Link and Firmware Specification

Spy Buffer Link (inter-FPGA and FLIC-to-Blade)

-- PRELIMINARY --

July 9th, 2015
Version 1.2

Originator: J. Anderson / J. Love / M. Oberling

1. GENERAL INFORMATION

This document describes the Spy Buffer link used in the ATLAS FTK system, specifically the FLIC board and the crate in which the FLIC resides. The FLIC receives data from up to 8 Second Stage Boards (SSBs) over fiber optic cables. This information is presented as "records", where each record contains at minimum a Record Header and a Record Trailer, as shown in Figure 1. The words tagged RHxx are the Record Header and the words tagged RTFxx are the Record Trailer. The colors used in Figure 1 and Figure 2 are provided for visual separation only. Data coming from the SSB modules is 16 bits wide. A record may contain any number of tracks from zero to a maximum of 145, at which point the receiving state machine truncates the record. Data reception logic that is not defined within this document performs all necessary housekeeping functions such as comma detection, link alignment and such so that the Spy Buffer link need only concern itself with packed data that contains no interruptions.

Bit=> Word	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
RH01	Region ID			0	1	1	0	1	0	0	1	1	0	1	0	0	<======== RECORD HEADER =========
RH02	Reserved																
RH03	Run Number - Bytes 3-2																
RH04	Run Number - Bytes 1-0																
RH05	Extended Level 1 ID - Bytes 3-2																
RH06	Extended Level 1 ID - Bytes 1-0																
RH07	Bunch Crossing ID - Bytes 3-2																
RH08	Bunch Crossing ID - Bytes 1-0																
RH09	Level 1 Trigger Type - Byte 3-2																
RH10	Level 1 Trigger Type - Byte 1-0																
RH11	Detector Event Type - Bytes 3-2																
RH12	Detector Event Type - Bytes 1-0																
RH13	NUM_TRACK_HI_PT_THRSH - Bytes 1-0																
RH14	NUM_TRACK_LO_PT_THRSH - Bytes 1-0																
RTF1	Fixed Value 0x0F0F																<= RECORD TRAILER =>
RTF2	Fixed Value 0xA5A5																
RTF3	Fixed Value 0xF0F0																
RTF4	Fixed Value 0x5A5A																
RTF5	Core Crate Status bits (format TBD)																
RTF6	Core Crate Status/Error bIts (format TBD)																

Figure 1 – Minimal input format for SSB data.

The vast majority of records, however, are expected to contain "tracks", additional blocks of data that are interposed between the Record Header and the Record Trailer. The format of a record is shown in Figure 2.

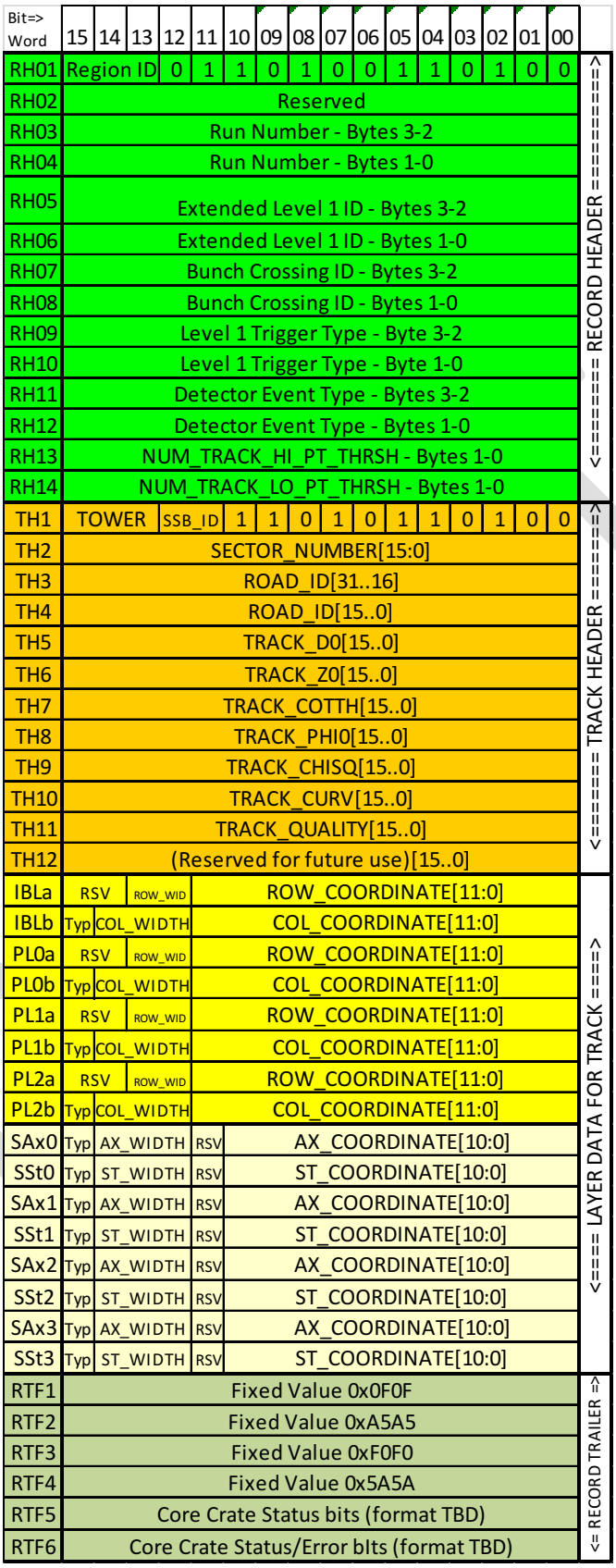

Figure 2 – Structure of a record with one track.

1.1 Architecture of FLIC associated with the Spy Buffer

The FLIC implements four main FPGAs connected via internal SERDES links as shown in Figure 3.

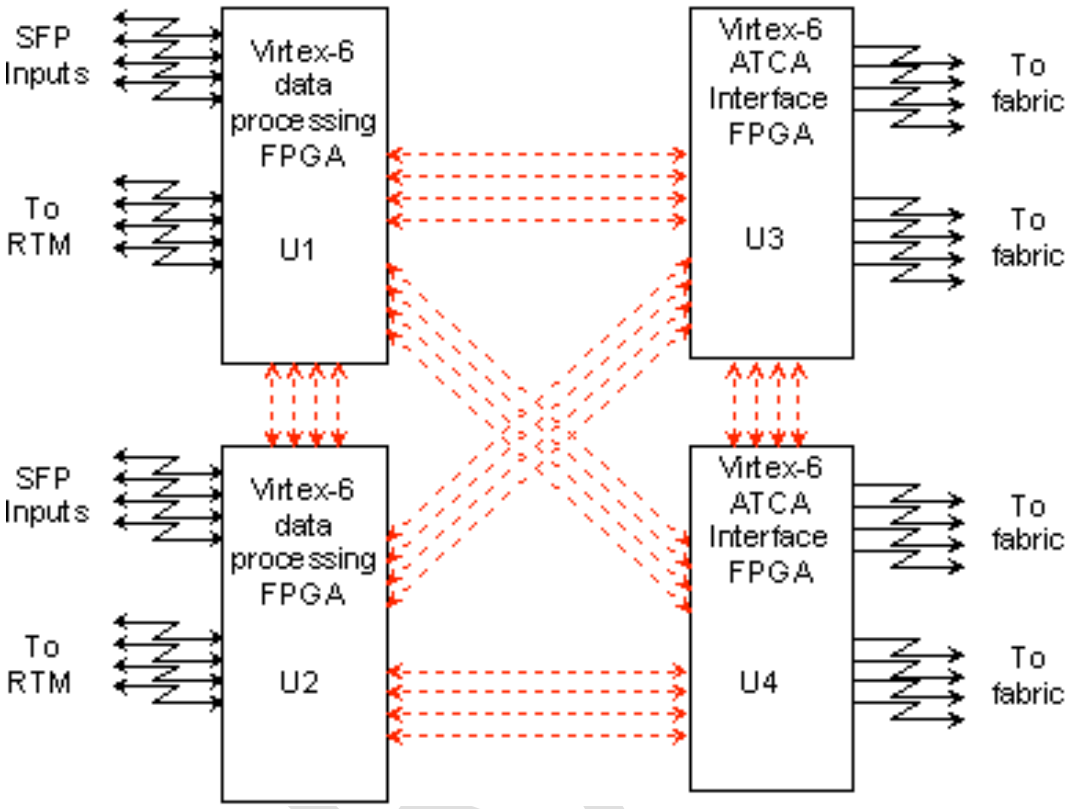

Figure 3 – Internal mesh architecture of FLIC.

The purpose of the Spy Buffer is to

a) select records entering either of the two data processing FPGAs via the SFP input SERDES links,
b) copy the data received over the internal mesh of links to either or both of the ATCA interface FPGAs,
c) then send all fragments associated with a given selection to one processor blade in the ATCA shelf by means of one of the four 10Gb Ethernet links driving the ATCA fabric.

To implement this, specific functionality in both the data processing and ATCA Interface FPGAs is required. Section 2 of this document shall describe both in detail.

2. THEORY OF OPERATION

Each SSB module sends records to the FLIC at rates of up to 100kHz. The master record identification value is the Level 1 ID, found in the 5th and 6th words of the Record Header shown in Figure 2. The FLIC implements an event-based pipeline architecture in which sets of state machines are connected by objects referred to as Event FIFOs.

Within the pipeline of the FLIC firmware, each FTK data record is referred to as an *event,* but the reader is cautioned that as each pipeline within the firmware is not processing the entire data from the detector, this term should not be conflated with the whole-detector *event* definition used in data analysis or scientific discussions. Within the confines of the pipeline firmware of the FLIC, an *event* is simply the reception of some block of data over one fiber and the processing of that block of data by a particular pipeline within the FLIC.

An Event FIFO is a standard first word fall-through (FWFT) FIFO with an event counter plus additional flag logic. Since the size of an event is indeterminate, any state machine *writing* to the FIFO is required to set an **Event End** flag bit on the 2nd to last word of the event being written. Setting this **Event End** bit has the effect of incrementing the internal event counter. If the internal event counter is non-zero, an **Event Available** flag is set.

A state machine on the reading side of the Event FIFO monitors the **Event Available** flag and does not begin reading the FIFO until this flag is set. Immediately upon beginning to read the event, the read-side state machine pulses the **Event Read** input, causing the counter to decrement. Thus, the counter always contains the number of full and complete events available for readout, and the **Event Available** flag is only set if there is at least one full and complete event available.

Figure 4 shows the overall architecture for one data processing FPGA, where two FIFOs – one **Event FIFO** that contains the data and one **Tag FIFO** that marks which events get copied to the ATCA Interface FPGAs are loaded in parallel. The Core Crate Receiver state machine fills both FIFOs on a per-SFP-link basis. The Level 1 ID value of each event is compared against registers as the Core Crate Receiver machine processes it, and the selection result is saved in the **Tag FIFO**. Four Merge state machines then use the bits in the **Tag FIFO** to select which events of those read from the **Event FIFO** are copied to additional **Event FIFO** structures for the SERDES links that connect to the two ATCA Interface FPGAs.

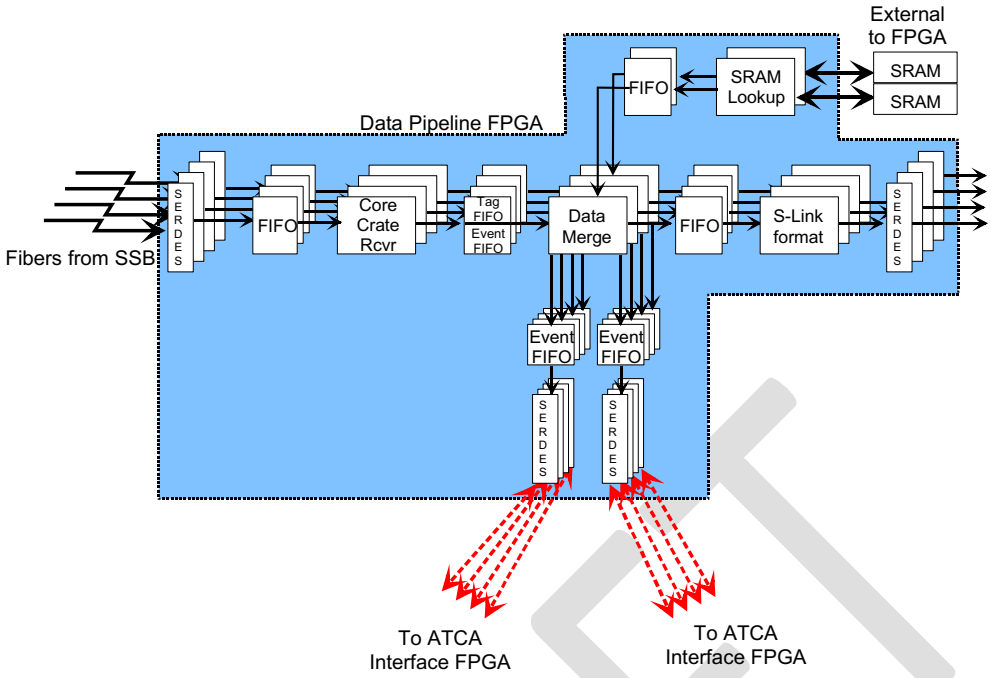

Figure 4 – Details of data flow in pipeline processor FPGA

2.1 Event Selection Methodology

The pipeline processor FPGA code for each of the two data processing FPGAs shall implement two registers for event selection. When events are received by the CoreCrateRcvr state machine, the value in each of the two registers shall be bit-by-bit ANDed with bits 23:0 of the Level 1 ID value of the event. If the resulting value of the bit-by-bit AND operation is non-zero, the event is selected for copying to the processor blade. The first register will select which events are copied to FPGA U3. The second register will select which events are copied to U4. If a selection register is zero (default power-up value), no events are selected.

2.1.1 Tag FIFO usage

The CoreCrateRcvr state machine in each of the four pipelines of a processing FPGA will generate two bits for every event as described in Section 2.1, Event Selection Methodology. These two bits shall be 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 event status, shall be written to the **Tag FIFO** for a total of four bits per event. The **Tag FIFO** shall be 32 events deep to ensure that the number of events that can be held by the **Tag FIFO** is greater than any foreseen number of events in the **Event FIFO** between the Core Crate Receiver and Merge state machines. The **Event FIFO** is 16,384 words deep, sufficient to hold 10 average-size events. The input logic of the FLIC truncates events larger than ¼ the size of the **Event FIFO.**

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

Bits 3:2 as read from the **Tag FIFO** shall define the Event Status field, interpreted thusly:

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

Bits 3:2 as read from the **Tag FIFO** shall define whether events are to be copied and also the destination of selected events. Bit 1 is the U4 Copy bit. If set, the event will be copied to FPGA U4. Similarly, bit 0 controls whether the event will be copied to FPGA U3.

2.1.1.1 Definition of a Truncated Event

A Truncated Event is one that has been cut short by the FLIC based upon one of two conditions:

A. The full record as sent cannot fit within the available space within the FIFOs of the pipeline, or
B. The device sending data to the FLIC overruns the FLIC's input buffer having ignored a flow control message sent from the FLIC.

In the first case the FLIC performs the size test by simply counting tracks processed and truncating when a maximum track count is exceeded. The FLIC will truncate the data at a Track Header boundary, throwing away all excess data until the Record Trailer for the offending event is processed.

In the second case the FLIC cannot be aware of how much data has been lost and can only attempt to re-synchronize on the next Record Trailer. It is possible in the overrun condition that the FLIC may lose multiple events and the pipeline may go out of sync with the other pipelines. This could affect the operation of the state machines in the ATCA Fabric FPGAs that collect and build the Spy Buffer data, and clearly must be forwarded to the processing blades in the ATCA shelf, thus indication of Truncated Events must be passed over to the other FPGAs via the inter-FPGA SERDES links.

A reasonable definition of the maximum number of tracks that can be processed without risk is defined by the FIFO sizes in the pipeline logic, the size of the input records into the FLIC and the thresholds at which flow control from the FLIC to the data source is asserted. The FIFOs used in the pipeline are 16,384 words deep and flow control is asserted by the FLIC whenever any FIFO becomes ¾ full. In the worst case scenario the FLIC has just begun processing a record when it must assert flow control, with 4,095 words remaining until a FIFO goes completely full. Further assuming that the FLIC can empty no data because the destination the FLIC drives is already asserting flow control, the maximum number of tracks that may be processed is given by the amount of depth remaining, minus record header & trailer size, divided

by the size of a track, or (4095 – 14 – 6) / 28. Thus the maximum # of tracks that should be allowed before truncation is 145.

2.1.1.2 Definition of a Malformed Event

A few words in the input data stream have fields with fixed data values, used by the CoreCrateRcvr state machine to synchronize itself to the incoming data. At these junction points (first word of a Record Header, first word of a Track Header and four words within the Record Trailer) the state machine can detect an error. In order to allow processing of the malformed event, the CoreCrateRcvr state machine will attempt where possible to save whatever data can be saved. The position of the fixed data values and the structure of the CoreCrateRcvr machine only allows for a limited set of results, as follows:

- If the fixed data in a Record Header is incorrect, the entire event is lost and no data is processed or copied across the inter-FPGA links. An error flag is asserted and an error counter is incremented.
    - No data will be copied to the Fabric FPGAs in this case, so there is no Event Type code associated with this condition.
- If there is a word received at the Track Header position that does not match the required bit pattern, the malformed event error flag is asserted. The event processed so far is already in the output FIFO, so a proper Record Trailer manufactured by the state machine is pushed into the output FIFO and all incoming data after the mismatch is thrown away until the CoreCrateRcvr machine can resynchronize to the incoming data stream.
    - The state machine handles an error of this type in the same way that a correct but over-long event is truncated (see Section 2.1.1.1). Such an event is technically both Malformed and Truncated, and is assigned Event Type “11”.
    - There is no guarantee that the CoreCrateRcvr will be able to resynchronize within the same event, but it is expected that this will be the usual case.
- If one of the fixed values in a Record Trailer is wrong, the event is also marked as malformed and the rest of the Record Trailer is generated by the CoreCrateRcvr state machine, with all incoming data is thrown away until a recognizable Record Trailer is seen.
    - This case will cause the loss of at least one event, as the resynchronization process is dependent upon a fully correct Record Trailer being received.

2.2 Data Transfer between pipeline and fabric 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 2.

2.3 Reception and processing of copied events

The ATCA Fabric FPGA SERDES blocks shall receive information from each of the eight SERDES links (four per processing FPGA). A variant of the CoreCrateRcvr state machine will lock onto the format and then load **Event FIFOs** with the received events. A **Filler Machine** scans the top event in each **Event FIFO** and marks which Level 1 ID value is present in each, as shown in Figure 5. Using a scratchpad RAM, the **Filler Machine** copies all event fragments with the same Level 1 ID into one **Assembly Region** in the DDR memory.

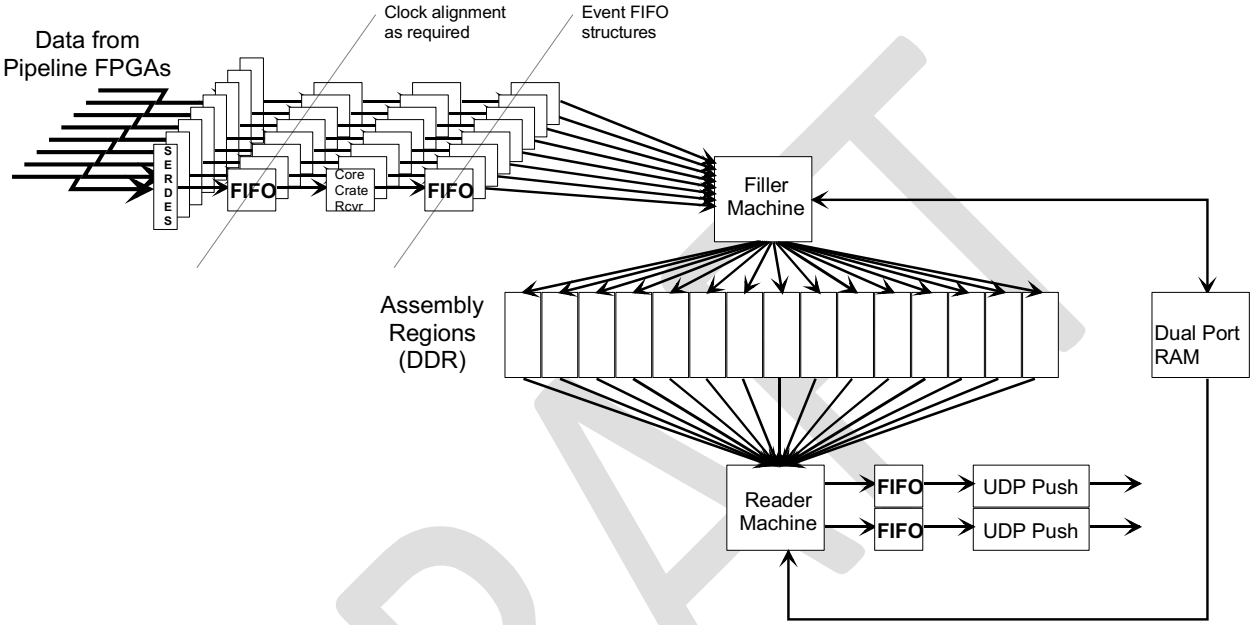

Figure 5 – Overall architecture of Spy Buffer logic in ATCA Fabric Interface FPGAs

The scratchpad dual-port RAM is used to keep track of which Level 1 ID is being assembled into each assembly region, to hold the total data length of the data in each assembly region as fragments are being added, and to hold the state of each assembly region. When a region is ready for readout, the **Reader Machine** is signaled and the region is read out into one of two FIFOs in ping-pong fashion. **UDP Push** state machines monitor their respective FIFOs and whenever an assembled event (half of a full FTK event from the detector standpoint) is available, it is sent to the processor blade on the other end of the ATCA backplane connection.

The number of Assembly Regions that should be supported is based upon the DDR size and the maximum record length per SERDES link. The DDR memory used in the FLIC is a 1Gbyte memory, and Section 2.1.1.1, Definition of a Truncated Event, defines the maximum record length as 4096 words (8192 bytes). Thus the size of an Assembly Region is defined as (8192 * 8) bytes, allowing for 16,384 Assembly regions. This calculation does not take into account added information for new header/trailer information needed within the Assembly Region logic, so for safety the DDR is defined as containing at maximum 8,192 Assembly Regions each 131,072 bytes (128K) long. The assembly logic is not required to implement all 8,192 regions. The minimum number to implement is defined as eight.

2.3.1 Masking of unused links

Akin to the processor pipelines, the ATCA fabric FPGAs shall implement a masking register. Each inter-FPGA SERDES link will only be processed if it is selected. Masked links will still accept data from the processor FPGAs but said data will be ignored and immediately flushed.

2.3.2 Assembly Region Logic

Each assembly region shall contain fragments associated with one and only one Level 1 ID value. The assembly region shall reserve five words at the beginning of the region for header information, followed by an indeterminate amount of data from the fragments that follow. The header field will be filled by the filler machine when the fragments have been collected, prior to alerting the reader machine that the region is available for readout[1].

Each assembly region in the DDR memory will be in one of four states at any given time:

1. Empty and available for filling
2. In the process of being filled
3. Complete and ready for readout
4. Incomplete but ready for readout (forced).

The "filler" machine collects data from each SERDES link in round-robin order. The index of which assembly region the fragment will be copied to is controlled by the dual-port RAM. One entry in the dual-port RAM is assigned for each non-empty Assembly Region available. The Level 1 ID value ***is not used in any way*** to generate the address to the dual-port RAM. Instead, the dual-port RAM is implemented as a circular buffer of assembly region status values, of which the Level 1 ID is one of those values. Unlike a traditional circular buffer, however, multiple elements between the "head" and "tail" of the circular buffer may be simultaneously active. 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.

2.3.3 Processing of out-of-order L1 ID values

While the expectation is that the normal operation will be to find all fragments for a given L1 ID available all at once, the "filler" machine will skip to another assembly region and start filling that region with the next available L1 ID if data from a link is missing. Should a fragment for the incomplete L1 ID arrive later, it will be added to the correct assembly region by use of the dual-port indexing RAM. Generally, however, regions are expected to be filled and read out in simple round-robin order with only a few assembly regions in use at any given time (one being filled, one being emptied, and one ready to be emptied).

However, it is possible that the "filler" machine will receive an L1 ID from one SERDES that does not match the L1 ID from the previous SERDES, but before all fragments expected from the previous L1 ID value have been found. In this case the dual-port RAM location for the

[1] If simpler, the five reserved words may be eliminated from the assembly region and, instead, read directly from the DPRAM. However, this does increase the size and complexity of the DPRAM and thus must be a design-time decision.

previous L1 ID, and the associated assembly region, will be left in the *filling* state. When an L1 ID mismatch occurs and there is at least one incomplete, non-empty assembly region in play the "filler" machine will be required to search the dual-port RAM to determine if the new L1 ID has previously been assigned to an assembly region. If true, the "filler" adds the fragment to the correct spot. If not, a new entry in the dual-port RAM is made at the next ordinal address in the circular buffer and a new assembly region assigned. The expectation is that at some time in the future that the laggard fragment will arrive, the assembly region marked *complete* and the readout performed before the circular buffer address wraps around and a collision occurs.

The above implies that the general design of the "filler" state machine will be as shown in Figure 6.

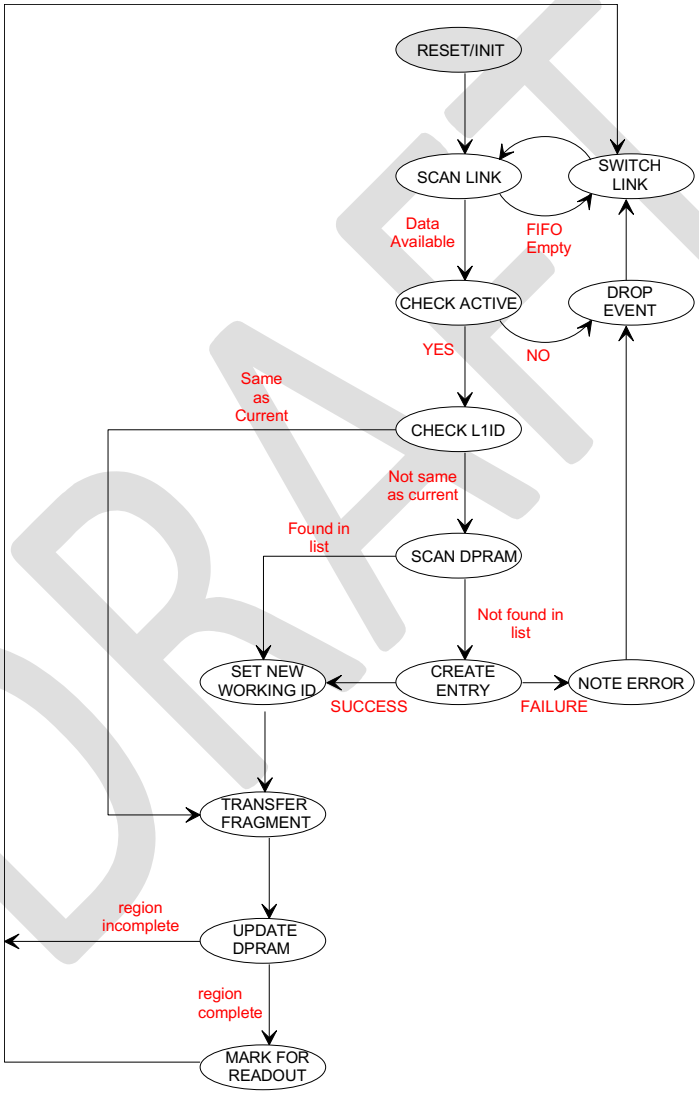

Figure 6 – Filler machine state diagram.

2.3.4 Forced readout of incomplete assembly regions

Gaps in the reception of event fragments that are not resolved before the circular buffer wraps around will result in the "filler" machine wanting to use a location at which the new entry would be placed is already marked as being in the *filling* state. In this case, because a part of the

data for that assembly region never arrived, that region is marked for *forced* readout. The "filler" machine then continues to advance, marking forced readouts as required until an available region is found to start filling. This provides a timeout mechanism to ensure that all regions are sent to the blades even if an input link to the FLIC fails or gets out of sync.

As the delay time from when a given assembly region becomes "as full as it will get" and when the readout of that region is forced depends upon the size of the dual-port RAM (and thus the number of assembly buffers), the number of buffers should be kept small to minimize the delay.

2.3.5 Final operations when an assembly region is complete

When the final fragment that is expected for a given assembly region has arrived and has been copied into the assembly region, a final set of housekeeping operations are performed before the region is marked *ready for readout*. These final operations copy data from the dual-port RAM regarding the status & size of assembly region into the first few words of the assembly region.

2.4 Format of data in an assembly region

Each DDR assembly region consists of a fixed size *Assembly Header*, followed by at least one *Event Fragment,* followed by a fixed size *Assembly Trailer,* as shown in Figure 7.

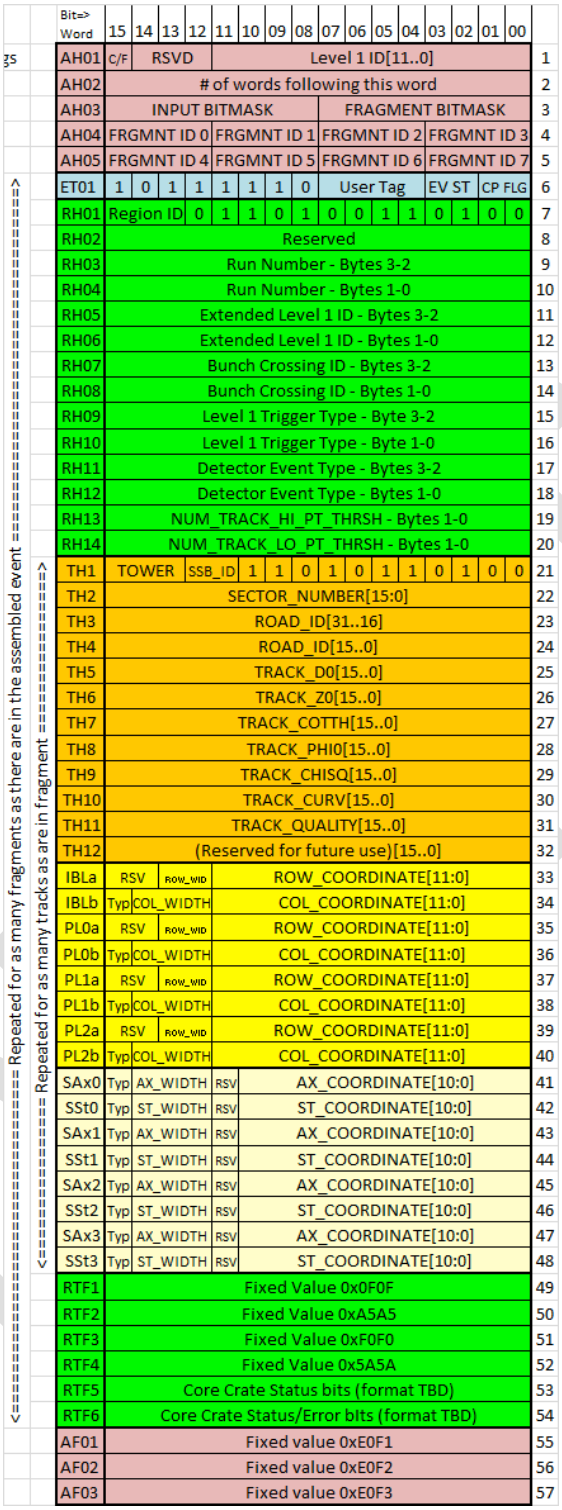

Figure 7 – Format of data in an Assembly Fragment

The Record Header, Track Header, Track and Record Trailer data is exactly copied from the SERDES link without modification. Each of these *Event Fragments* is the data received from one SERDES for the Level 1 ID value associated with the entire assembled event. 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.

When assembly of fragments into a particular assembly region begins the five words of the *Assembly Header* are mostly undefined, but as the fragments from each SERDES link are processed the sections are filled in. At the very end, after the last fragment is copied, the header will be completed by copying the total event length into the 2nd word of the *Assembly Header.* It is expected that the writing of the three fixed *Assembly Trailer* words, and the length value in the *Assembly Header*, are the final steps before the assembly region is marked as complete. The total length and *Assembly Trailer* are written into the assembly region irrespective of whether the event was complete or forced.

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.

2.4.1 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 • "0111" indicates data from the SERDES associated with pipeline index 3 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 1 – Assembly header field definitions.

2.5 Reader Machine operation

The Reader machine scans the Dual-Port RAM looking for assembly fragments that are ready for readout, either by completion or forcing. When a ready fragment is available the Reader machine copies the data, ***without any modification whatsoever***, to one of two output FIFOs. A simple ping-pong technique is assumed, although an option to send all data to just one of the two 10GbE ports of a given Fabric FPGA is allowed. These FIFOs are **Event FIFO** structures as previously described, and the Reader machine sets the **Event End** flag when each assembled event is fully copied.

When the event has been fully copied from the assembly region the Dual-Port RAM segment associated with the assembly region is then released, changing the status of the region from *complete* or *forced* back to *empty.*

2.6 UDP Push processes

Two copies of a UDP Push process monitor each of the two **Event FIFOs** that are filled by the Reader machine. Each UDP Push process, upon noting that a full and complete event is available, pulls that event out in 576 byte blocks (minimum IPv4 datagram size) that, when padded with the standard 8-byte UDP header plus 20-byte IP header, will form packets 604 bytes long. The firmware shall not support jumbo packets. Implementation of a register to set a larger packet size is allowed but not required, so long as the post-reset value of said register is 576.

Such register may not allow selection of 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.