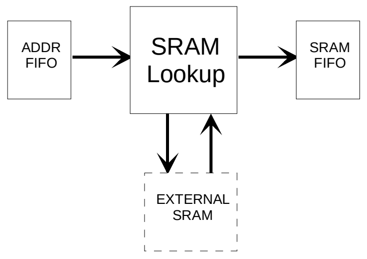


Figure 21 – SRAM lookup machine detail

In response to receipt of a base address, the SRAM Lookup state machine uses the base address as the upper 17 bits of a 21-bit SRAM address that is asserted to an external CY7C1071DV33 (2,097,152 x 16) static RAM. These RAMs have an access time of 12ns and this is the limiting factor defining the overall speed of the pipeline.

Upon asserting the base address the SRAM Lookup machine performs 12 reads of the SRAM at sequential addresses, forming the lower four bits of the address from an internal counter. These 12 words are then written into the SRAM FIFO and are used by the Data Merge machine to form the output record.

5.4.1 Timing Considerations

Each record processed by the Core Crate Receiver has an indeterminate number of tracks. Each track has a track header, as shown in Table 8. There are 28 words in a track. At a processing speed of 200MHz (5ns per word), each track requires 140ns to process, meaning that a new word is loaded into the Address FIFO every 140ns. In response to a base address the SRAM Lookup machine must fetch 12 words from the SRAM. An 80MHz clock is used for the SRAM Lookup machine (12.5ns period, to match the 12ns access time of the SRAM). Reading 12 words thus requires 12*12.5, or 150ns. In addition the SRAM Lookup machine requires a clock or two for setup of the data transfer.

This difference between the track processing time and the SRAM lookup time means that as the record is processed the SRAM lookup machine slowly falls behind the Core Crate Receiver. However, the Data Merge machine does not start to process the record until the record is completely stored in the CCMUX FIFO. This means that the SRAM lookup gains time to catch up as the Merge machine processes the Record Header. As shown in Table 7 the Record Header is 14 words long, requiring some 70ns to process. The SRAM lookup also recovers time to fetch additional information as the Data Merge machine processes the tracks previously looked up. The net effect of this pipelining is that the SRAM lookup process only slows records with relatively large numbers of tracks; the break-even point is in the 20s of tracks per record.

5.5 Data Merge Machine

The Data Merge machine reads the data from the CCMUX FIFO and the SRAM FIFO (output of the SRAM lookup machine), and merges the data together into the Merge FIFO for eventual transmission over S-Link to the ROS. While the machine is running it reads the accumulated status byte from the Core Crate Receiver's STAT FIFO, accumulates a 2nd byte of Merge machine status, and at the end of the record a 16-bit combined status is written to the Merge machine's STAT FIFO. The TAG FIFO is read by the Merge machine when processing of the record within the CCMUX FIFO begins; the value read from the TAG FIFO determines whether the data read from the CCMUX FIFO is copied verbatim into either or both of the two SERDES links that run from the Pipeline FPGAs to the ATCA Interface FPGAs. This connectivity is summarized in Figure 22.

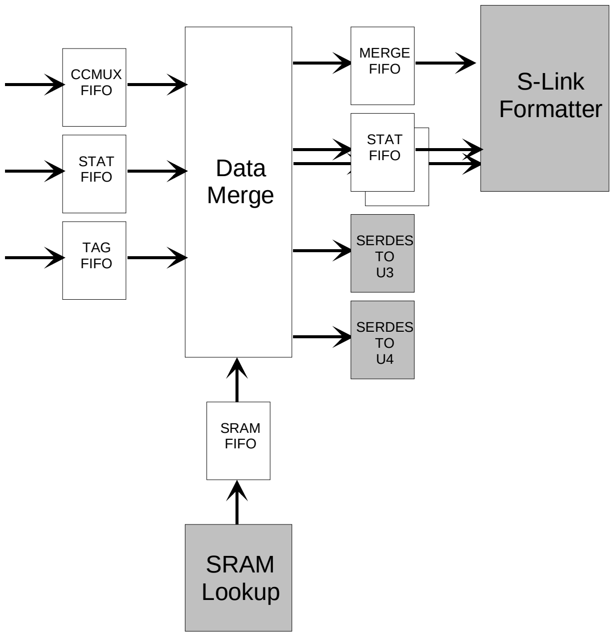


Figure 22 – Overall connectivity of the Data Merge machine

The Data Merge waits for entire records to be stored in the CCMUX FIFO before it begins to process them. As soon as a record is available (assertion of signal EVENT_AVAILABLE from the Event-based FIFO as described in section 5.2) the state machine starts pulling the record. Unlike the Core Crate Receiver, where the machine must constantly poll the FIFO Empty signal to correctly cross clock domains, the Data Merge need only check that a record is available at record boundaries.

Immediately upon sensing that a record is available, the Merge machine copies the entire Record Header section of the data from the CCMUX FIFO to the Merge FIFO. The L1 ID match flags from the Tag FIFO are sampled immediately and then the Tag FIFO advanced in preparation for the next record. The L1 ID match flags control the write enable to the input FIFOs of the two SERDES blocks that go to U3 and U4. If the L1 ID match flag is set, the read enable of the CCMUX FIFO becomes the write enable to the FIFO of the SERDES link, causing a direct and un-modified copy of the input data to the Merge machine to be sent over the internal mesh to the appropriate ATCA Interface FPGA.

After counting off the number of words expected for the Record Header, the next word is checked to determine whether it is a Track Header or a Record Trailer, similar to the same comparison that is done in the Core Crate Receiver. If a Track Header, the next set of words from the CCMUX FIFO is merged with the data from the SRAM FIFO to generate the reformatted track data with inserted Module ID values. If a Record Trailer, the machine enters a set of states that copy the trailer and debug information. If neither match occurs, that should only happen if the record is so malformed that the Core Crate Receiver can't reconstitute it with a manufactured trailer, the Merge machine simply copies the rest of the data from the CCMUX FIFO to the Merge FIFO verbatim until the end-mark of the record is seen in the CCMUX FIFO.

5.5.1 Merging of SRAM data with track data

For each Track Header that is identified, the Merge machine copies the Track Header verbatim from the CCMUX FIFO to the Merge FIFO, and then checks to see if the SRAM FIFO is ready. As noted previously in section 5.4.1, records with a sufficiently large number of tracks will at some point require the Merge machine to stop momentarily to allow the SRAM Lookup machine to catch up. This is done on a track by track basis. As soon as the SRAM data is available the Merge machine resumes.

5.5.1.1 Merge process for Pixel layers

The first four layers of the detector are Pixel (IBL) layers. Each of these layers is defined by two 16-bit words in the CCMUX FIFO as shown in Table 10.

IBLa	0	COL WIDTH	COL COORDINATE [11:0]
IBLb	s	ROW_WIDTH	ROW COORDINATE [11:0]
PL0a	0	COL WIDTH	COL COORDINATE [11:0]
PL0b	s	ROW_WIDTH	ROW COORDINATE [11:0]
PL1a	0	COL WIDTH	COL COORDINATE [11:0]
PL1b	s	ROW_WIDTH	ROW COORDINATE [11:0]
PL2a	0	COL WIDTH	COL COORDINATE [11:0]
PL2b	s	ROW_WIDTH	ROW COORDINATE [11:0]

Table 10 – Pixel/IBL layer data pre-merge

The Merge machine reads one 16-bit Module ID value from the SRAM FIFO for each layer. Since FTK has defined all ROS data to be 32-bit objects, additional padding is required to expand each pixel/IBL layer to four 16-bit words in the Merge FIFO as shown in Table 11.

IBLa	Reserved		
IBLb	Reserved		Module ID[11:0]
IBLc	0	COL WIDTH	COL COORDINATE [11:0]
IBLd	s	ROW_WIDTH	ROW COORDINATE [11:0]
PL0a	Reserved		
PL0b	Reserved		Module ID[11:0]
PL0c	0	COL WIDTH	COL COORDINATE [11:0]
PL0d	s	ROW_WIDTH	ROW COORDINATE [11:0]
PL1a	Reserved		
PL1b	Reserved		Module ID[11:0]
PL1c	0	COL WIDTH	COL COORDINATE [11:0]
PL1d	s	ROW_WIDTH	ROW COORDINATE [11:0]
PL2a	Reserved		
PL2b	Reserved		Module ID[11:0]
PL2c	0	COL WIDTH	COL COORDINATE [11:0]
PL2d	s	ROW_WIDTH	ROW COORDINATE [11:0]

Table 11 – Pixel/IBL layer data post-merge

All “reserved” fields are zero. The rest of the data is a copy operation from one FIFO or the other.

5.5.1.2 Merge operation for Silicon layers

Merging of the Silicon layer data is somewhat different because unique Module ID values are associated with both the Axial and Stereo portions. Thus, each of the eight Silicon layer 16-bit words is expanded to a single 32-bit word in its own merge operation, as shown in Table 12 and Table 13.

SAx0	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]
SSt0	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]
SAx1	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]
SSt1	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]
SAx2	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]
SSt2	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]
SAx3	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]
SSt3	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]

Table 12 – Silicon layer data pre-merge

SAx0a	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]
SAx0b	Reserved			Module ID[12:0]
SSt0a	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]
SSt0b	Reserved			Module ID[12:0]
SAx1a	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]
SAx1b	Reserved			Module ID[12:0]
SSt1a	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]
SSt1b	Reserved			Module ID[12:0]
SAx2a	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]
SAx2b	Reserved			Module ID[12:0]
SSt2a	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]
SSt2b	Reserved			Module ID[12:0]
SAx3a	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]
SAx3b	Reserved			Module ID[12:0]
SSt3a	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]
SSt3b	Reserved			Module ID[12:0]

Table 13 – Silicon layer data post-merge

After merging all tracks with Module IDs, when the record trailer is seen it too is copied from input to output.

5.5.2 Merging of the Record Trailer

When the Record Trailer is processed the Merge machine performs a second check on the debug block to verify that the start/end marks are present when expected and that the two copies of the length match. If any errors are seen the Merge machine falls to the PULL_TO_END state and flushes the record as-is from that point. If no error in the debug block is found then the Merge machine follows specifications and strips the last four words (0xE0F0, 0xA5A5, 0x5A5A & 0x0E0F) from the Record Trailer as these words only exist for Core Crate Receiver synchronization purposes.

5.5.3 Error accumulation and forwarding in the Merge machine

The Merge machine reads the accumulated status vector from the Core Crate Receiver from the STAT FIFO when processing of a record begins. A merge-specific eight-bit accumulated status vector is generated during the Merge machine processing. At the end of the record both status bytes are written to a pair of STAT FIFOs (one forwarding the Core Crate Receiver status and the 2nd holding the Merge status) to form a 16-bit accumulated status vector.

The 16-bit accumulated status vector from the Merge machine is defined in Table 14.

Bit	Meaning	Notes
0	HEADER_ERROR was asserted.	From CCR
1	TRUNCATION_ERROR was asserted.	From CCR
2	The SFP FIFO was overrun (went full).	From CCR
3	TRACK_ERROR was asserted.	From CCR
4	Error in the diagnostic block of the Record Trailer.	From CCR
5	The record was tagged for copy to the Spy Buffer.	From CCR
6	Fixed word error in Record Trailer.	From CCR
7	The record contains a manufactured Record Trailer.	From CCR
8	Event tag from CCMUX received too early	From Merge
9	Merge machine had error, flushed record	From Merge
10	Word expected to be Track Header or Record Trailer was neither (similar to bit 3)	From Merge
11	0xEODF at start of debug block not found where expected. (similar to bit 4)	From Merge
12	Two copies of debug block length did not match	From Merge
13	Record was copied to ATCA Interface FIFO U3	From Merge
14	Record was copied to ATCA Interface FIFO U4	From Merge
15	Merge FIFO asserted PROG_FULL during record	From Merge

Table 14 – Combined status vector from Merge machine

5.6 S-Link Formatter

The S-Link Formatter state machine is responsible for adding the S-Link header and S-Link footer to the data record from the Merge machine before forwarding the record to the GTX 116 (rear transition module) serial transmitter. The S-Link header is a fixed set of data values as shown in Table 15.

SH01	Start of Header Marker - Bytes 3-2, Fixed Value 0xEE12
SH02	Start of Header Marker - Bytes 1-0, Fixed Value 0x34EE
SH03	Header Size - Bytes 3-2, Fixed Value 0x0000
SH04	Header Size - Bytes 1-0, Fixed Value 0x0009
SH05	Format Version Number - Bytes 3-2
SH06	Format Version Number - Bytes 1-0
SH07	Source Identifier - Bytes 3-2
SH08	Source Identifier - Bytes 1-0

Table 15 – S-Link fixed header

The S-Link formatter machine waits for a record to be available in the Merge FIFO before proceeding. Upon notification that the Merge machine has processed a full record the machine writes the 8 fixed words of the S-Link header and then proceeds to copy the data from the Merge FIFO. The data from the Merge FIFO is copied verbatim into the RTM FIFO before entering the SERDES block containing the S-

Link (HOLA) core firmware from CERN, as shown in Figure 23. During the copy the number of words copied is counted.

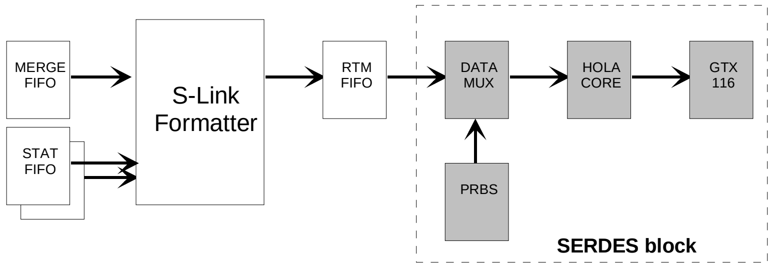


Figure 23 – S-Link Formatter and SERDES block connection details

5.6.1 Error masking and the S-Link footer

A top-level monitoring process collects an additional 16 bits of status information based upon the overall status of the pipeline. These 16 top-level bits are merged with the 16 bits of status fed forward through the STAT FIFOs to create a 32-bit status vector within the S-Link Formatter. The full 32-bit vector is given in Table 16.

Bit	Meaning	Notes
0	HEADER_ERROR was asserted.	From CCR
1	TRUNCATION_ERROR was asserted.	From CCR
2	The SFP FIFO was overrun (went full).	From CCR
3	TRACK_ERROR was asserted.	rom CCR
4	Error in the diagnostic block of the Record Trailer.	From CCR
5	The record was tagged for copy to the Spy Buffer.	From CCR
6	Fixed word error in Record Trailer.	From CCR
7	The record contains a manufactured Record Trailer.	From CCR
8	Event tag from CCMUX received too early	From Merge
9	Merge machine had error, flushed record	From Merge
10	Word expected to be Track Header or Record Trailer was neither (similar to bit 3)	From Merge
11	0xEODF at start of debug block not found where expected. (similar to bit 4)	From Merge
12	Two copies of debug block length did not match	From Merge
13	Record was copied to ATCA Interface FIFO U3	From Merge
14	Record was copied to ATCA Interface FIFO U4	From Merge
15	Merge FIFO asserted PROG_FULL during record	From Merge
16	SFP input FIFO PROG_FULL status bit	Collected at top
17	CCMUX FIFO PROG_FULL status bit	Collected at top
18	SRAM FIFO PROG_FULL status bit	Collected at top
19	MERGE FIFO PROG_FULL status bit	Collected at top
20	RTM FIFO PROG_FULL status bit	Collected at top
21	COLLECTED_PIPELINE PROG_FULL (all above plus user bit)	Collected at top
22	U3_TAGGED_EVENT_FIFO PROG_FULL	Collected at top
23	U4_TAGGED_EVENT_FIFO PROG_FULL	Collected at top
24	S-Link LDOWN flag (inverted polarity, high is link down, set on falling edge)	Collected at top
25	S-Link LFF flag (inverted polarity, high is link FIFO full, set on falling edge)	Collected at top
26	S-Link is asserting XOFF	Collected at top
27	Core Crate Receiver not in sync	Collected at top
28	User defined status/error bit from register	Collected at top
29	User defined status/error bit from register	Collected at top

30	Reserved	
31	Reserved for status collection failure bit	S-Link formatter

Table 16 – Full 32-bit accumulated status vector from FLIC

This full 32-bit vector is bit-by-bit ANDed with a 32-bit *ERROR MASK* register to allow slow control selection of which bits are to be interpreted as ***errors*** as opposed to mere ***status.*** The ATLAS data acquisition system allows for only two kinds of S-Link records:

- "Normal" records have a status length of **zero** and may report no status of any kind;
- "Error" records have a status length of exactly **one** and may report a single 32-bit word of combined status/error.

Because there is no provision for status reporting within "normal" records the FLIC collects status on each record and allows the user to define which, if any, of the status bits constitutes an error condition requiring the trigger to read and analyze the 32-bit status vector.

5.6.1.1 S-Link footer for records with no error

If the masking operation results in a record being declared as "normal" the S-Link Formatter issues a fixed-format S-Link footer as defined in Table 17.

SLF1	Number of Status Elements - Bytes 3-2, 0x0000
SLF2	Number of Status Elements - Bytes 1-0, 0x0000
SLF3	Number of Data Elements, Bytes 3-2
SLF4	Number of Data Elements, Bytes 1-0
SLF5	Status Block Position - Bytes 3-2, Fixed Value 0x0000
SLF6	Status Block Position - Bytes 1-0, Fixed Value 0x0001

Table 17 – S-Link footer generated for "normal" records

5.6.1.2 Added data and S-Link footer for records with error

If the masking operation results in an record being declared "in error" (non-zero result after AND), the S-Link Formatter issues the 32-bit accumulated status vector as two added words after the Record Trailer and issues a fixed-format S-Link footer as defined in Table 18.

RTF13	FLIC Status (format TBD)
RTF14	FLIC Status (format TBD)
SLF1	Number of Status Elements - Bytes 3-2, 0x0000
SLF2	Number of Status Elements - Bytes 1-0, 0x0001
SLF3	Number of Data Elements, Bytes 3-2
SLF4	Number of Data Elements, Bytes 1-0
SLF5	Status Block Position - Bytes 3-2, Fixed Value 0x0000
SLF6	Status Block Position - Bytes 1-0, Fixed Value 0x0001

Table 18 – S-Link footer generated for "error" records

5.6.1.3 Special S-Link Footer generated upon failure of status collection subsystem

If there is a failure in the status collection subsystem such that no accumulated status is passed forward through the STAT FIFOs, the S-Link Formatter generates a special version of the "error" footer in which the FLIC Status value is explicitly defined as 0xF0F0F0F0. This is the only case in which the FLIC status value will have bit 31 set, and thus may be uniquely identified.

5.7 Implementation of CERN S-Link (HOLA) core

The data written to the RTM FIFO by the S-Link Formatter machine is, as is the case for all other FIFOs of the processing pipeline, written as 16-bit data words. However, this does not match with the specifications for the S-Link (HOLA) core firmware provided by CERN. This firmware was written assuming the use of a 32-bit interface. This 32-bit data is then written into a short FIFO within the CERN code. The interested reader may note that the CERN code then takes the 32-bit data and immediately de-multiplexes it to 16-bit data before processing further, and ask why the FLIC takes 16 bit data, multiplexes it into 32-bit data, only to have the CERN code immediately convert back to 16 bit width. The answer to this query is that the FLIC firmware was developed under an agreement in which the CERN code was required to be used "as-is" with no modifications.

Inside the FLIC a small reset controller state machine is implemented that resets the CERN code in accordance with its requirements. Once this is performed the S-Link code then figures out whether the link is "up" or "down" based upon successful reception of specific K characters in the data stream. The S-Link code emulates the TLK2501 transceiver chip from Texas Instruments that was used on the ODIN and HOLA cards; the reader is advised to download and study this data sheet before attempting to understand the S-Link code.

The GTX116 block of the FLIC pipeline firmware implements a data multiplexer that allows selection between transmitting S-Link data and PRBS test data, identical to that implemented for the front-side GTX112 transceiver block; see Figure 23. The PRBS mode is typically only used for board checkout and repair. While nominally the front-side GTX112 and back-side GTX116 transceivers run at the same rate, they have separately controlled reference clocks. Thus, if the S-Link firmware is at some point upgraded to run at a faster speed, the FLIC may be reprogrammed to match.

The main diagnostic signals out of the S-Link firmware are LDOWN (Link DOWN) and LFF (Link FIFO Full). There is also a flow control signal XOFF. The FLIC firmware will not attempt to send data if any of these conditions are present. This can make diagnosis of problems difficult, so a register is implemented with a FORCE LINK UP bit to tell the FLIC that the S-Link is operational irrespective of the state of the link. In the FORCE LINK UP mode the data is read from the RTM and, instead of being sent to the S-Link code, is simply thrown away.

5.8 Flow Control in the processing pipeline

The "forward" flow of data from SSB to S-Link in the pipeline is mirrored by a "backwards" flow of status information. In the top-level code of the Pipeline FPGA, there is a status collection process for each pipeline that monitors all FIFO buffers. This process collects the PROG_FULL, PROG_EMPTY and S-Link status into three bit vectors named COLLECTED_PROG_FULLs, COLLECTED_PROG_EMPTYs and COLLECTED_OTHER_STOPs. Manual register controls allow the user to generate a "fake" PROG_FULL and a "fake" PROG_EMPTY for testing of the flow control logic. These collected status bits feed into bit 0 of the status reported back to the SSB as detailed in section 5.1.2.

Additionally, each state machine (Core Crate Receiver, Data Merge, S-Link Formatter) monitors the PROG_FULL of the FIFO it writes to, and if the destination FIFO is asserting PROG_FULL the machine will enter a wait state upon completing the record it is currently working on.

5.9 Event Selection Methodology

The pipeline processor FPGA code for each of the two data processing FPGAs implements two registers for record selection. As records are received by the CoreCrateRcvr state machine, the two registers allow for selection of which records are to be copied to the Spy Buffer logic implemented in the two ATCA Interface FPGAs "U3" and "U4". The selection registers implement a bit-by-bit 'AND' methodology based upon the Level 1 ID value of the record. The 23 bits of the Level 1 ID are bit-by-bit ANDed with a 24-bit field in the register, and if the result of the bit-by-bit AND is exactly zero, a match is made. Once a match by AND pattern has been made, the logic will then copy the next 'n' records to the spy buffer based upon a count value field within the selection register. A secondary mode is supported in which only records with the Level 1 ID of exactly zero are copied.

5.9.1 Tag FIFO Usage

The CoreCrateRcvr state machine in each of the four pipelines of a processing FPGA generates two bits for every record as described above, one for the "U3" match and one for the "U4" match. These two bits are written to the Tag FIFO, a small FIFO buffer parallel to and distinct from the Event FIFO that is the normal output destination of the CoreCrateRcvr data. Two additional bits, defining record status, are written to the Tag FIFO for a total of four bits per record. The Tag FIFO is sufficiently deep to ensure that the number of records that can be held by the Tag FIFO is greater than any foreseen number of records in the Event FIFO between the Core Crate Receiver and Merge state machines.

The Merge state machine reads one four-bit nibble from the Tag FIFO as a preliminary step that occurs after an record is available for processing in the Event FIFO but before any of the data is read from the CCMUX FIFO. The Merge machine uses the four-bit value from the Event Tag FIFO to control how the record that is about to be read is processed.

The two status bits in the nibble define the overall status of the record:

- A value of "00" indicates Normal Record (no errors).
- A value of "01" indicates Truncated Record (record may be processed but was truncated).
- A value of "10" indicates Malformed Record (record is damaged)
- A value of "11" indicates a Malformed Record in a way that did not require resynchronization, but was simply truncated at the point of error.

As of this writing the Data Merge machine has access to but does not use the status bits to determine whether the record is to be copied to the Spy Buffer logic; only the two Tag bits are used.

5.9.2 Data Transfer between FPGAs

Events tagged for transfer between FPGAs will be transmitted using a simple copy process that does not reformat the data in any way, but shall add a single header word that precedes the copied data. The header word shall have the value 0xBExy where bits 7:4 are available to be driven by a register, and bits 3:0 shall contain the Event Tag information. The data received by the fabric FPGA will be the identical data that the CoreCrateRcvr sends to the Merge machine, and thus will have the format shown in Figure 12.

5.9.3 Pipeline-level selection control in ATCA Interface FPGAs

Each of the eight independent pipelines in the two Pipeline FPGAs transmits the selected records to the ATCA Interface FPGAs using separate and independent serial links. The ATCA Interface FPGAs then implement a pipeline-level selection mask to control which of these eight independent streams of fragments are to be assembled into the final packet sent over the ATCA backplane. This selection mechanism allows removal of non-working SSB-FLIC links from the Spy Buffer process.

6 ATCA Interface Firmware

The ATCA Interface FPGAs receive and assemble fragments that have been tagged for spy buffer readout by Data Pipeline FPGAs. Each ATCA Interface FPGA, U3 and U4, four SERDES links connect to U1 and four links connect to U2. This provides a dedicated link for transferring tagged record fragments (records) from each SSB input to each ATCA Interface FPGA.

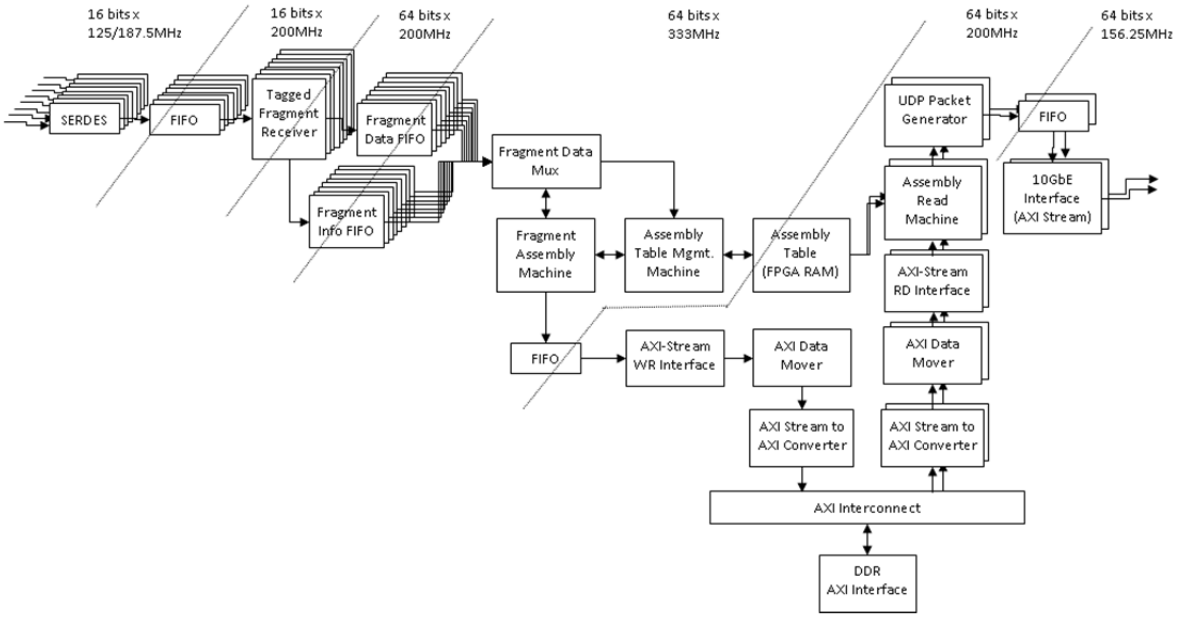


Figure 24 – ATCA Interface FPGA design overview

Each input link implements a Tagged Record Receiver to perform format checking and extract fragment information required for the fragment assembly into the Fragment Info FIFO, while transferring the unmodified data into the Fragment Data FIFO. These FIFOs are then processed by the Fragment Assembly Machine.

The Fragment Assembly Machine continuously scans the input Fragment FIFOs for fragment data, and uses the extracted Level 1 ID field (L1 ID) from each record as defined in Section 5.3.3) and input channel ID to sort them into the DDR. The DDR memory is divided into 16 fragment assembly regions. Each region is allocated 8MB of space, which is far more than required. These buffers are filled sequentially. The first fragment written a region sets the region’s L1 ID. As additional are received their L1 IDs are either matched to an existing assembly region or assigned to the next available region. There is no ordering of fragments within an assembly buffer. A buffer region is released for reuse once its data is sent to the processing blade. If a new L1 ID is encountered and no assembly buffers are available, the oldest working buffer will be forced to transfer to the blade. Otherwise, buffers are only transferred once the expected SSB channels (as set by the user) have reported fragments.

Once all fragments have been assembled within a DDR region, the data is read out and merged with an assembly header and footer prior to being packed into UDP packets for transmission to processor blades over the ATCA Fabric Interface. Each of the Interface FPGAs has two 10GbE links, for a total of four per FLIC. Each 10GbE link has its own assembly handling pipeline for transferring data from the DDR to the processing blade.

6.1 Inter-FPGA SERDES Transceiver

The design of the ATCA Interface FPGA’s GTX transceiver is the same as implemented in the front panel SFP interfaces of U1 & U2, except for the FIFO implementation.

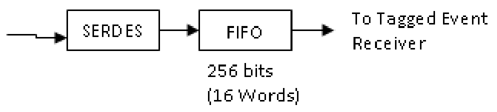


Figure 25 – Inter FPGA GTX design overview

Each GTX runs as four independent links, and supports the same diagnostic modes as U1/U2. GTX113 connects to U2 and GTX114 connects to U1. GTX112 cross connects U3 and U4, but is unused during normal operation. The ATCA Interface FPGA’s never transmit data back to U1 or U2, and as such will never assert flow control to the Data Pipeline FPGAs.

GTX 115 and GTX 116 are handled directly by the Xilinx 10GbE core, their design is covered in external documentation.

6.1.1 FIFOs

The FIFOs in the ATCA Interface FPGA’s GTX transceivers provide a generic parameter in the firmware to independently specify the read and write depth. The Tagged Record Receivers operate at 200 MHz which is sufficient to continuously process fragment data at either 2 to 3Gbps. By design, the Tagged Record Receivers are always reading the GTX RX FIFOs, and as such, the FIFOs are only used for clock domain crossing. As a result FTX 112, GTX113 and GTX114 have all been configured with the minimum selectable depth of 16 words, which preserves BRAM for the Fragment Data FIFO.

This also means, that the ATCA Interface FPGAs will never assert flow control back to the Data Processing FPGAs. Moreover, the ATCA Interface FPGAs can will only sent diagnostic test patterns, or under normal operation, commas characters.

6.2 Tagged Record Reception Logic

A tagged record or fragment is one that was selected by Data Pipeline FPGAs for assembly and transfer to the processing blade. They are sent unmodified from Data Pipeline FPGAs to ATCA Interface FPGAs, with the exception that an over length is added onto the end for data checking purposes. The Tagged Record Receiver logic is structured into four main components:

1. The Tagged Record Receiver state machine
2. The Fragment Info FIFO
3. The Fragment Data Width Adapter
4. The Fragment Data FIFO

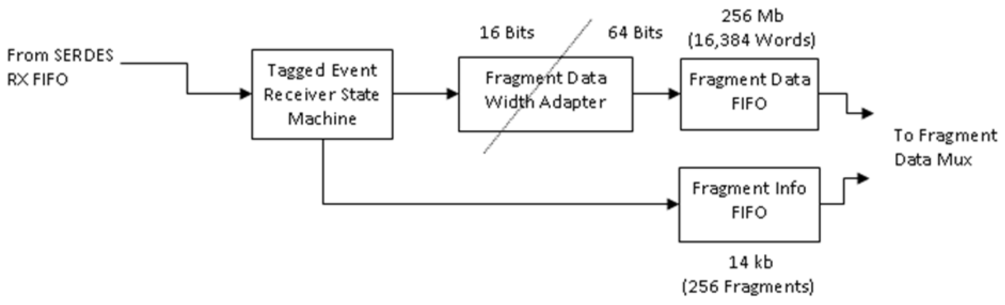


Figure 26 – Tagged Record Receiver block diagram

6.2.1 Tagged Record Receiver State Machine

. The Tagged Record Receiver state machine continuously reads the GTX RX FIFO. Prior to processing tagged record fragments the state machine must synchronize its operation with the incoming data stream. This is done in precisely the same way as described in Section 5.3.1. After locking on to the data pattern, record fragments are received and processed in to the Fragment Info and Data FIFOs. The Tagged Record Receiver state machine extracts the L1 ID and tracks the total record fragment length, which are both written into the Fragment Info FIFO. The fragment length is also verified against the fragment length sent from the pipeline FPGAs. Any errors encountered during the reception of the fragment are also written to the Fragment Info FIFO.

Record fragments are initially stored in the Fragment Data FIFO. Each channel’s Fragment Data FIFO is configured with a programmable Full status flag that is configured to assert when less than 4,100 free words remains. This is more than the defined maximum truncated record size of 4096 words. If the FIFO’s Full flag is asserted at the start of a new fragment, it will be dropped even if its length is less than the space available. This occurs as the total length of the fragment cannot be predetermined by the header information, which prevents partial or corrupted fragments from propagating on to the Fragment Assembly logic. Once a complete record has been received, this machine notifies the Fragment Assembly logic via a double handshake of the “Event” FIFO controller’s “Event Available” flag and the Fragment FIFO status flags.

It also can be configured to operate with either full or truncated fragment headers (must be chosen at compile). This is configured by the TRUNCATED_HEADER generic option. If it is enabled, the machine will expect that the Data Pipeline FPGAs have striped out the first four word of the header data. The machine will instead expect that the fragment start with the first word of the run number; word

"RH05" as specified in Table 7. If the TRUNCATED_HEADER option is not enabled, the logic provides a second generic option to strip the first four header words as they are received by the Tagged Record Receiver to produce the same effect. This is controlled by the KEEP_FULL_HEADER option.

When a link is disabled from readout the Tagged Record Receiver will continue to read the GTX RX FIFO, however, no processing of the data occurs.

6.2.2 Fragment Info FIFO

The Fragment Description FIFO holds the fragment information required for assembly. For each fragment, the Tagged Record Receiver writes the following three items into the Fragment Info FIFO:

1. The 32-bit L1 ID
2. The total fragment length as received in bytes. (The lowest bit is always zero as the GTX data width is 16-bits.)
3. A 7-bit receive error status.

The 7-bit error status provides the following indicators (all bits are active high):

Bit	Name	Meaning
0	ANY_ERROR	Set when any error is detected.
1	NOT_IN_SYNC_ERROR	Set while not in sync. This should never be seen in a fragment passed to the assembly logic.
2	HEADER_ERROR	For non-truncated header mode only. Set if a word other than 0xB0D0 is received while waiting for the next fragment. Causes loss of sync condition. This should never be seen in a fragment passed to the assembly logic.
3	TRACK_ERROR	Set if an error is detected within the track data region. This will result in the forced termination (truncation) of the record.
4	TRAILER_ERROR	Set when a record is truncated. If no other error bits are set (other than ANY_ERROR), then the truncations error was the result of a malformed fragment trailer.
5	TRUNCATION_ERROR	Despite its name, it is only set if the truncation is the result of a fragment exceeding the maximum track count. (max. track count = 0x90)
6	LENGTH_ERROR	Received length does not match length reported by the Data Pipeline FPGA.

Table 19 – Status values from the Tagged Record Receiver.

6.2.3 Fragment Data Width Adapter

The width of the fragment data bus is changed from 16-bits to 64-bit prior to writing it to the Fragment Data FIFO. This prevents throughput bottlenecks in the Fragment Assembly logic, into which all fragment data must be funneled. 64-bits is also the required width for interfacing to both the DDR Interface and 10G Ethernet Interface IP Cores via various pieces of Xilinx AXI bus IP.

It is not assumed that the fragment length will be an integer multiple of 64 bits. The data width adapter is designed to start every new fragment with 64-bit alignment. For every four 16-bit words written into the adapter, one 64-bit word is written to the Fragment Data FIFO. However, the last word is always written to the Fragment Data FIFO as a fully padded 64-bit when the Tagged Record Receiver signals the end of a fragment. The total fragment length information not including any alignment padding is preserved in the Fragment Info FIFO.

6.2.4 Fragment Data FIFO

Each Tagged Record Receiver implements a reception buffer of 32,772kB. The FIFO's data interface is 64-bit wide. This also serves as a domain crossing FIFO at the final stage of throughput expansion. While data is written at 200 Mhz, it is read out in at 333MHz, the processing clock used for fragment assembly. This frequency allows for fullest utilization of the DDR bandwidth after accounting for the required processing delays. There is also a large FIFO between the Fragment Assembly Logic and the DDR for additional burst handling capability. The net fragment assembly throughput approaches 21Gbps for large fragments, but falls off to just below 16Gbps for zero track fragments.

6.3 Fragment Assembly

The Fragment Assembly Logic consists of four main components:

1. The Fragment Data Multiplexer
2. The Fragment Assembly State Machine
3. The Assembly Table
4. The Assembly Table Management State Machine

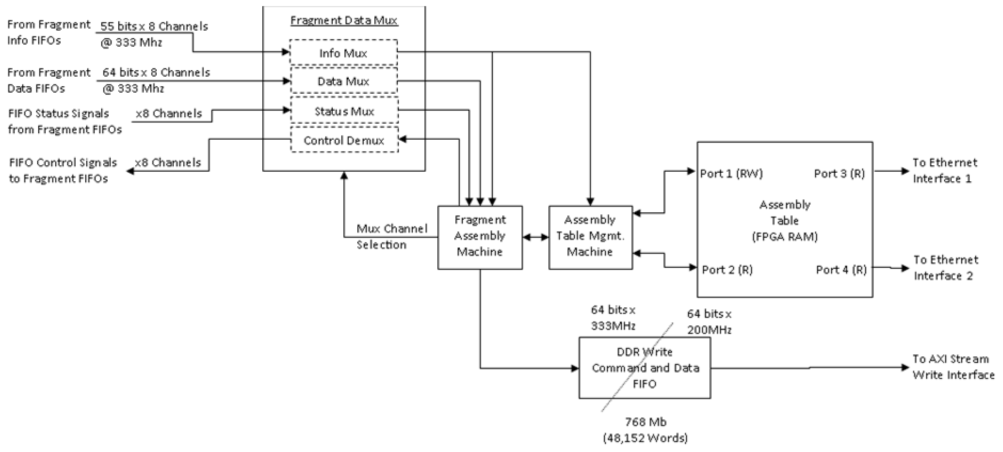


Figure 27 – Fragment Assembly Logic block diagram

The fragment assembly is managed by the two state machines. The Fragment Assembly State Machine is responsible for the following:

- Controls the Fragment Data Multiplexer
- Handles the fragment data and controls the read side of the Fragment FIFOs.
- Delegates Assembly Table actions to the Assembly Table Management State Machine.

- Generates DDR write commands that, along with the fragment data are passed to the AXI-Stream Write Interface.

The Assembly Table Management State Machine handles all direct interactions with the Assembly Table. This includes:

- Allocating assembly regions.
- Searching the current set of assembly regions for an L1 ID.
- Calculating the DDR write address when appending a fragment to an existing region.
- Storing fragment information required to generate the Assembly Header.
- Signaling the Assembly Read Machine to transfer an assembly once this machine determines that it is complete.
- Forcing the transmission of Assemblies if one or more channels do not report a fragment for a given L1ID.
- Monitor the status of ongoing DDR transfers.

6.3.1 Fragment Data Multiplexer

The Fragment Data Multiplexer is the interface between the assembly logic and the eight Tagged Record Receivers. The Multiplexer design consists of three multiplexers and one a demultiplexer or “decoder”.

Multiplexers:

1. 8x64-bit Fragment Data FIFO bus.
2. 8x55-bit Fragment Info FIFO bus.
3. 8x10-bit Fragment FIFO status bus.

Demultiplexers/Decoder:

1. 3x1:8 one-hot decoders for driving Fragment FIFO control signals.

All four components share a common selection control. Due to the large number of signals that fan-in and fan-out between the Fragment Assembly Logic and the Fragment FIFOs and the speed of the transfers (running at 333MHz), the design has a one clock latency for both input and outputs, except for the Fragment Data FIFO bus mux, which has a 2 clock latency. The selection control has a one clock latency for all muxes/demuxes.

Due the pipeline delays careful consideration must be made to latencies in relation to the operation of the Fragment Assembly State Machine. For example, asserting the FIFO read takes one clock to be received by the FIFO, and two additional clocks to return new data. Also, when switching channel the FIFO status signals take one clock before they are valid.

These latencies are preferred over running at a slow processing speed. As the typically the most time is spent either transferring Fragments to the AXI-Stream Write Interface’s Input FIFO and running Assembly Table Operations, both of which greatly benefit from high clock speed, such that the throughput improvements far outweighs the few clocks of Fragment FIFO switching and control latencies. This is further mitigated by the parallel operation of the Fragment Assembly State Machine and Assembly Table Management State Machine.

6.3.2 Fragment Assembly State Machine

The purpose of the Fragment Assembly State Machine is to control the transfer of fragment data from the Fragment Data FIFOs (via the Fragment Data Multiplexer) into the appropriate assembly regions within the DDR address space.

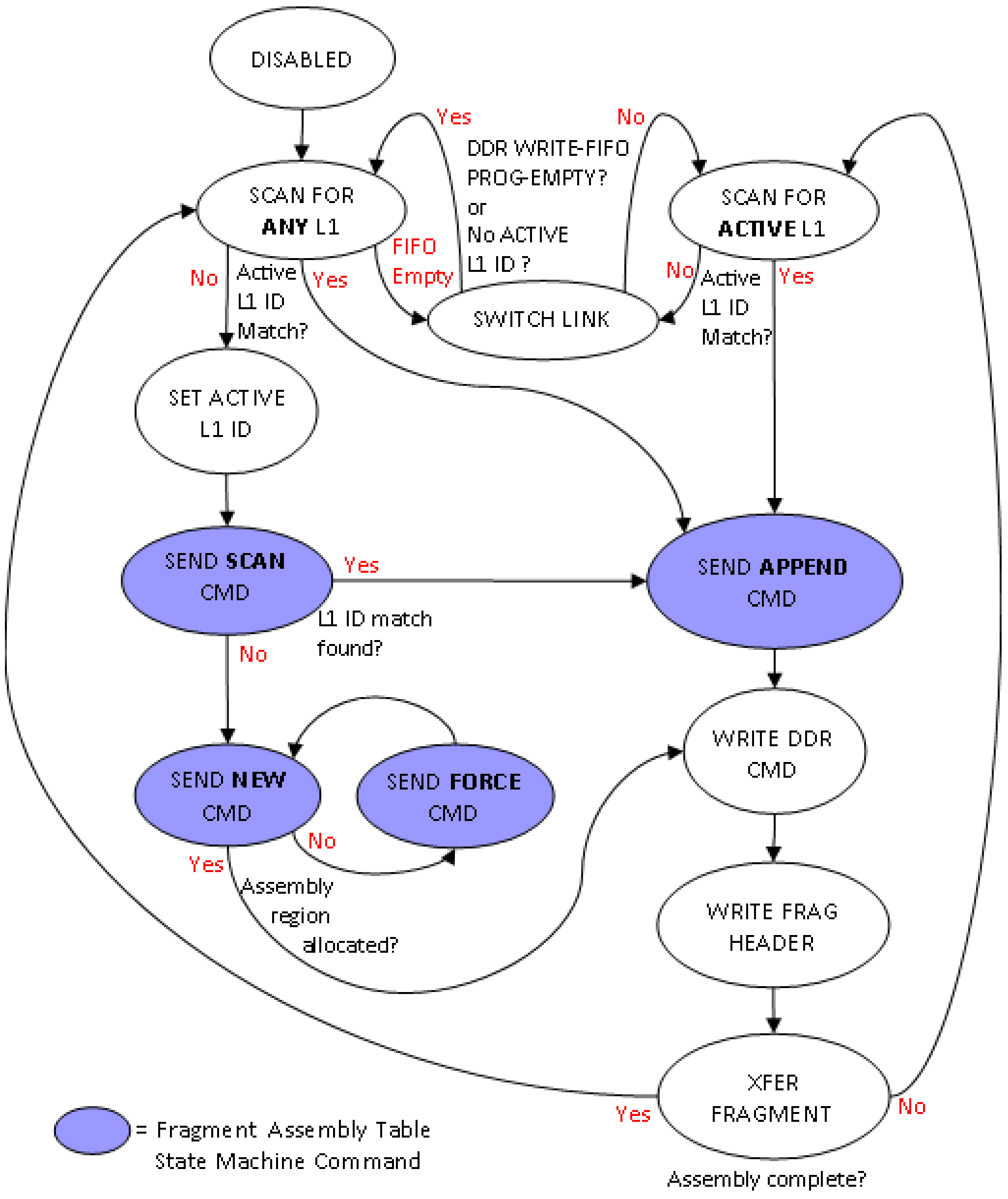


Figure 28 – Simplified Fragment Assembly State Machine Diagram

The three scan and switch states define the idle loop. The "Switch Link" state increments the selection control to the Fragment Data Multiplexer. The state machine continuously and sequentially scans the Fragment FIFO Fragment status for available data. For either of the two “SCAN” states, when the Fragment Assembly State Machine finds an input channel with a complete fragment, it will check the extracted L1 ID in the Fragment Info FIFO against the “Active L1 ID,” if the Active L1 ID is valid.

The “Active L1 ID” is defined as the last L1 ID that was processed (or is being processed). It is defined to be active so long as it’s assembly has not been completed. If the L1 ID matches, it proceed to the “SEND APPEND CMD” state, otherwise it depends on the SCAN state. In the “SCAN FOR ANY L1 ID” state it proceeds to back the current L1 ID the “Active L1 ID”; in the other scan state the data is ignored and the next link is checked. So long as the AXI-Stream Write Interface’s Input FIFO is not close to running empty, the Fragment Assembly State Machine will continue to hunt for the “Active L1 ID” as an APPEND operations are far faster than a SEARCH and NEW, or SEARCH and APPEND operation. Otherwise, it reverts to looking for any record fragment, as it is more important to maximize the DDR bandwidth than to minimize the number of Fragment Data Table operations. Also, the Fragment Assembly State Machine immediately begins looking for any available fragment if the last assembly was marked as complete by the Assembly Table Management State Machine.

In either scan scenario, once a fragment has been identified for assembly by the Fragment Assembly State Machine, it will initiate a series of interactions with Assembly Table by sending commands the Assembly Table Management State Machine. After issuing commands, the Fragment Assembly State Machine will wait until either the Assembly Table Management State Machine reports “SUCCESS” or “COMPLETE”. The machine signals SUCCESS as soon as it has been determined that the commands will be successfully completed and all output to the Fragment Assembly State Machine are valid.

In the case of the APPEND commands, this allows the Fragment Assembly State Machine to nearly immediately proceed with generating the DDR write command building the fragment header, and start the transfer of fragment data, while the Assembly Table Management State Machine updates address pointer, total assembly length and writing the updating the other elements in the Assembly Table.

Every SUCCESSful command will eventually also be reported as COMPLETEd. However, Every COMPLETEd will not be SUCCESSful, and this routinely occurs as part of normal operation. In the case of the “SEND SCAN CMD” state, the Fragment Assembly State Machine waits for COMPLETE. As in this state it will be immediately sending either an APPEND or NEW command, it must wait for the COMPLETE status as the Assembly Table Management State Machine can only process one command at a time. A SUCCESSful SCAN signals that an existing assembly region was found for the L1 ID and may by APPENDED, otherwise a NEW region must be allocated.

As long as the Scan/New is successful, it proceeds to transfer the fragment to the DDR after writing the Fragment Header. However if the New operation is unsuccessful, it means that all available assembly regions are occupied with incomplete fragment assemblies. One must now force the completion of the oldest region to make space for the new L1 ID. This occurs in the “SEND FORCE CMD” state in the diagram above. It continues to cycle between FORCE and NEW until a successful NEW operation is reported. To maximize bandwidth once the Fragment Assembly Machine begins a transfer it will continue uninterrupted until the end of the fragment. This is achieved by verifying that sufficient space is available in the DDR Write FIFO prior to starting transfer, similar to the way the Fragment Receiver State Machine verifies available space in the Fragment Data FIFO prior to transfer.

The write command and write data share command FIFO. First the write command is written followed by the data. The format of the data written to the write FIFO is shown in the table below.

	Bit Field							
Word	63..56	55..48	47..32	31..16	15..0			
Write Command	Reserved	WRITE ACK ID	WRITE LENGTH	WRITE ADDRESS				
Write Data	Fragment Data Word 3		Fragment Data Word 2	Fragment Data Word 1	10111110	USER TAG	E V	C P
Write Data	Fragment Data Word 7		Fragment Data Word 6	Fragment Data Word 5	Fragment Data Word 4			
Write Data	…		…	…	…			

Figure 29 –

6.3.3 Assembly Table

The Assembly Table holds all the assembly status and assembly header information. The table is constructed as a RAM with 1 read-write port and 3 read-only ports. The RAM is segmented in to separately writable sections blocks. While all the read and write pointer are tied in common, for any given port, the write enables are unique for each elements of the table. The table contains 16 addressable elements, one for each fragment assembly region. The Fragment Data Table stores the following information regarding the fragment assemblies:

1. CHANNEL_MASK – 8 bits – Included in the assembly header. Indicates which SSB links have reported matching L1 IDs. The corresponding bit of the mask is set once a fragment has been stored within its designated assembly region of the DDR memory.
2. DDR_ADDR – 28 bit field – Value stores the next writable address with the corresponding assembly region.
3. FRAGMENT_CH_ID – 8 x 3 bit array of values – Included in the assembly header. This array is treated as a list of 8 elements that store the channel order of the fragments stores at the corresponding assembly region. Fragments are stored as they arrive, channels are not ordered within the assembly regions.
4. FREE – 1 bit – This bit is set when the assembly region is clear for reuse.
5. L1_ID – 32 bit – Included in the assembly header. This field stores the L1_ID. Once a region has been designated for a particular L1_ID, it cannot be changed until the assembly has been sent by either completing the assembly, or by forcing it's completion due to a missing or late fragment.
6. LOCKED - 1 bit – This bit is set to lock access to the assembly region once it has been handed off to the Ethernet side of the design. This prevents an assembly that has already been forced for readout from being later appended should the fragment show up late. This also prevents more than 8 fragments from being written to a single region in the case that an L1_ID is duplicated. Once sent this bit remains set until a "garbage" collection cycle of the assembly table sees that the assembly has been sent to the blade and control has been released by the Ethernet logic.

7. NUM_FRAGMENTS - 4 bits – Included in the assembly header. Value stores the number of fragments that have been written to the corresponding assembly region.

In addition to the elements of the Fragment Assembly table implemented as RAM, there are three special elements that are implemented in the fabric logic: two S-R Flip Flops and one 3-bit counter, for each assembly region. These are not implemented as RAM due to the need to have shared write control. These elements are:

8. MARKED_TO_SEND – S-R Flip Flop –This FF is cleared when the Assembly Table Management Machine frees a region for reuse. It is set when the DDR write logic, confirms that the last fragment has been written to the DDR. Will also be set if a region is forced to readout.
9. SENT - S-R Flip Flop – This FF is cleared when the Assemble Table Management Machine frees a region for reuse. It is set when the Ethernet logic, has completed reading the data from the region, and is returning control to the Assembly Table Management Machine.
10. UNACKED_WRITES – 3-bit Counter – This counter tracks the number of Fragments that have been allocated for the region but not yet fully transferred into DDR. Since the DDR is slower than the Assembly logic, one or more fragments may be temporarily chased in the DDR write interface FIFO and the various buffering elements of the AXI components. The sole purposes of these counters are to prevent the Assemble Table Management Machine from forcing an assembly for readout if one or more fragments have not been completely transferred into the region. The counter is incremented by the Assemble Table Management Machine each time the fragment is allocated and decremented each time a write ack is received from the DDR interface.

6.3.4 Fragment Data Table Management State Machine

The Fragment Data Table Management State Machine (FDTMSM) handles all interactions with the Fragment Data Table. It also calculates DDR memory address offsets for placement of new fragments into the assembly regions. As each fragment is copied to a given assembly region, the dual-port RAM location associated with that region is updated with the total length of data for the assembly region in use and which SERDES links have had their data copied into that assembly region,

The interface to the FDTMSM is via 5 possible commands: Append, Clean, Force, New and Scan. The diagram shows the general flow of how these commands are executed.

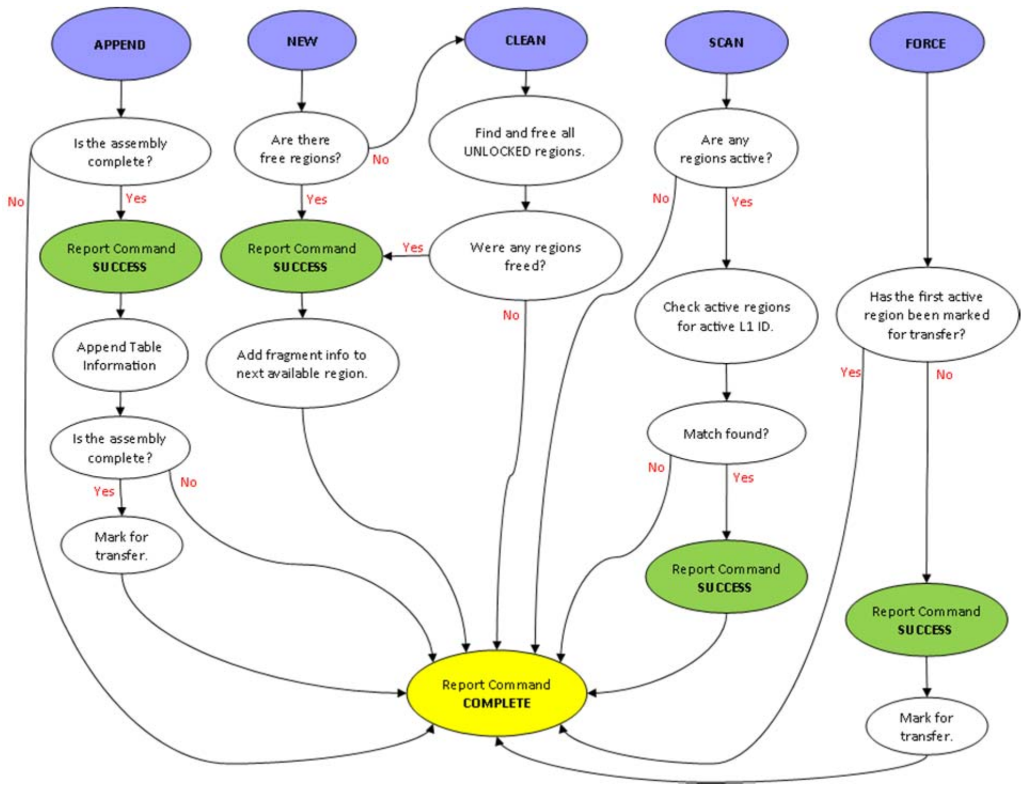

APPEND
NEW
CLEAN
SCAN
FORCE
Is the assembly complete?
Are there free regions?
Find and free all UNLOCKED regions.
Are any regions active?
No
Yes
Report Command SUCCESS
Were any regions freed?
Append Table Information
Add fragment info to next available region.
Check active regions for active L1 ID.
Has the first active region been marked for transfer?
Is the assembly complete?
Match found?
Mark for transfer.
Report Command COMPLETE
Mark for transfer.

Figure 30 –

For each operation the FDTMSM responds with an operation complete and operation success status. If success signal is set the operation completed normally, otherwise the operation was unsuccessful. The definition of success is dependent on the operation being performed. The operations are:

1. New – Attempt to allocate a new assembly region with the current L1 ID. If the operation is successful it will automatically record the first fragments information into the table.
   This will fail if no assembly regions are available. If the table is full, the New operation will automatically invoke a clean operation to free any assembles regions that have been completely processed, including transmission to the blade.
2. Scan – Search for an incomplete assembly matching the current L1 ID. This will only return the success status if an existing assembly region with less than 8 fragments is found. A successful Scan does not update the table, but it does change to the current working region for future (append) operations.
3. Append – Appends the current fragment to currently selected assembly. This will always be successful as long as the assembly has not been marked as complete. After appending the fragment to the assembly, it checks whether the assembly is complete by comparing the enabled channels to the channels that have been recorded in this region.
4. Force – Forces the oldest assembly to be marked for readout. This operation is requested by the Fragment Data Assembly State Machine in the event that a New operation fails. This

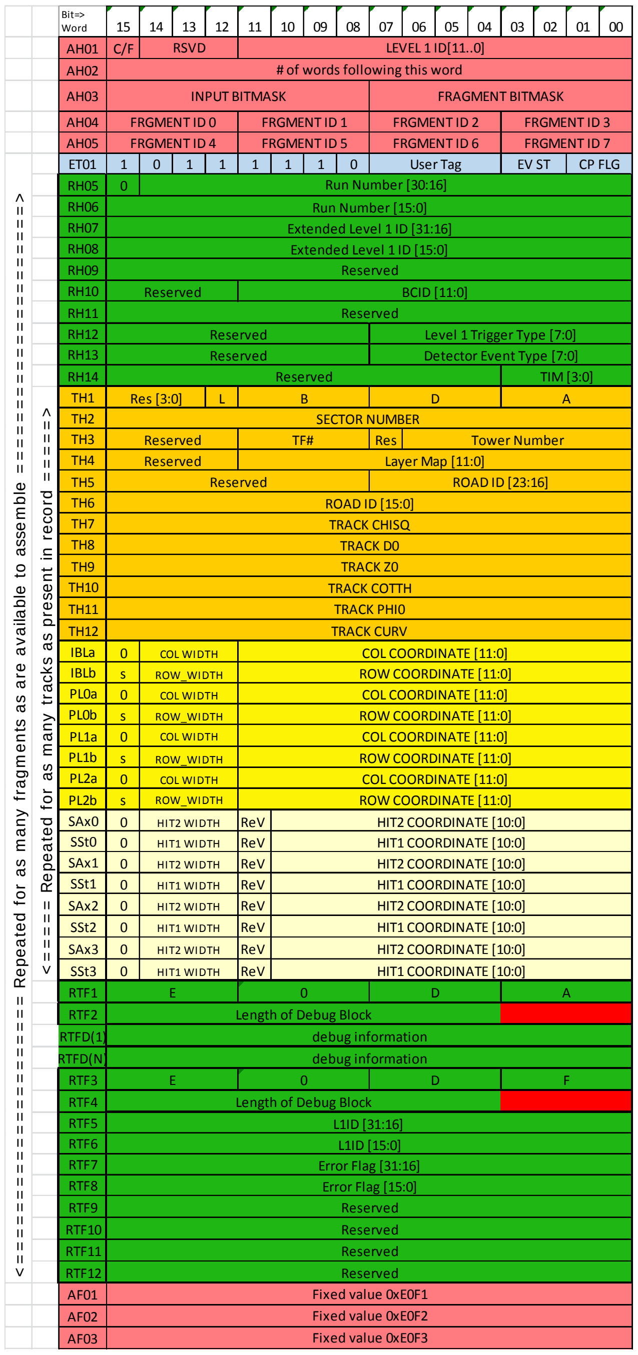
forces incomplete fragments to read out in the event that one or more channels never report a fragment.

5. Clean – Performed automatically when a New operation is executed and no free regions are available. Due to pipelining, it is more efficient to free assembly regions in bursts, when space is required rather than one at a time when assemblies are sent. The Clean operation is never commanded by the Fragment Data Assembly State Machine.

Once an assembly has been marked complete, due to either all eight fragment slots being used or the all expected fragments being processed, the Data Packet Generator will automatically transfer the data to the Ethernet Interfaces. Assemblies are sent as they are completed, not in order of L1 ID. Once sent, the Data Packet Generator will signal the FDTMSM, which then marks the region as sent so that they may be freed during a Clean operation. This may occur even, while other operations are in progress.

6.4 Format of the data assembly

The purpose of the Fragment Data Assembly State Machine (FDASM) is to control the transfer of fragment data from the Fragment Data FIFOs (via the Fragment Data Multiplexer) into the appropriate assembly regions within the DDR address space.



Bit=> Word	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
AH01	C/F	RSVD			LEVEL 1 ID[11..0]											
AH02	# of words following this word															
AH03	INPUT BITMASK								FRAGMENT BITMASK							
AH04	FRGMENT ID 0				FRGMENT ID 1				FRGMENT ID 2				FRGMENT ID 3			
AH05	FRGMENT ID 4				FRGMENT ID 5				FRGMENT ID 6				FRGMENT ID 7			
ET01	1	0	1	1	1	1	1	0	User Tag				EV ST		CP FLG	
RH05	0	Run Number [30:16]														
RH06	Run Number [15:0]															
RH07	Extended Level 1 ID [31:16]															
RH08	Extended Level 1 ID [15:0]															
RH09	Reserved															
RH10	Reserved				BCID [11:0]											
RH11	Reserved															
RH12	Reserved								Level 1 Trigger Type [7:0]							
RH13	Reserved								Detector Event Type [7:0]							
RH14	Reserved												TIM [3:0]			
TH1	Res [3:0]			L	B				D				A			
TH2	SECTOR NUMBER															
TH3	Reserved				TF#				Res	Tower Number						
TH4	Reserved				Layer Map [11:0]											
TH5	Reserved								ROAD ID [23:16]							
TH6	ROAD ID [15:0]															
TH7	TRACK CHISQ															
TH8	TRACK D0															
TH9	TRACK Z0															
TH10	TRACK COTTH															
TH11	TRACK PHI0															
TH12	TRACK CURV															
IBLa	0	COL WIDTH			COL COORDINATE [11:0]											
IBLb	s	ROW_WIDTH			ROW COORDINATE [11:0]											
PL0a	0	COL WIDTH			COL COORDINATE [11:0]											
PL0b	s	ROW_WIDTH			ROW COORDINATE [11:0]											
PL1a	0	COL WIDTH			COL COORDINATE [11:0]											
PL1b	s	ROW_WIDTH			ROW COORDINATE [11:0]											
PL2a	0	COL WIDTH			COL COORDINATE [11:0]											
PL2b	s	ROW_WIDTH			ROW COORDINATE [11:0]											
SAx0	0	HIT2 WIDTH			ReV	HIT2 COORDINATE [10:0]										
SSt0	0	HIT1 WIDTH			ReV	HIT1 COORDINATE [10:0]										
SAx1	0	HIT2 WIDTH			ReV	HIT2 COORDINATE [10:0]										
SSt1	0	HIT1 WIDTH			ReV	HIT1 COORDINATE [10:0]										
SAx2	0	HIT2 WIDTH			ReV	HIT2 COORDINATE [10:0]										
SSt2	0	HIT1 WIDTH			ReV	HIT1 COORDINATE [10:0]										
SAx3	0	HIT2 WIDTH			ReV	HIT2 COORDINATE [10:0]										
SSt3	0	HIT1 WIDTH			ReV	HIT1 COORDINATE [10:0]										
RTF1	E				0				D				A			
RTF2	Length of Debug Block												🟥			
RTFD(1)	debug information															
RTFD(N)	debug information															
RTF3	E				0				D				F			
RTF4	Length of Debug Block												🟥			
RTF5	L1ID [31:16]															
RTF6	L1ID [15:0]															
RTF7	Error Flag [31:16]															
RTF8	Error Flag [15:0]															
RTF9	Reserved															
RTF10	Reserved															
RTF11	Reserved															
RTF12	Reserved															
AF01	Fixed value 0xE0F1															
AF02	Fixed value 0xE0F2															
AF03	Fixed value 0xE0F3															

<= = = = = = = = = = = = Repeated for as many fragments as are available to assemble = = = = = = = = = = = = = = =>

<= = = = = Repeated for as many tracks as present in record = = = = =>

Figure 31 – Format of data in a completed assembly fragment

The Record Header, Track Header, Track and Record Trailer data of the tagged record is exactly copied from the internal mesh SERDES link without modification. Each of these tagged records is the data received from one pipeline for a selected Level 1 ID value. The red “AH” and “AF” sections are added by the assembly logic. The single word labeled “ET01” is the Event Tagging information that was added by the pipeline FPGA to indicate whether the event fragment has errors, has been marked through user intervention or was copied more than once.

It is required that all data in the assembly region be loaded by the "filler" machine and that the "reader" machine perform absolutely no modification of the data as taken from the assembly region in the process of copying the event to the UDP buffers.

6.4.1 Details of the assembly header

This section provides explanation of the various fields in the Assembly Header.

Word	Bit(s)	Explanation
AH01	15	This bit is the Complete/Forced bit. If set, readout of this event was forced and thus at least one fragment is missing.
AH01	14:12	Reserved bits.
AH01	11:0	Lower 12 bits of the Level 1 ID identifying this event.
AH02	15:0	16-bit count of the number of 16-bit words in this event that follow word AH02, inclusive of all Event Tags, Record Headers, Track Headers, Track Data and Record Trailers in all fragments plus the three Assembly Trailer words.
AH03	15:8	The Input Bitmask field identifies which SERDES links were enabled for inclusion in this assembly region event. Bits 15:12 are associated with pipeline indices 3, 2, 1 & 0 of the "U1" FPGA, respectively. Bits 11:8 are associated with pipeline indices 3, 2, 1 & 0 of the "U2" FPGA, respectively.
AH03	7:0	The Fragment Bitmask field identifies which SERDES links reported data, and were included, in this assembly region event. The combination of the two fields of word AH03 suffice to identify which fragments may be missing, but do not provide sufficient information to know the order in which the fragments were collected.
AH04 and AH05	All	Each four-bit field in words AH04 and AH05 identifies which SERDES link and by association which SSB) provided each fragment in the assembly region in the order in which they were collected. Fragment ID 0 specifies the SERDES link from which the first fragment came; fragment ID 1 specifies the SERDES link from which the second fragment came, etc. The numbering convention shall be • "0000" indicates data from the SERDES associated with pipeline index 0 of the "U1" FPGA • "0001" indicates data from the SERDES associated with pipeline index 1 of the "U1" FPGA • "0010" indicates data from the SERDES associated with pipeline index 2 of the "U1" FPGA • "0011" indicates data from the SERDES associated with pipeline index 3 of the "U1" FPGA • "0100" indicates data from the SERDES associated with pipeline index 0 of the "U2" FPGA • "0101" indicates data from the SERDES associated with pipeline index 1 of the "U2" FPGA • "0110" indicates data from the SERDES associated with pipeline index 2 of the "U2" FPGA • "1111" indicates that the fragment does not exist in the data to follow. Fragments shall be collected in the simplest way possible and the fields of words AH04 and AH05 shall indicate the temporal order in which fragments of the event were found.

Table 20 – Assembly header field definitions

6.5 AXI Based Interconnect

Many firmware components and hardware interfaces are interconnected via Xilinx based AXI IP cores. The green blocks in the diagram below show all Xilinx AXI IP used in the design.

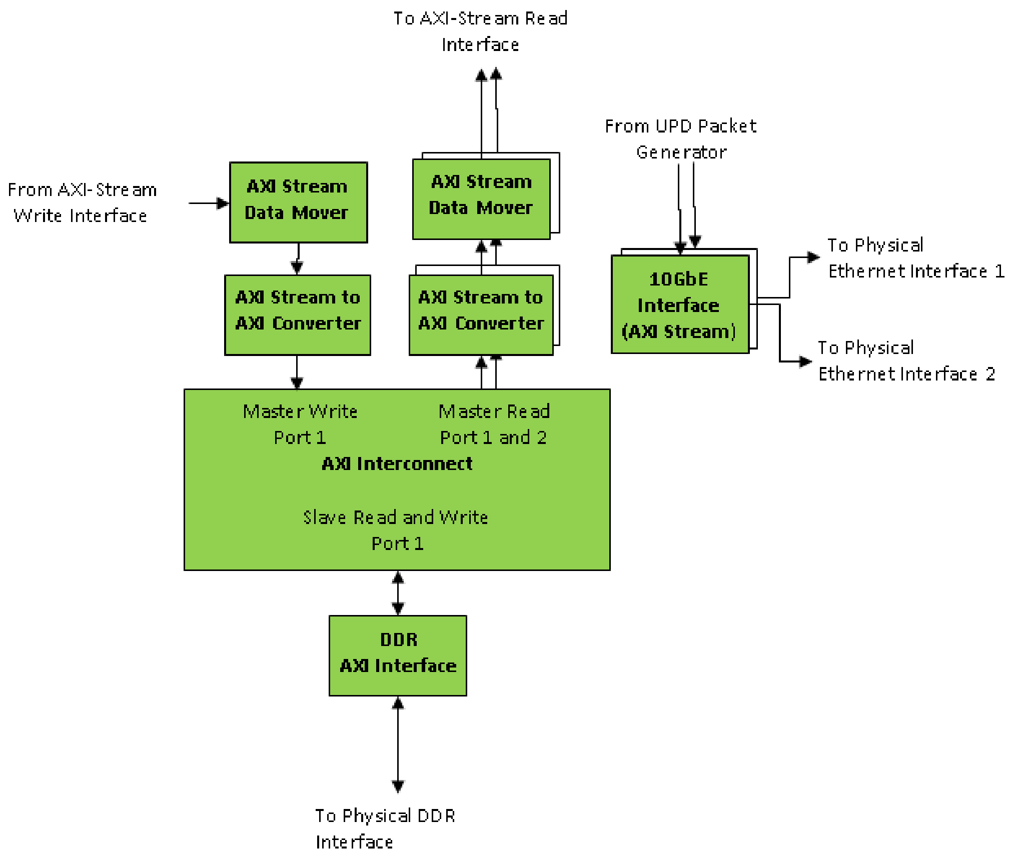

To AXI-Stream Read Interface
From UPD Packet Generator
From AXI-Stream Write Interface
AXI Stream Data Mover
AXI Stream Data Mover
To Physical Ethernet Interface 1
AXI Stream to AXI Converter
AXI Stream to AXI Converter
10GbE Interface (AXI Stream)
To Physical Ethernet Interface 2
Master Write Port 1
Master Read Port 1 and 2
AXI Interconnect
Slave Read and Write Port 1
DDR AXI Interface
To Physical DDR Interface

Figure 32 –

All custom firmware components interface to AXI via the Stream variant of the AXI standard. The AXI Steam interface either communicates directly with the target (as is the case with the 10GbE core) or via AXI Stream to AXI converters in the case where the target does not support AXI stream. All access to the DDR is performed over a Xilinx AXI Crossbar IP. The cross bar implements 2 write channels and 2 read channels. This crossbar interconnect manages arbitration between the read and write masters. Access priority is always given to the read interfaces in avoid delaying a completed assembly from transmission to the blade.

Refer to Xilinx documentation for more detail regarding Xilinx AXI and AXI Stream based IP.

6.6 UDP Packet Generator

Once a fragment assembly has been marked for readout, the UDP packet generation logic reads the assembled fragments from the DDR and form it into UDP packets. UDP Fragmentation protocol is used send assemblies that are greater than 1500 bytes. Packets are transferred to the XAUI core. Once all the assembly data has been transferred, it is marked for cleanup by the FDTMSM.

Two copies of a UDP Packet Generator process monitor each of the two Event FIFOs that are filled by the Reader machine. Each UDP Packet Generator process, upon being signal by the fragment reader that data is available, pulls data out in 1480 byte blocks, the default value of the packet size register. This is consistent with a maximum transmission unit (MTU) size that, when padded with the standard 8-byte UDP header plus 20-byte IP header, will form packets 1500 bytes long. The firmware does not support jumbo packets. The packet size register may not be set to a size that would qualify as a jumbo packet. No checksum is calculated. Software within the blade shall make no assumptions regarding packet size as the size of assembled fragments is indeterminate. The firmware shall make no attempt to pad or otherwise align the total data size with some integer number of packets.

7 Clock Generator circuits

The FLIC GTX transceivers may run at a variety of line rates as set by the external clock generators and PLL settings. The PLL settings are defined in the Virtex-6 GTX User's guide. The PLL design is shown in that guide as Figure 2.9, copied here for reference:

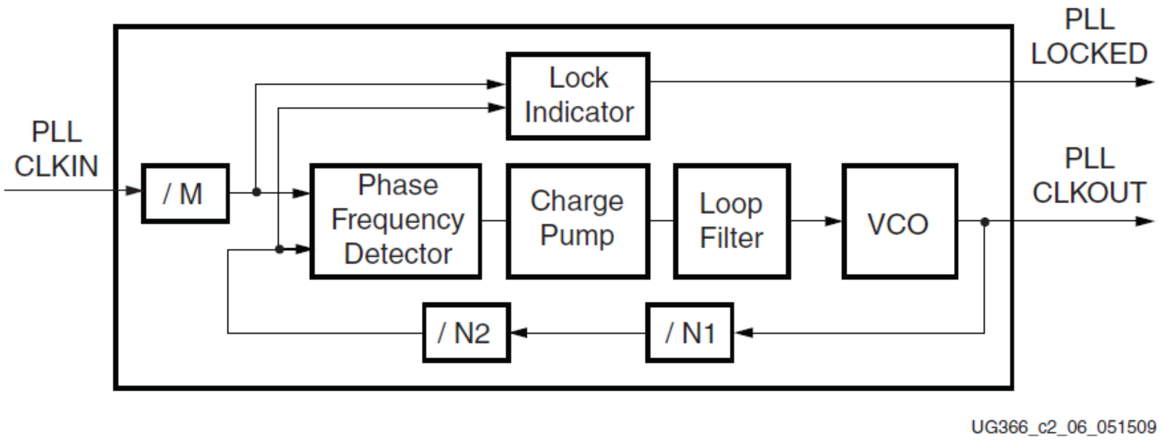


Figure 33

The input clock is derived from the external clock generator chips. Frequency control parameters M, N1 and N2 are defined in the constraints file at compile time. In the FLIC, factor M may be set to 1 or 2. To date, the constraint files of the FLIC firmware always set this factor to 1. Similarly, factor N1 may be set to 4 or 5. Factor N2, according to the data sheet, may be set to a value of 2, 4 or 5 subject to the limitations of the VCO frequency limits. The electrical specfications of the Virtex-6 define that the minimum VCO frequency is 1.2GHz and the maximum is 2.7GHz. Similarly, the input clock is restricted to the range 62.5MHz – 650MHz.

The reference clock supplied to the pipeline FPGAs of the FLIC for serial communication is developed by a CDCM61004 programmable clock generator (U48). A high stability 25.0000MHz crystal is used as the base input to the CDCM61004. Three OD (Output Divide) and two PR (Prescale/Feedback) control pins allow a variety of frequencies to be synthesized from the 25MHz input, from 46.875MHz to 625MHz. The external clock generator frequency is modified by the factors N1, N2 and M of the PLL within the FPGA using the equation

$$Fpll = Fclkin * \frac{N1 * N2}{M}$$

to derive the PLL frequency used for serial transmission. Table 21 – Summary of all possible GTX PLL frequency settings in the FLIC summarizes all the possible combinations of F_{pll} that can be obtained.

TXPLL_DIVSEL_FB	2	2	4	4	5	5
TXPLL_DIVSEL45_FB	4	5	4	5	4	5
Refclk in, MHZ						
46.875						
62.5	500.00	625.00	1000.00	1250.00	1250.00	1562.50
62.5	500.00	625.00	1000.00	1250.00	1250.00	1562.50
75	600.00	750.00	1200.00	1500.00	1500.00	1875.00
78.125	625.00	781.25	1250.00	1562.50	1562.50	1953.13
83.333	666.66	833.33	1333.33	1666.66	1666.66	2083.33
93.75	750.00	937.50	1500.00	1875.00	1875.00	2343.75
100	800.00	1000.00	1600.00	2000.00	2000.00	2500.00
104.167	833.34	1041.67	1666.67	2083.34	2083.34	2604.18
125	1000.00	1250.00	2000.00	2500.00	2500.00	3125.00
125	1000.00	1250.00	2000.00	2500.00	2500.00	3125.00
150	1200.00	1500.00	2400.00	3000.00	3000.00	3750.00
156.25	1250.00	1562.50	2500.00	3125.00	3125.00	3906.25
166.667	1333.34	1666.67	2666.67	3333.34	3333.34	4166.68
187.5	1500.00	1875.00	3000.00	3750.00	3750.00	4687.50
200	1600.00	2000.00	3200.00	4000.00	4000.00	5000.00
208.333	1666.66	2083.33	3333.33	4166.66	4166.66	5208.33
250	2000.00	2500.00	4000.00	5000.00	5000.00	6250.00
312.5	2500.00	3125.00	5000.00	6250.00	6250.00	7812.50
375	3000.00	3750.00	6000.00	7500.00	7500.00	9375.00
500	4000.00	5000.00	8000.00	10000.00	10000.00	12500.00
600	4800.00	6000.00	9600.00	12000.00	12000.00	15000.00
625	5000.00	6250.00	10000.00	12500.00	12500.00	15625.00

Table 21 – Summary of all possible GTX PLL frequency settings in the FLIC

The cells shaded yellow are unreachable because the input reference frequency is too low. The orange shaded cells indicate unusable settings that violate the allowed PLL frequency range. The serial line rate that can be achieved for the different allowable PLL frequencies is controlled by a final division factor that can be set in the compilation constraints file to a value of 1, 2 or 4. The result of the division factor setting is given by

$$Fline = \frac{Fpll * 2}{D}$$

Thus the minimum serial line rate that can be generated by the FLIC is (1200 * 0.5), or 600Mb/sec, and the maximum serial rate is (2666.67 * 2), or 5.33Gb/sec. Common values of 1.0, 2.0, 2.5, 3.0, 3.125, 4.0 & 5.0 Gb/sec are all achievable.

Appendix A: Acronyms and Abbreviations

FLIC	FTK to Level-2 Interface Card
ATCA	Advanced Telecommunications Computing Architecture
FTK	Fast TracKer
SSB	Second Stage Board
RAM	Random Access Memory
ROS	ReadOut Subsystem
ATLAS	A Toroidal LHC Apparatus
SFP	Small Form-Factor Pluggable
Gb	Gigabit
DDR	Double Data Rate
FPGA	Field-Programmable Gate Array
10GbE	10 Gigabit Ethernet
DIMM	Dual In-Line Memory Module
IPMC	Intelligent Platform Management Controller
LAPP	Laboratoire d'Annecy-le-Vieux de Physique des Particules
LED	Light Emitting Diode
RTM	Rear Transition Module
JTAG	Joint Test Action Group
SRAM	Static Random Access Memory
ID	Identification
ADC	Analog to Digital Converter
I2C	Inter-Integrated Circuit