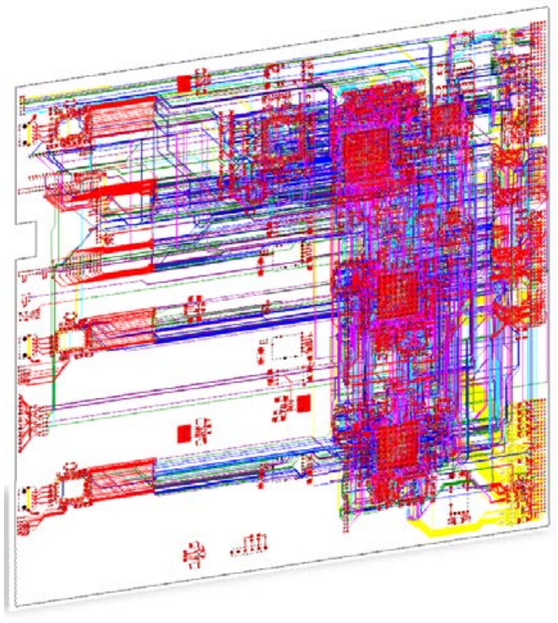
IBL ROD control and data path specification



Bing Chen
University of Washington EE/Physics, Seattle

Gabriele Balbi, Davide Falchieri
University of Bologna and INFN, Bologna

May 2013

Version management

Number	Date	Changes	Author
1	22/05/2013	Initial version	Davide Falchieri

Contents

Purpose of this document 4
ROD firmware 5
ROD firmware 5
ROD control path: VHDL blocks 7
ROD control path: registers 12
ROD data path: VHDL blocks 13
Formatter 13
Formatter: output data 16
Formatter: registers 18
EFB 20
EFB: registers 26
Router 27
Router: registers 28
References 29

<u>Purpose of this document</u>

This document arises from the need of a clear and ordered list of the features for the IBL ROD firmware. The ultimate goal is to collect a list of most of the needed functionalities, so to be able to assign the design of the blocks to different teams.
Any suggestion on what is needed is obviously welcome.
As a reference, most of the material presented is taken from the “ATLAS Silicon Readout Driver Users Manual” [1] and from the related VHDL code.
The idea is to stick as much as possible to the firmware architecture of the Pixel ROD. The main architectural changes with respect to it are due to the following differences:

1. the Pixel ROD has to manage data coming from up to 96 links, while the IBL ROD has to manage data from 32 links (= 32 FeI4 chips). So, for instance, the format of the “EFB Event ID and Dynamic Mask Data” features 192 bits in the Pixel ROD (2 bits per link), while it features 64 bits in the IBL ROD.
2. With the new front end architecture (FeI4 vs FeI3-MCC) all the error diagnosis blocks need to be changed accordingly.

ROD firmware

The ATLAS IBL Read Out Driver (IBL ROD) electronics board, used in the ATLAS IBL subsystem, is designed specifically to interface with IBL modules and their front-end electron ics (FeI4). Each IBL ROD interfaces directly with one IBL Back-of-Crate (IBL BOC) electronics board that connects to the IBL modules using fiber optic links. One Timing Interface Module (TIM) electronics board [2] distributes the global timing and trigger information to 15` IBL ROD modules. The IBL ROD communicates with a system host through VME and Ethernet connection. Formatted event data are transmitted off the IBL ROD back to the IBL BOC through the crate backplane.

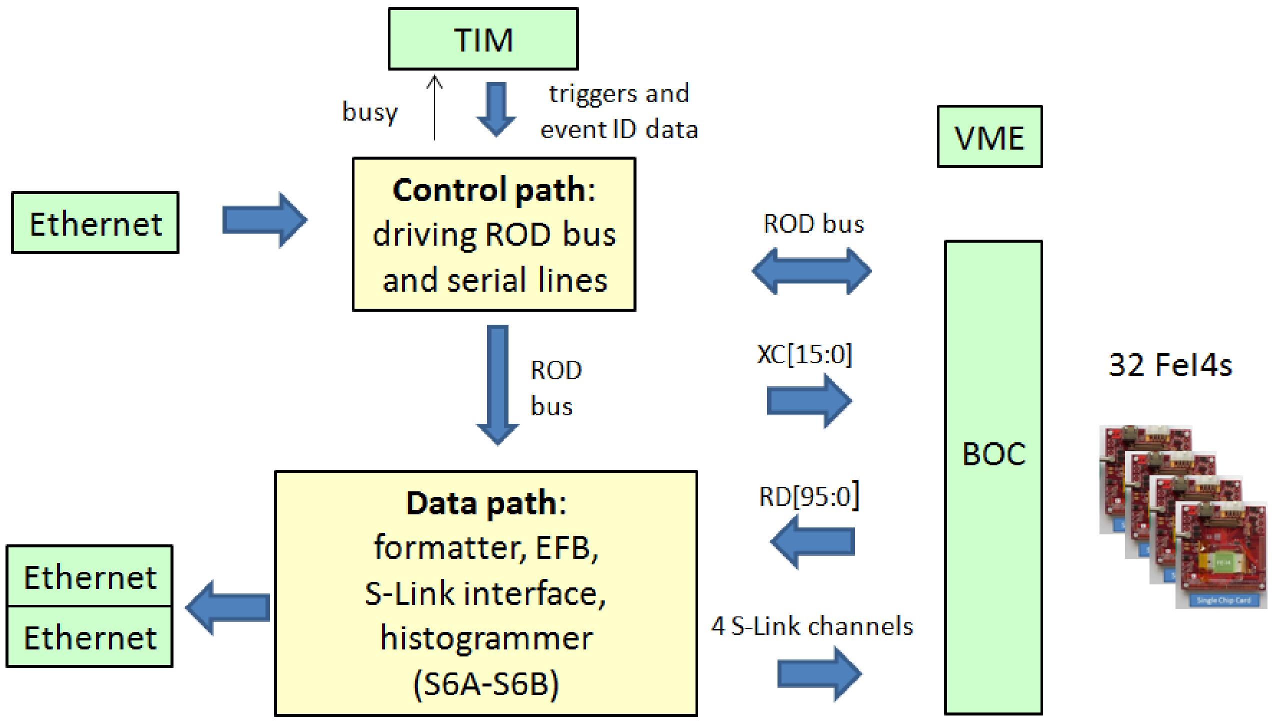

TIM
busy
triggers and event ID data
VME
Ethernet
Control path: driving ROD bus and serial lines
ROD bus
XC[15:0]
32 FeI4s
ROD bus
BOC
RD[95:0]
Data path: formatter, EFB, S-Link interface, histogrammer (S6A-S6B)
Ethernet
Ethernet
4 S-Link channels

Figure 1: IBL ROD general structure

The IBL ROD-BOC system deals with 32 FeI4 chips and 4 S-Links.
One basic function of the IBL ROD is to transmit control commands and configuration data through the IBL BOC to the FeI4 chips. The transmit data is sent to the modules as 40 MHz serial data streams and can be Level 1 triggers, Event Counter resets, Bunch-Counter resets, calibration commands or module register data.
A second basic function of the IBL ROD is to receive data streams from the FE chips, after the IBL BOC performs 8B/10B decoding on 160 Mb/s serial data lines (one per chip). The Field Programmable Gate Arrays (FPGA) on the IBL ROD are designed to detect and mark format errors in the event data, build event fragments and transmit the event fragments to the Read Out Link (ROL) system (via the IBL BOC).

In normal running (physics data taking) the TIM supplies the trigger and event description data to the IBL ROD. The trigger is detected and expanded into the module trigger commands required by the IBL FE modules. The FE Trigger code is sent to a 16-bit wide mask gate that sends the trigger to the active modules.
Here’s the sequence of operations:

- At the start of the run, the PPC sets the active links for the default mask;
- after the IBL ROD sends the Trigger code to the FE modules, a set of dynamic mode bits are sent to all of the Formatters;
- after sending the L1 pulse to the IBL ROD, the TIM transmits serial data that describes the rest of the ATLAS trigger information.
- when the TIM data for a specific trigger has been transmitted, that data (a dynamic mask and the number of event counter resets) are sent to the Event Fragment Builder (EFB) block. After a delay of a few clock periods to allow the event description data to propagate to the EFB, a token is sent to the Formatters to start the readout of the incoming data.

ROD control path: VHDL blocks

The ROD Virtex5 FPGA coordinates all the real-time control operations that are required for normal data taking, module calibration and on-board diagnostics; it connects the data path FPGAs (the slaves) and the BOC board to the PPC and interfaces with the TIM module for access to the TTC data and commands.

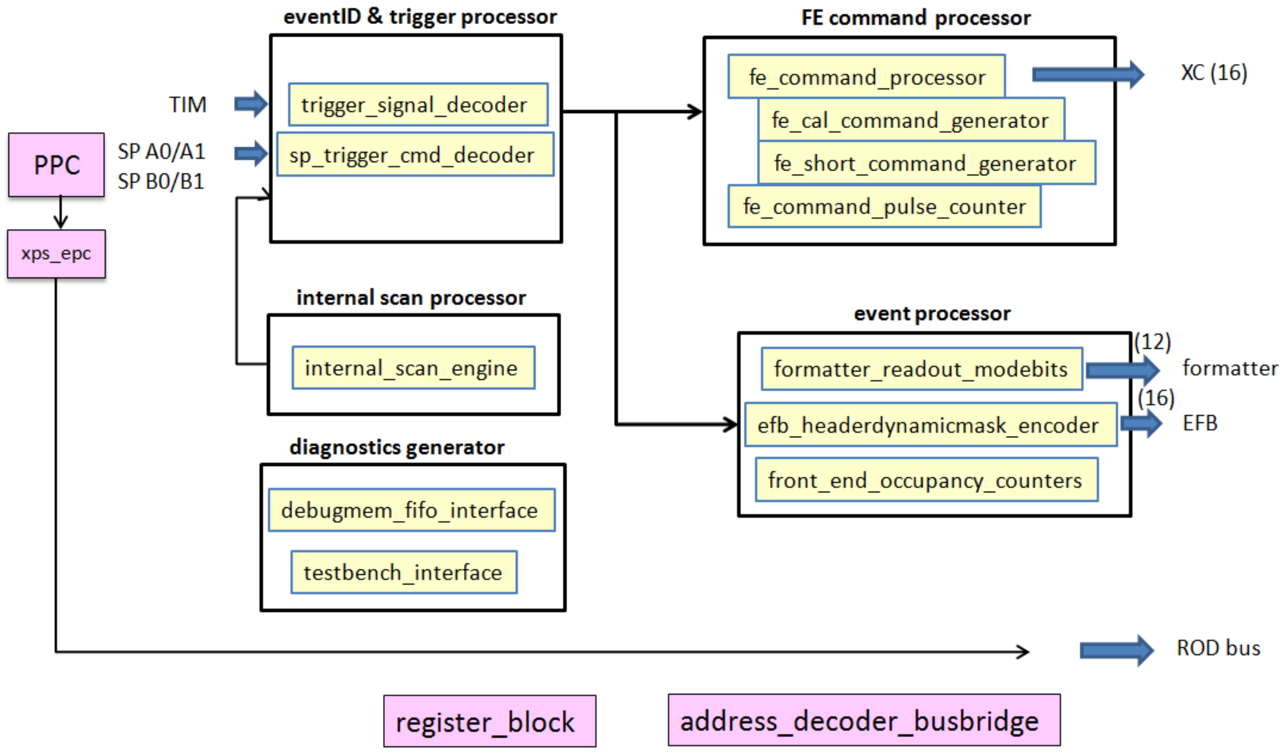


Figure 2: ROD control path

eventID & trigger processor

The Event ID and Trigger processor is the primary collection of real-time event building operations in the ROD. Its primary function is to gather triggers and event description data from all of the trigger sources, format the data and transmit triggers to the FE Command Processor and data to the Event Processor functions in the V5 FPGA. Trigger sources available to and on the ROD include the TIM module, the Internal Scan Processor, the PPC and the Diagnostics Generator.

trigger_signal_decoder

Implements the serialL1ID and TriggerType serial decoders with simple no error detection shift registers. There is no examination for consistency with local L1/BC counters, proper size or any sort of bit error or CRC checking.
Serial input data streams are all positive logic, LSB first:
Serial ID: leading zero(s) & 1 & 24bits L1ID & 12bits BCID & trailing_zero(s)

Serial TT: leading zero(s) & 1 & 8+2bits trigger type & trailing_zero(s)

sp_trigger_cmd_decoder

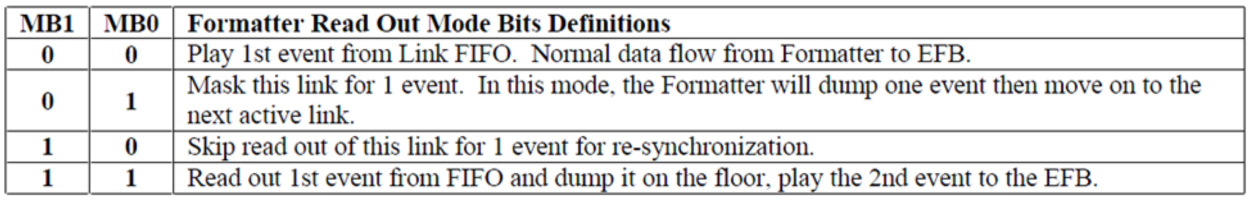
This block counts L1 Triggers issued by the PPC during calibration and special testing modes. Command bit streams to front end modules are:

l1accept:	"11101"	
bcr:	"101100001"	Bunch Counter Reset
ecr:	"101100010"	Event Counter Reset
cal:	"101100100"	Calibration Trigger

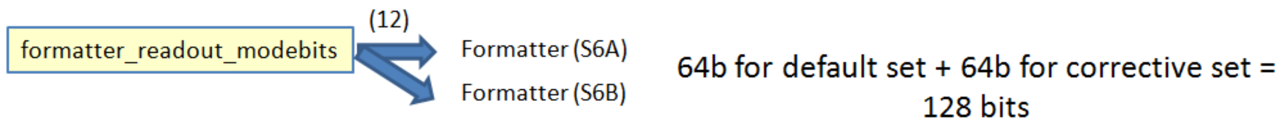
When a L1A is detected on one of the serial port inputs, the ID values are loaded into the queue FIFO on the rising edge of the next system clock. Each serial port input has a L1 ID counter. The L1ID counters are reset to 0 when an ECR command is detected and incremented by 1 when and L1 trigger is detected. The block also contains 2 free running Bunch Counter ID (BCID) counters. When a Bunch Counter Reset (BCR) is detected, the BCID counters are reset to 0. When a L1A is detected, the value of the BCID counter at that time is latched into register and can be used in the Event Header as the event BCID.

formatter_readout_modebits

The Formatter Read Out Mode Bits Encoder sends the appropriate Mode Bit mask to the Formatters when the ROD detects a L1 trigger from any source.
The Formatter Read Out Mode Bits are stored in a LUT in internal Block RAM in the V5.
The ROD has 2 sets of Read Out Mode Bits: the default set and the corrective set.
The default mode bits are sent to the Formatters for all "normal" (no correction required) triggers received from the TIM. If the ROD is using the PPC Serial Port Outputs (SP_A0/A1 & SP_B0/B1) for trigger generation, the Default Mode Bits are sent when a trigger is detected on SP_A and the Corrective Mode Bits are sent when a trigger is detected on SP_B.



MB1	MB0	Formatter Read Out Mode Bits Definitions
0	0	Play 1st event from Link FIFO. Normal data flow from Formatter to EFB.
0	1	Mask this link for 1 event. In this mode, the Formatter will dump one event then move on to the next active link.
1	0	Skip read out of this link for 1 event for re-synchronization.
1	1	Read out 1st event from FIFO and dump it on the floor, play the 2nd event to the EFB.


formatter_readout_modebits
(12)
Formatter (S6A)
Formatter (S6B)
64b for default set + 64b for corrective set = 128 bits

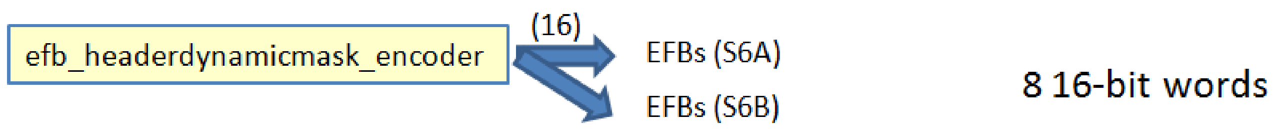
efb_headerdynamicmask_encoder

The EFB Header ID and Dynamic Mask Encoder is used to transmit the trigger information that is sent from the TTC system or generated internally on the ROD to the EFB prior to the

arrival of event data from the FE modules. The EFB Header/Dynamic Mask Encoder is implemented in the V5 and consists of 5 FIFOs, 2 Look-Up-Tables (LUT) and a FSM to control the Event Header building task. The primary function is to collect all of the trigger description data, format and send the Event ID Header and the default or corrective Dynamic Mask information to the EFB.
The Header ID values are used to generate the Event Fragment Header and consists of the event specific L1ID, ECRID, BCID, Atlas Trigger Type, TIM Trigger Type and the ROD Event Type. The source for the Header ID values depends on the current mode of the ROD.
If the TIM Trigger Signal Decoder is enabled, the L1ID, BCID and Atlas and TIM Trigger Types will be sent to the ROD on the TIM TTC Bus inputs and, when received, loaded to the Header/Dynamic Mask Output FIFO. A L1A from the TIM will load the ROD Trigger Type from the internal memory location to the Header/Dynamic Mask Output FIFO. If the ROD is using the PPC Serial Port Outputs (SP_A & SP_B) for triggers, the Header ID information is generated using internal counters and registers in the V5 for each input.
The Dynamic Mask Bits, 2 bits per link, enable dynamic synchronization error correction in the EFB on a link by link basis. A corrective mask, used for error correction, can be written to the Corrective Dynamic Mask Bits Register and sent to the EFB when the ROD receives an L1A from the TIM and the “Corrective Mode Bits and Dynamic Mask Ready” bit is set in the V5 Main Control Register (0x00404410, Bit 23). When a corrective mask is sent to the EFB, the Corrective Dynamic Mask bits are written into the Default Mask LUT.
When the ROD receives a “normal” (no correction required) L1A, the Default Dynamic Mask Registers are sent as the Dynamic Mask. If the ROD is using the Serial Port Inputs (SP_A & SP_B), the Default Dynamic Mask Bits are sent when a trigger is detected on SP_A and the Corrective Dynamic Mask Bits are sent when a trigger is detected on SP_B. The following table shows the definitions of the Dynamic Mask Bits.

DMB[1:0]	EFB Dynamic Mask Bits Definitions
00	No Change to L1 ID
01	Increment L1 ID by 1
01	Decrement L1 ID by 1

word 0 : L1 ID[15:0]
word 1 : ECR ID[7:0] & L1 ID[23:16]
word 2 : BC ID[11:0] & RoL & BOC OK & TIM OK
word 3 : TT - ROD[5:0] & TIM[1:0] & ATLAS[7:0]
word 4 : Dynamic Mask 0 for Links [7: 0]
word 5 : Dynamic Mask 1 for Links [15: 8]
word 6 : Dynamic Mask 2 for Links [23:16]
word 7 : Dynamic Mask 3 for Links [31:24]

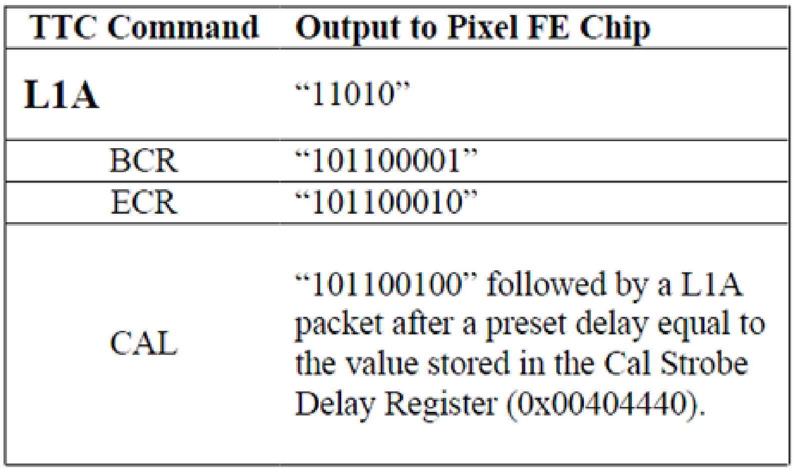


front_end_occupancy_counters

For each link, this block increments a 4-bit counter when a L1 trigger is issued to the FE chips and decrements it when a trailer is written in to the Link Formatter FIFO. The FE Occupancy Counters can be used as a diagnostic tool to monitor the occupancy of the Formatters and the status of data flow from a FE module.
The expectation is that the Front End Occupancy Counters are to be used with periodic resets to determine when the ROD can start configuring FE modules that are not acting properly. The counter can also be used as a diagnostic on the performance of the system and they are used by the Internal Scan Engine to indicate when a new trigger can be issued to the FE Modules during a calibration run.
The “FE Occupancy Counters All Zero” bit in the V5 Main Status Register is set if all of the FE Occupancy Counters have the value 0. The “All Zero” bit can be used to indicate that all modules have finished sending event data to the ROD. It will have a value of 0 if any Counter has a value greater than 0.
Each link counter can be reset independently of all others by setting the bits in the FE Occupancy Counter Reset Register that correspond to the inactive link. The counters can be disabled individually by setting the bits in the FE Dead Data Link Mask Register that correspond to links that are turned off at the Formatter. For diagnostic purposes, the FE Occupancy Counters can be loaded with a pre-set value, the same for all links.

fe_command_processor

The FE Command Processor accepts trigger and timing commands from the TIM module, PPC, Internal Scan Engine, Diagnostics Generator and distributes these signals to the BOC module (via the slaves) and internally to specific ROD functions. The primary operation of the FE Command Processor block is to detect all input signals that correspond to L1 Accept (L1A) triggers, Event Counter Reset commands, Bunch Counter Reset commands and Calibration Strobe commands as they arrive on the ROD.
The values set in the FE Command Link Mask Registers determine which FE Modules will receive command streams.



TTC Command	Output to Pixel FE Chip
L1A	"11010"
BCR	"101100001"
ECR	"101100010"
CAL	"101100100" followed by a L1A packet after a preset delay equal to the value stored in the Cal Strobe Delay Register (0x00404440).

internal_scan_engine

The Internal Scan Engine in the V5 provides a speed-optimized system for running calibration scans on the ROD. The primary structure of the process allows only L1 triggers and CAL commands to be sent on the FE command links, so all ROD setup, FE module configuration data, BCR and ECR commands must be controlled by the PPC. The configuration of the Scan Engine is accomplished using VME accessible static registers that allow the user to set the specific scan variables.

diagnostics generator

testbench_interface

The Test Bench Interface is used to run internal, self-test diagnostics on the ROD that will be used primarily in production to verify the function and performance of all of the fabricated RODs. This allow for testing of the ROD with trigger rates of 100kHz. There is a second variant where triggers come from the serial data stream from the PPC and the data from slave internal RAM.

ROD control path: registers

...

ROD data path: VHDL blocks

The ROD data path is physically implemented over the 2 Spartan6 devices (slaves) on the IBL ROD. The following paragraphs try to show the blocks which are part of the overall firmware and what are their functions. The 5 main blocks of the data path are:

- formatter
- event fragment builder (EFB)
- router
- histogrammer
- S-Link interface

Formatter

The formatter block takes care of the data coming directly from the BOC: it has to manage data coming from up to 4 FeI4 chips on a 12-bit bus. It performs:

- error detection,
- 32-bit word encoding,
- data buffering,
- mode bits data managing,
- 4 queues scheduling in a fixed sequential order (from link 0 to link 3)

The formatter shall implement the following mechanisms:

- **timeout**: there is a maximum allowed time for a module to transmit event data following a L1 trigger. This value is user configurable in the *register_block* (it's name is FMT_READOUT_LIMIT). If a module does not respond with data before the counter in the Formatter reaches the timeout limit, a timeout error event is inserted into the event fragment and the following link begins to be readout next. The default value at reset for FMT_READOUT_LIMIT on the Pixel ROD is 0x800h (51.2 μs). What is a reasonable value for the IBL ROD ?
- **skipped events**: On the FeI4B chip, when the 16-word deep buffer of pending triggers is full, any additional trigger (due to a new L1A command or to multiple triggers from a multiplier greater than 1) will result in skipped triggers. These triggers are ignored, but the FE-I4B chip keeps track of how many have been ignored with the 10-bit skipped trigger counter. This counter is read out with Service Record 15, which comes out automatically if there are skipped triggers (unless explicitly disabled). The formatter extracts and saves the number of lost events from the appropriate header and inserts the required number of empty events into the data flow. The inserted empty events will contain the correct L1ID and sequential BCID and will contain a flag in the header that will indicate the number of events that were skipped for the current L1 trigger. This process performs the required function of keeping events in synchronization and informs the offline system that an error has occurred on a module.

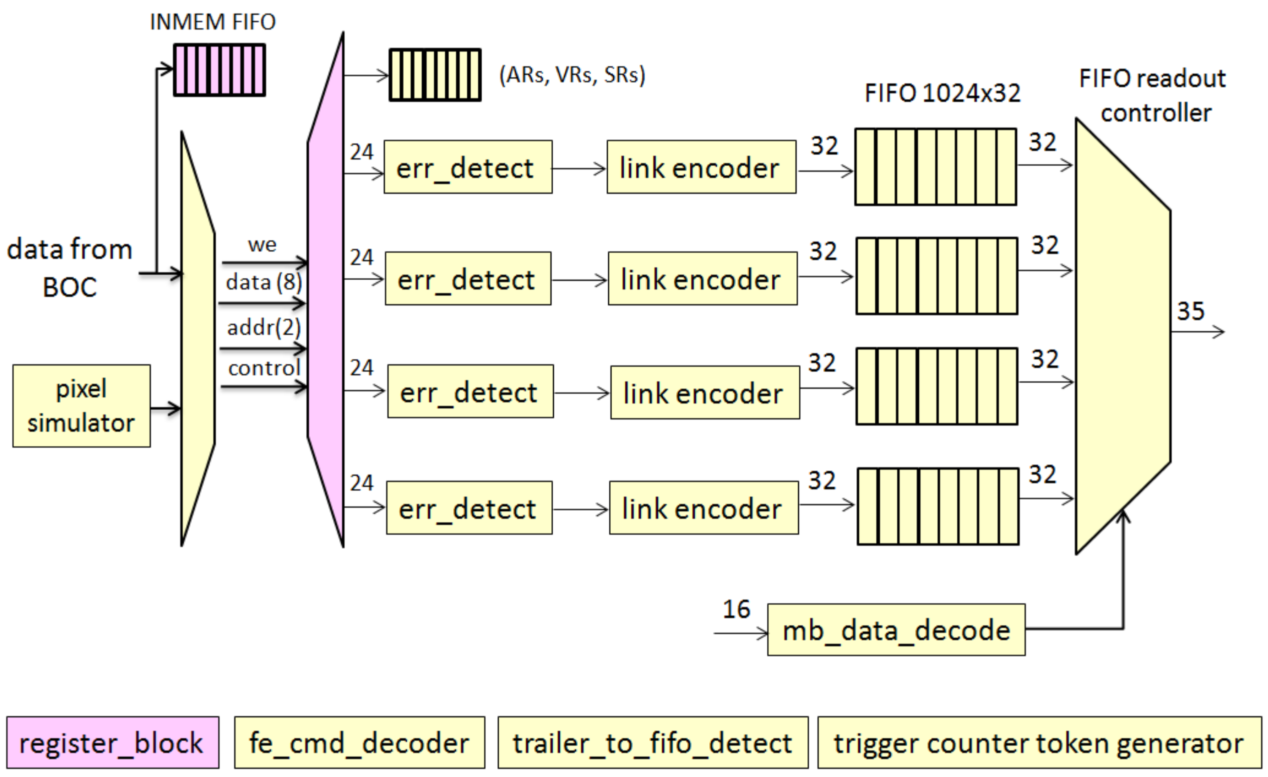


Figure 3: Formatter schematic diagram

error_detect

- This block finds error in the incoming data streams. Errors that can be detected:
    - header errors (e.g.: first word from FeI4 is not a data header)
    - word number errors (e.g.: words from the FeI4 are not a multiple of 3)
    - data check errors (e.g. row & column values found outside the allowed range)
    - readout timeout errors (e.g.: no data is received before a timeout is reached after receiveing a L1a)
    - header trailer limit errors (e.g.: the event is too large)
    - other errors which might be worthy to know about ?

  The errors detected shall be reported into the 24-bit error field inside the trailer at the end of each link event.

link_encoder

- getting 24-bit words out of the 8-bit bus from the BOC
- encoding the 32-bit data words into:
    - header (flag value, BCID and L1ID are received from the Data Header of the FeI4 event)
    - hit
    - FE flag error (service records received inside events are encoded as FE flag errors)
    - trailer

- implement expanded/condensed format,
- insert the appropriate number of events, in case they have not been received from the FE,

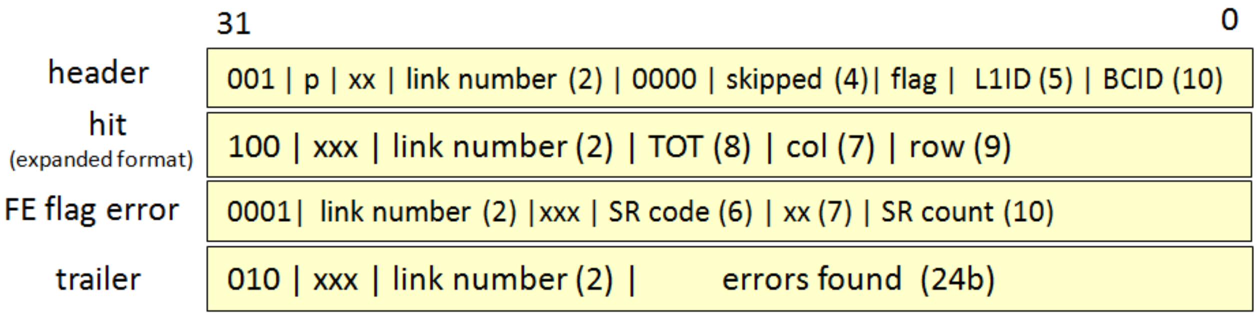

31
0
header
001 | p | xx | link number (2) | 0000 | skipped (4)| flag | L1ID (5) | BCID (10)
hit (expanded format)
100 | xxx | link number (2) | TOT (8) | col (7) | row (9)
FE flag error
0001| link number (2) |xxx | SR code (6) | xx (7) | SR count (10)
trailer
010 | xxx | link number (2) | errors found (24b)

Figure 4: link_encoder output data format

FIFO readout controller

The FIFO readout controller block:

- receives the Mode Bits from the **mb_data_decode block**, which, in turn, receives the MBs from the V5,
- reads out one event per link at a time starting from 0 to 3 (only the enabled links), in this way (in case all 4 links are enabled):
  event0 from link0
  event0 from link1
  event0 from link2
  event0 from link3
  event1 from link0
  event1 from link1
  event1 from link2
  event1 from link3
  event2 from link0
  …
- deals with the Mode Bits depending on the value received in order to allow for resynchronization between links (see doe and roe states in the FIFO readout controller state machine).

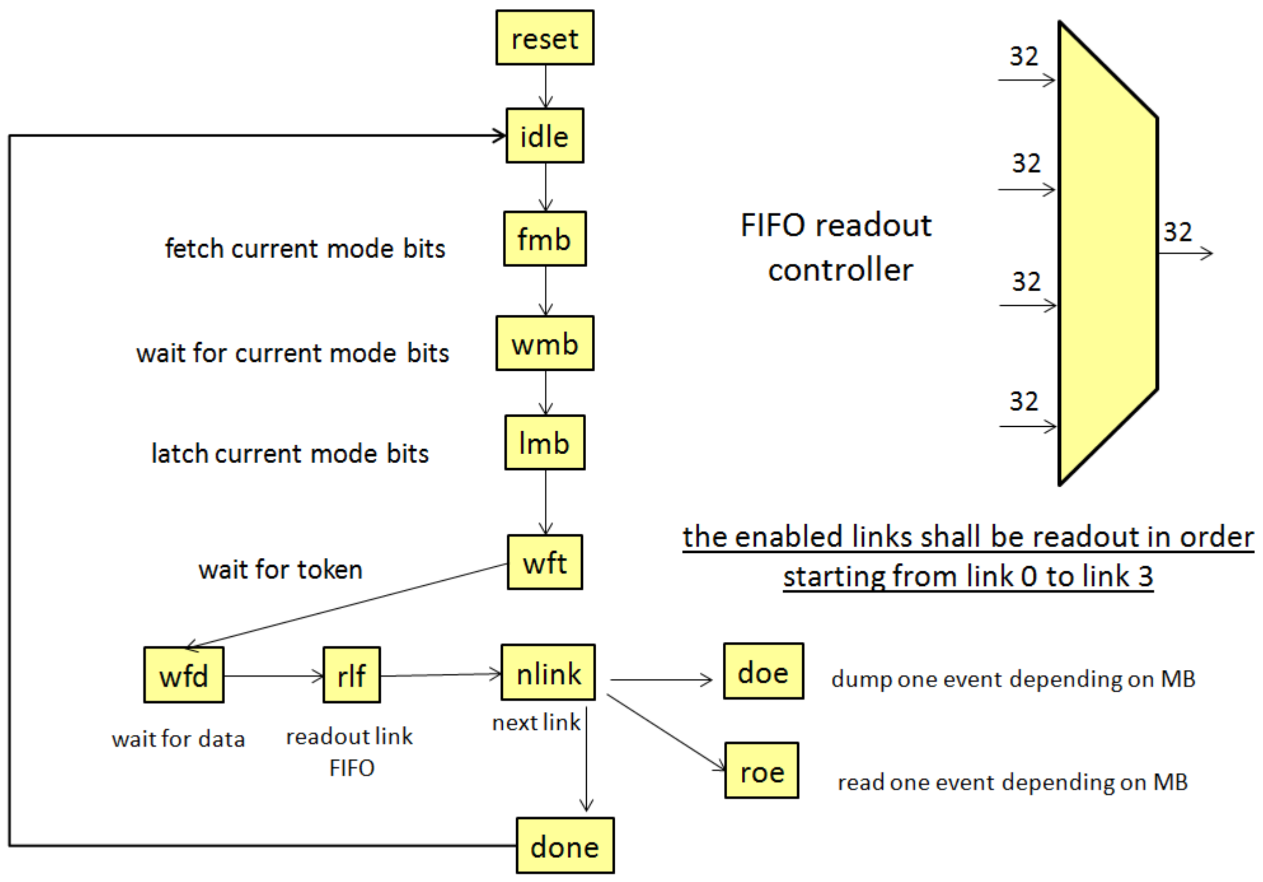


Figure 5: FIFO readout controller state machine

mb_data_decoder

This block receives the Mode Bits (64 bits for the default set + 64 bits for the corrective set = 128 bits) from the V5 on a 12-bit data bus, thus requiring 11 clock periods for the transfer. The Mode Bits are temporarily stored on a 512x24 bits FIFO before being used by the FIFO readout controller.

Formatter: output data

Name	Bits[31:0]
header	001pxxnn0000MMMMFLLLLLBBBBBBBBBB
hit (long)	100xxxnnTTTTTTTTCCCCCCCRRRRRRRRR
hit (condensed mode)	101RRRRTTTTTTTTCCCCCCCRRRRRRRRR 1CCCRRRRRRRRRTTTTTTTTCCCCCCCRRRR 1TTTCCCCCCCRRRRRRRRRTTTTTTTTCCCC 111TTTTTTTTCCCCCCCRRRRRRRRRTTTTT
FE flag error	0001nnxxxSSSSSxxxxxxxDDDDDDDDDD

trailer	010xxxnnEMPxxxxxxxxxxxxxxxxxxxxx

n: link number
F: FeI4B flag bit
L: L1ID
B: BCID
T: ToT
C: hit column
R: row column

S: service code
D: service code counter
E: timeout error bit
M: condensed mode
P: link masked by PPC
M: number of skipped triggers on the FE
p: preamble error (???)

Bit 32	TimeOut Error Bit (for all words)
Bit 33	Condensed Mode (for all words)
Bit 34	Link masked by PPC (for all words)

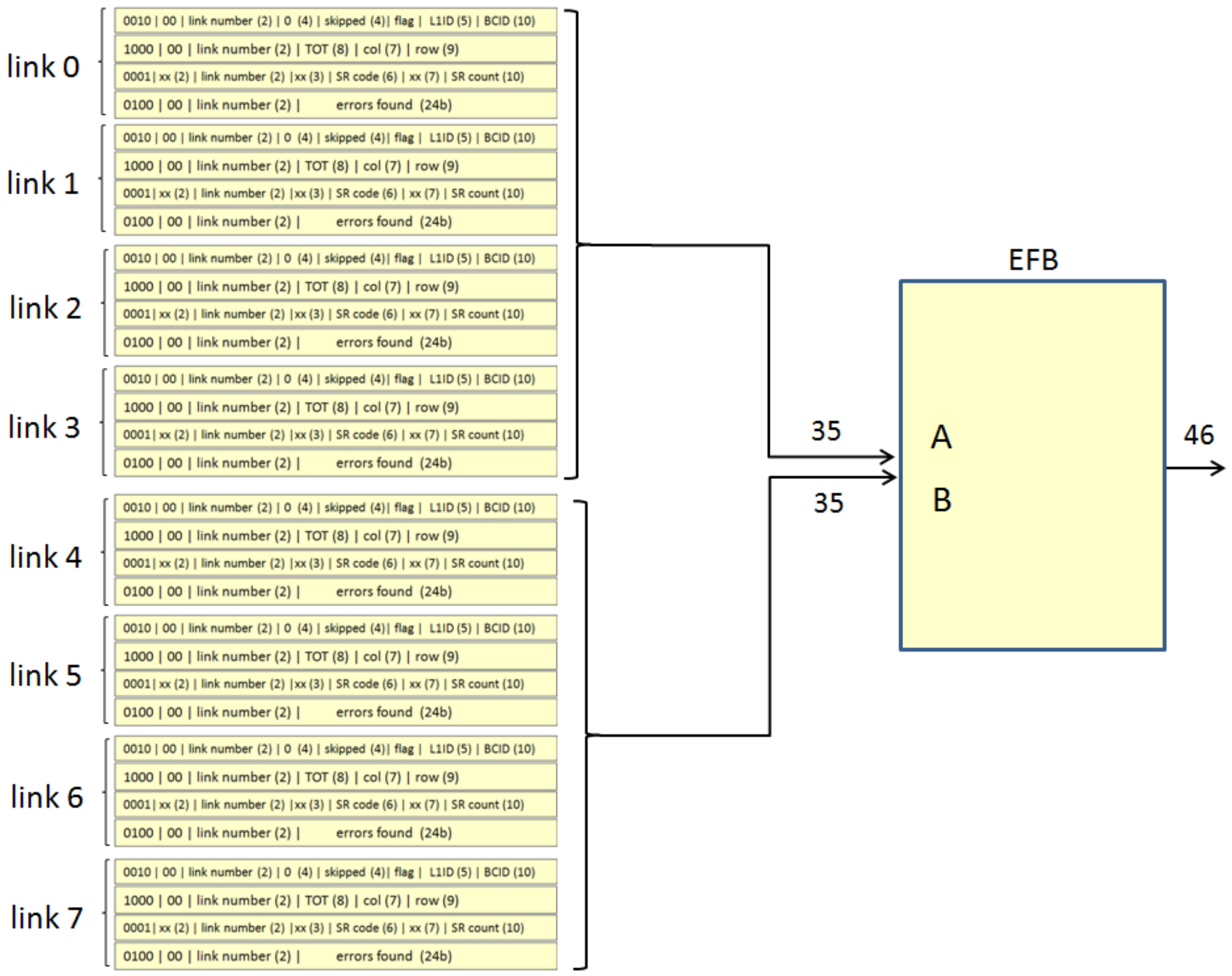

link 0
0010 | 00 | link number (2) | 0 (4) | skipped (4)| flag | L1ID (5) | BCID (10)
1000 | 00 | link number (2) | TOT (8) | col (7) | row (9)
0001| xx (2) | link number (2) |xx (3) | SR code (6) | xx (7) | SR count (10)
0100 | 00 | link number (2) | errors found (24b)
link 1
0010 | 00 | link number (2) | 0 (4) | skipped (4)| flag | L1ID (5) | BCID (10)
1000 | 00 | link number (2) | TOT (8) | col (7) | row (9)
0001| xx (2) | link number (2) |xx (3) | SR code (6) | xx (7) | SR count (10)
0100 | 00 | link number (2) | errors found (24b)
link 2
0010 | 00 | link number (2) | 0 (4) | skipped (4)| flag | L1ID (5) | BCID (10)
1000 | 00 | link number (2) | TOT (8) | col (7) | row (9)
0001| xx (2) | link number (2) |xx (3) | SR code (6) | xx (7) | SR count (10)
0100 | 00 | link number (2) | errors found (24b)
link 3
0010 | 00 | link number (2) | 0 (4) | skipped (4)| flag | L1ID (5) | BCID (10)
1000 | 00 | link number (2) | TOT (8) | col (7) | row (9)
0001| xx (2) | link number (2) |xx (3) | SR code (6) | xx (7) | SR count (10)
0100 | 00 | link number (2) | errors found (24b)
link 4
0010 | 00 | link number (2) | 0 (4) | skipped (4)| flag | L1ID (5) | BCID (10)
1000 | 00 | link number (2) | TOT (8) | col (7) | row (9)
0001| xx (2) | link number (2) |xx (3) | SR code (6) | xx (7) | SR count (10)
0100 | 00 | link number (2) | errors found (24b)
link 5
0010 | 00 | link number (2) | 0 (4) | skipped (4)| flag | L1ID (5) | BCID (10)
1000 | 00 | link number (2) | TOT (8) | col (7) | row (9)
0001| xx (2) | link number (2) |xx (3) | SR code (6) | xx (7) | SR count (10)
0100 | 00 | link number (2) | errors found (24b)
link 6
0010 | 00 | link number (2) | 0 (4) | skipped (4)| flag | L1ID (5) | BCID (10)
1000 | 00 | link number (2) | TOT (8) | col (7) | row (9)
0001| xx (2) | link number (2) |xx (3) | SR code (6) | xx (7) | SR count (10)
0100 | 00 | link number (2) | errors found (24b)
link 7
0010 | 00 | link number (2) | 0 (4) | skipped (4)| flag | L1ID (5) | BCID (10)
1000 | 00 | link number (2) | TOT (8) | col (7) | row (9)
0001| xx (2) | link number (2) |xx (3) | SR code (6) | xx (7) | SR count (10)
0100 | 00 | link number (2) | errors found (24b)
EFB
35
35
A
B
46

Figure 6: Data packet from two formatter blocks to the EFB in case all 8 modules are enabled (only one hit per event is shown)

Formatter: registers

Register	Access	Width	Purpose
Link configuration registers			
FMT_LINK_EN	RW	16	Link Enable
Each bit enables the data readout of a link in the formatter			
FMT_EXP_MODE_EN	RW	16	Expanded mode
Each bit decides if the link is encoded in expanded/condensed mode			
Error limit setup registers			
FMT_READOUT_TIMEOUT	RW	32	Timeout limit register
The FMT_READOUT_LIMIT is a user configurable value that is used to determine the time that the ROD allows for a module to transmit event data following a L1 Trigger. If a module does not respond with data before the counter in the Formatter reaches the timeout limit, a time out error event is inserted into the event fragment and the token is passed to the next link. 25ns increments. The default value at reset is 0x800h (51.2 us).			
FMT_DATA_OVERFLOW_LIMIT	RW	32	Data overflow limit
This value sets the maximum number of words played out of the Output FIFO before the ROD determines that a link has received too much data for 1 Event. At reset, the default value is set to 0x200h.			
FMT_HEADER_TRAILER_LIMIT	RW	32	Header trailer limit
This value in the HTL register sets the link FIFO Almost Full word count threshold. When HTL is true, the Formatter only writes Headers and Trailers to the Link FIFO and drops all hit data. At reset, the default value for the Pixel Formatter the default value is set to 0x700.			
FMT_ROD_BUSY_LIMIT	RW	32	ROD busy limit
The value sets the input link FIFO word count threshold, that, when exceeded, will assert and send ROD BUSY to the trigger system. Because the threshold is configurable, a link that has exceeded ROD BUSY limit may still have a room for more data, but recovery from ROD Busy limit usually requires reset of the link in error. At reset, the default RBL value for the Pixel Formatter the default value is 0x780h.			
Pixel Trigger Number of Accepts registers			
FMT_PXL_LINK_L1A_CNT	RW	32	Number of L1 Accepts
This register sets a counter value in the Formatter readout state machine that allows all of the data from a multi trigger event to be played out with one trigger.			
Diagnostics Control registers			
L1A count	R	24	L1A Counter Register
This register keeps track of all L1A triggers that are sent to the Formatter. It is cleared when the Formatter receives an ECR command.			
BCR-ECR count	R	32	BCR and ECR Counter Register
This register keeps track of all ECR and BCR commands triggers that are sent to the Formatter. It is cleared when the Formatter is reset. Bits [31:16] : BCR Count, Bits [15:0] : ECR Count			
FMT_MB_DIAG_REN	RW	8	Mode bits diagnostics
Mode Bits Diagnostic REN: This bit enables the readout of Mode Bits to check that they are being sent to the Formatters correctly as board diagnostics.			
FMT_LINK_OCC_COUNT (x16)	R	12	FIFO occupancy counter per link
The value in the Data FIFO Occupancy Count Register indicates the number of data words (headers/hits/trailers) that are currently stored for a specific link. The counter increments by 1 for each word stored and decrements by one for each word transmitted on the ROD data path. The number of words in the link FIFO is not equal to the number of hits received from the FE modules.			

Error status registers			
FMT_TIMEOUT_ERROR	R	32	Timeout error register
Bit to indicate that the link has received the token but no data has arrived from the module within the time limit set by the Token to Readout Timeout Limit Register. When a time out occurs on a link, the status is latched into this register. This register is cleared when it is read by the host.			
FMT_DATA_OVERFLOW_ERROR	R	32	Data overflow error
Bit to indicate that too much data has gone into the FIFO for one event. When a data overflow error occurs on a link, the status is latched into this register. This register is cleared when the host reads it.			
FMT_HEADER_TRAILER_ERR	R	32	Header trailer error
Status of the Header/Trailer Limit bit for this link. Indicates that the link is almost full and the Link Formatter is only writing Header/Trailer words to the Link FIFO. When a header trailer (FIFO almost full) error occurs on a link, the status is latched into this register. This register is cleared when the host reads it.			
FMT_ROD_BUSY_ERR	R	32	ROD busy error
Status of the ROD Busy Limit bit for this link. Indicates that the FIFO for this is full and data is still arriving from the FE Module. This is a fatal error. Resetting the Formatter is the only way to recover from a ROD Busy Limit Error. This register is cleared when the host reads it.			
Diagnostics status registers			
FMT_VERSION	R	32	Formatter version

EFB

The EFB block:

- deals with 2 Formatter blocks (each of which captures 4 FE links). Two EFBs per Slave chip and 2 Slave Chips per ROD. Totals to 32 FE links.
- supports 8 FE links
- generates 4-bit link address for each output_data word
  - "one-hot" link decoding from "form_enabled" 4b to 2b
  - adds 1 bit gatherer_id
  - adds 1 bit efb_engine_id

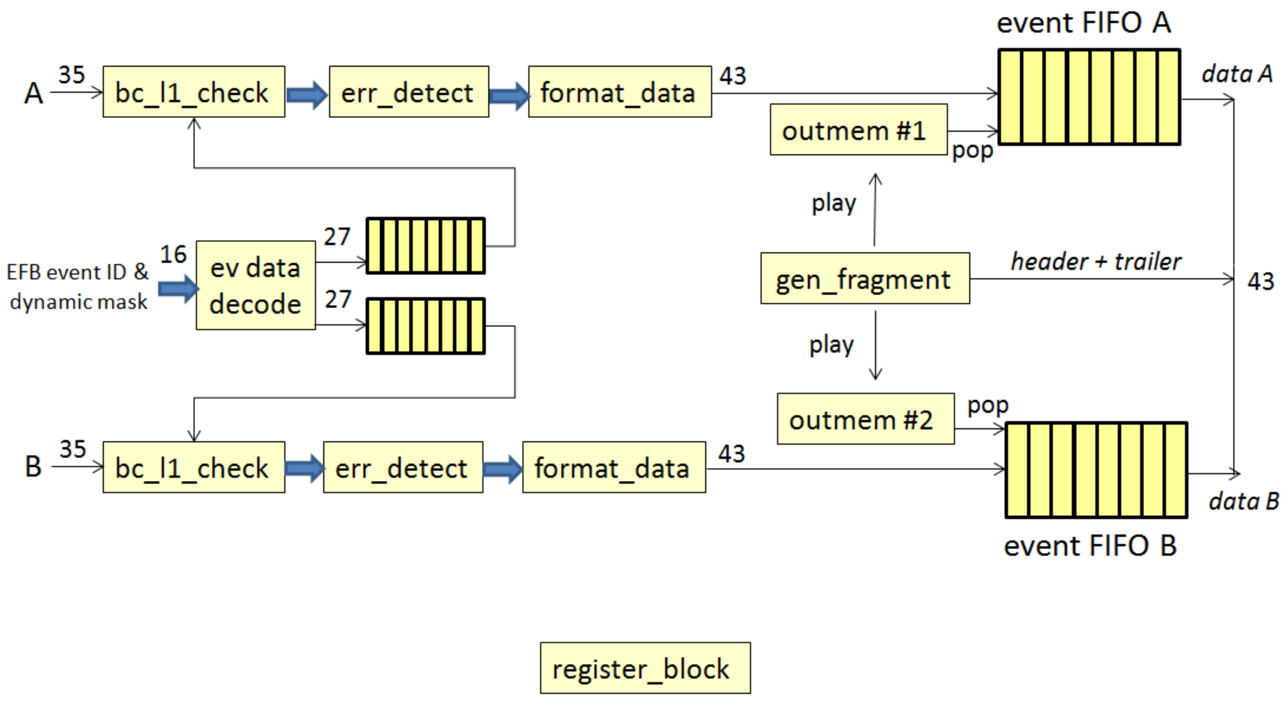


Figure 7: EFB schematic diagram

EFB input data format:

INPUT DATA FORMAT: data_in[34:0] (from Formatter that muxes 4 FE links)
data_in[31:0] event data (4 types of words)
- header => [001| p | xx | link#(2) | 0000 | skip (4) | flag | L1ID(5) | BCID(10)]
- hit => [100 | xxx | link#(2) | ToT(8) | col(7) | row(9)]
- error_flags => [0001 | link#(2) | xxx | SR code(6) | xxxxxxx | SR count(10)]
- trailer => [010 | xxx | link#(2) | errors found(24)]
data_in[32] timeout error (set in Formatter)

```
data_in[33]    condensed mode        (set in Formatter)
data_in[34]    link masked by PPC    (set in Formatter)
```

EFB output data format:

OUTPUT DATA FORMAT: data_out[42:0] (to the Router)

```
data_out[31:0] detector data
    data_out[35:32] link address
      - [34:32] link# (0-3, 4-7)    -- encoded from incoming data (2b) + gatherer ID (1b)
      - [35]   efb_engine_id (0-1)  -- adds to link address (to be removed ?)
    data_out[36] timeout error
    data_out[37] condensed mode
    data_out[38] L1ID error
    data_out[39] BCID error
    data_out[40] link Masked by PPC
    data_out[41] end of event fragment
    data_out[42] non sequential chip error
```

event_data_decode

This block reads the Event ID data coming from the V5 and writes in output two 27-bit bus ev_id_data1 and ev_id_data2 each one containing:

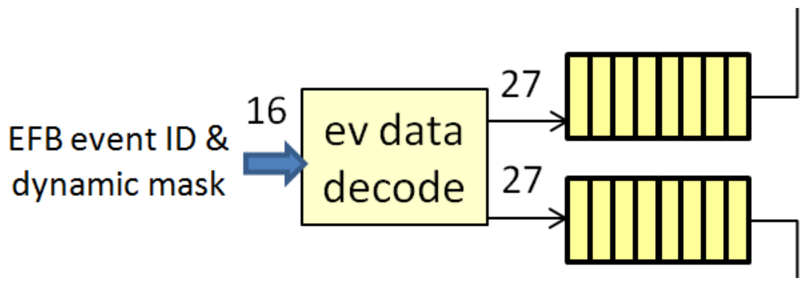
- 10-bits BCID,
- 5-bits L1ID,
- 4-bits ROD trigger type,
- 8-bits dynamic mask (2-bits for 4 links)

Ev_id_data1 and ev_id_data2 flow into the Event ID FIFOs. The full L1, BC ID and trigger type info are sent to the logic that generates the event fragment header/trailer.

Event ID from the V5
word 0 : L1 ID[15:0]
word 1 : ECR ID[7:0] & L1 ID[23:16]
word 2 : BC ID[11:0] & RoL & BOC OK & TIM OK
word 3 : TT - ROD[5:0] & TIM[1:0] & ATLAS[7:0]
word 4 : Dynamic Mask 0 for Links [7: 0]
word 5 : Dynamic Mask 1 for Links [15: 8]
word 6 : Dynamic Mask 2 for Links [23:16]
word 7 : Dynamic Mask 3 for Links [31:24]

EVENT ID FIFO INPUT DATA FORMAT: ev_id_data1[26:0]
ev_fifo_data1[9:0] BCID(10) ev_fifo_data1[14:10] L1ID(5) ev_fifo_data1[18:15] ROD Trigger Type(4) ev_fifo_data1[26:19] Dynamic Mask # (8) supports links 0 to 3 with 2-bits each - L1 Offset Link 0 <= ev_fifo_data (20 downto 19) - L1 Offset Link 1 <= ev_fifo_data (22 downto 21) - L1 Offset Link 2 <= ev_fifo_data (24 downto 23) - L1 Offset Link 3 <= ev_fifo_data (26 downto 25)

EVENT ID FIFO INPUT DATA FORMAT: ev_id_data2[26:0]
ev_fifo_data1[9:0] BCID(10) ev_fifo_data1[14:10] L1ID(5) ev_fifo_data1[18:15] ROD Trigger Type(4) ev_fifo_data2[26:19] Dynamic Mask # (8) supports links 4 to 7 with 2-bits each - L1 Offset Link 4 <= ev_fifo_data (20 downto 19) - L1 Offset Link 5 <= ev_fifo_data (22 downto 21) - L1 Offset Link 6 <= ev_fifo_data (24 downto 23) - L1 Offset Link 7 <= ev_fifo_data (26 downto 25)



bc_l1_check

The main goals of this block are:

- determine event start/end
- get BCID and L1ID event ID
- data from the event FIFO keeps track of an L1 offset value for each link. This offset allows the EFB to compensate for missing or extra triggers on a given link.
- decode link number
- flag L1ID and BCID errors
- check the incoming hits to make sure that the FE links are sending data from the chips in ascending order.counts group event

The following VHDL sub-blocks (processes) are contained into bc_l1_check:

Register_data (kept original)

Takes in *data_valid*, *form_enabled* and *data_in* and puts through 3 shift registers. This process is useful for determining *first_word_in*, *last_word_out*, *next_to_last_word_out*, decode link number and non-sequential chip error flag. Most of the data processing is done at last stage registered output.

Detect_event_start_end (kept original)

Sets *get_event flag* to get more data when almost out of words and sets *processing_event* flag when event FIFO is still read-enabled. This block also checks and flags *event_fifo_empty_error*.

Get_l1_bc_id (modified)

Determines if there are new event ID available and fetches event L1ID, BCID and Group ID. Traverse dynamic mask for all 8 links from *event_id_data* and adjust group offset for L1ID (link-to-link basis). This compensated missing or extra trigger.

Compute_expected_l1 (modified)

Combine the fetched L1ID and link offset for each of the 8 links and compute the expected L1ID value. Expected L1ID <= fetched L1ID + offset +1

Decode_link_number (modified)

For a given 3-b *link_number*, assigns gatherer_id to MSB and use one-hot decoding (of the registered *form_enabled*) to get the remaining 2 bit link address.

Check_id_tags (need to be further modified/re-written)

Assigns *data_out* word with registered *data_in*, link number, EFB ID, timeout error, condensed mode and masked by PPC flag.
While masking headers with timeout errors, checks data_in L1ID against calculated expected L1ID, then flags L1 error in data_out. It does the same thing for BCID, except the expected BCID is already given from event_id_fifo.
Clarification: Is the expected BCID provided as the module input or should this be modified to calculate?
The non-sequential chip error test is performed and flagged here (it is used by the error detect block). *Last_word_out* is assigned to *data_out[41]* to mark as end of event fragment.
We are getting rid of the rod_type in this whole module (and IBL ROD).

Get_group_event_count (yet to be modified)

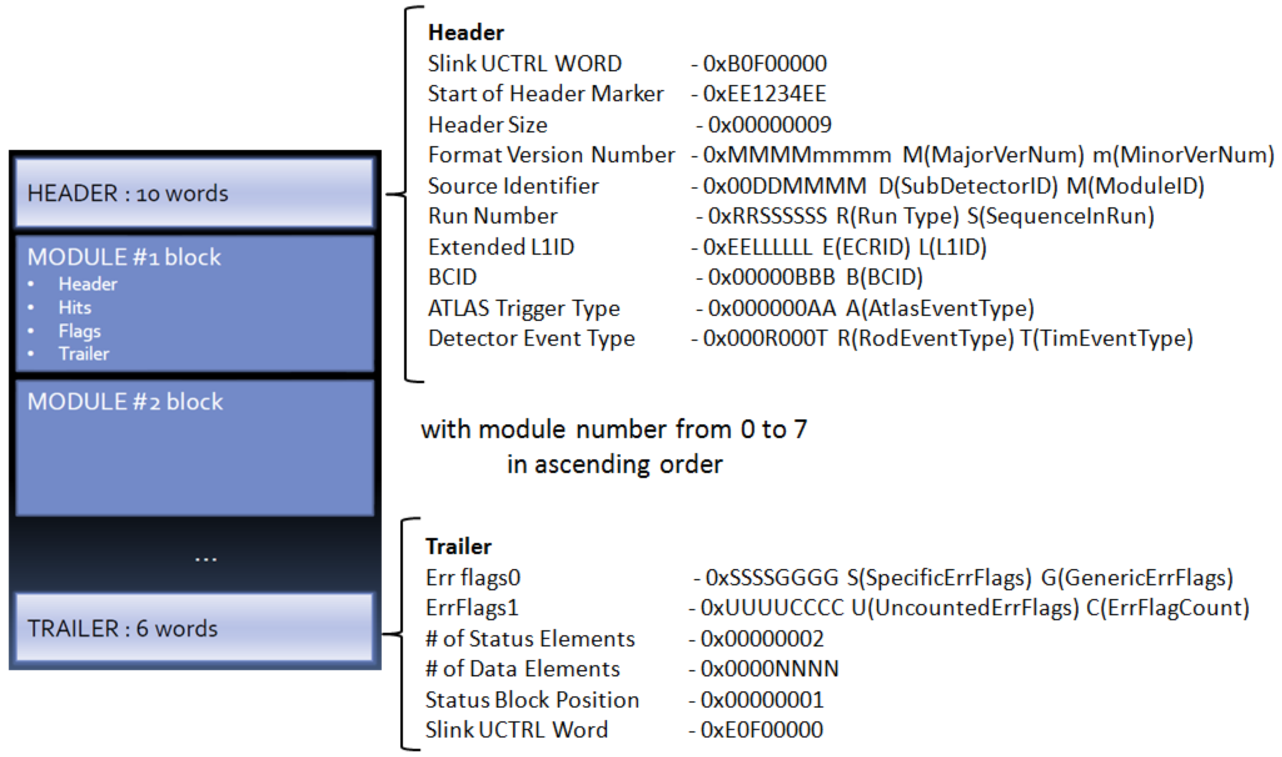
When there is an end of event fragment, calculates the group event count according to different *group_id*.

err_detect
Scans for errors in the data stream, contains maskable error counters. Generates the 2 error flag words which will be transmitted in the trailer.

format_data
Performs final data formatting, removes events with no data and counts how many words are in the output memories for the given event fragment.

outmem
Reads out a block of data from the FIFO to the EFB output.

gen_fragment
Generates the event fragment to be sent in output with header + trailer words.



Header

Slink UCTRL WORD	- 0xB0F00000
Start of Header Marker	- 0xEE1234EE
Header Size	- 0x00000009
Format Version Number	- 0xMMMMmmmm M(MajorVerNum) m(MinorVerNum)
Source Identifier	- 0x00DDMMMM D(SubDetectorID) M(ModuleID)
Run Number	- 0xRRSSSSSS R(Run Type) S(SequenceInRun)
Extended L1ID	- 0xEELLLLLL E(ECRID) L(L1ID)
BCID	- 0x00000BBB B(BCID)
ATLAS Trigger Type	- 0x000000AA A(AtlasEventType)
Detector Event Type	- 0x000R000T R(RodEventType) T(TimEventType)

Trailer

Err flags0	- 0xSSSSGGGG S(SpecificErrFlags) G(GenericErrFlags)
ErrFlags1	- 0xUUUUCCCC U(UncountedErrFlags) C(ErrFlagCount)
# of Status Elements	- 0x00000002
# of Data Elements	- 0x0000NNNN
Status Block Position	- 0x00000001
Slink UCTRL Word	- 0xE0F00000

Name	Bits[31:0]
header	`001xxnnn00000000FLLLLLBBBBBBBBBB`
hit (long)	`100xxnnnTTTTTTTTCCCCCCCRRRRRRRRR`
hit (condensed mode)	`101RRRRTTTTTTTTCCCCCCCRRRRRRRRR`
	`1CCRRRRRRRRTTTTTTTTCCCCCCCRRRR`
	`1TTTCCCCCCCRRRRRRRRRTTTTTTTTCCCC`
	`111TTTTTTTTCCCCCCCRRRRRRRRRTTTTT`
FE flag error	`0001nnnxxSSSSSSxxxxxxxDDDDDDDDDD`
trailer	`010xxnnnEMPplbzhvMMMMMMMMMxxxxx`

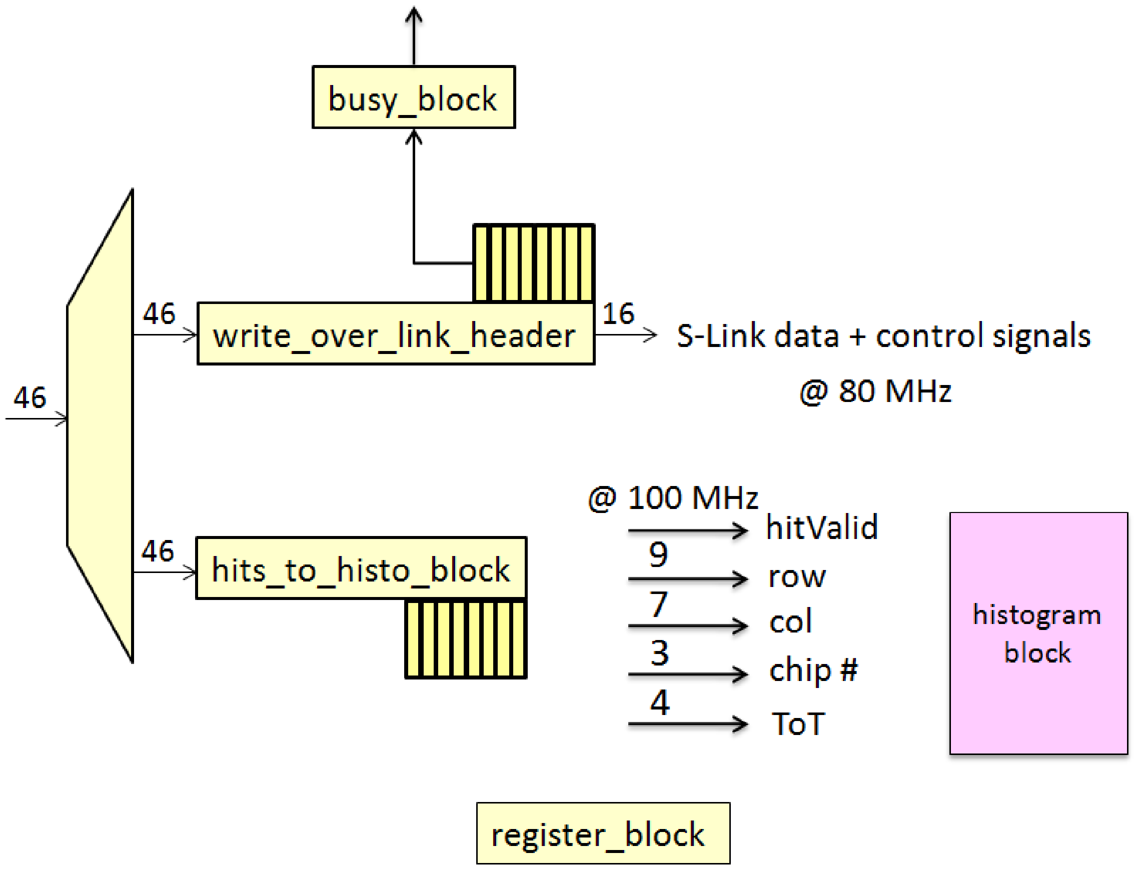
n: link number
F: FeI4B flag bit
L: L1ID
B: BCID
T: hit ToT
C: hit column
R: row column
S: service code
D: service code counter

E: timeout error bit
M: condensed mode
P: link masked by PPC
p: preamble error
l/b: L1ID/BCID error
z: trailer error
h: header/trailer limit error
v: data overflow error
M: skipped trigger counter

EFB: registers

...

Router



hits_to_histo_block

- Strips all header + trailer information and sends hits to the histogrammer clock in the 100 MHz clock domain on separate buses:
    - row (9),
    - col (7),
    - chip number (3)
    - ToT (4)
- Extracts information on adjacent hits and produces two separate words for each double hits.

write_over_link_header

Buffers into a FIFO the incoming data from the EFB and sends the event fragment towards the BOC on a 16-bit 80 MHz data bus with the related control signals.

busy_block

The busy signal is asserted when 7/8 of the capacity of the FIFO is full (when the backpressure is applied from the RoS). This block can also take into account the occupancy of the formatter and EFB FIFOs in order to assert the busy signal when needed.

Router: registers

…

References

[1] https://twiki.cern.ch/twiki/bin/view/Main/AtlasSiliconRodGroup
[2] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.3055&rep=rep1&type=pdf