ATLAS Note

ATLAS-INT-2019-XXX

1st May 2020

Draft version 2.0

New Small Wheel Trigger Processor

Firmware Requirements & Implementation

NSW Trigger Processor Group

Contents

Revision History

Revision	Date	Author(s)	Description
0.0	10-08-2017	L. Levinson	Created
0.1	02-10-2017	L. Levinson	Released to reviewers
0.2	12-06-2018	L. Levinson	Base for June 2018 IDR: corrected Pad Trigger data, Table 5.
0.3	20-04-2019	L. Levinson	Base for April 2019 review. SCX protocol is now defined so that no changes are needed to any OPC UA software. The firmware was moved from an SVN to a Git repository. sTGC input changed from 3 × 40 bits to 4 × 30 bits, Section 2.1.1. Changed format of the Pad Trigger data sent, Table 5. The bitmap ART output format is used; references to list format were removed from Section 2.2.2. Extensively updated L1A readout, Section 7. Added details on the transfer of segments from MM to sTGC, the removal of "duplicate" segments, and fixed latency, to Section 5. Extensively updated Monitoring, Section 6. Correlated monitoring by exchanging data replaced by just signaling to send monitoring data for the current BC via FELIX. Latency table updated. Table 12. FPGA configuration, Section 9 updated. Added Playback, Section 10. Removed Manpower and Phase-2 compatibility sections.
1.0	27-04-2019	L. Levinson	Released to reviewers
1.1	30-04-2019	L. Levinson	Updates to Figures 6, 22. Added link to Excel file listing status and configuration registers. Updated MM Monitoring section. Formatting, English and spelling corrections.
1.2	22-05-2019	L. Levinson	Corrected Table 12, Latency. Still not final.
1.3	10-10-2019	L. Levinson	Added description of BC clock recovery to Section 3.3. Added use of the swROD's round-robin E-links feature to attain needed L1A output bandwidth in Section 7. Replace table for MM readout data format in Section 7.4. Updated Monitoring, Section 6.

2.0	30-04-2020	L. Levinson, *et al.*	For May 2020 PRR: Added a brief description of the carrier card to the Introduction, Section 1, including reference. Updated diagram of carrier card (with latest information from Samway), Figure 5. Added Section 2.2.4, Input data alignment and clock domain crossing. Correct position of Monitor bit in the output to the Sector Logic to be high bit (Table 9). Added comment on bands that span two TDS ASIC's in Section 2.1. Added overview of clocking as Section 2.3. Revise Section 3.3 on fixed latency of the output to the Sector Logic. Added description of the high-level user interface that exports all the needed formats needed to define the configuration and status registers, Section 4. Replaced L1A data format diagram, Figure 27. Added L1A read out bandwidth, Section 7.4.3. Updated Micromegas latency in Section 5. Add text for IPMI monitoring for the new carrier, Section **??**. Improved text for FPGA configuration and IPMI monitoring, Section **??**. A graphical tool is no longer used to create the top level VHDL of the sTGC design (Section 11). Newly instituted Continuous Integration and Continuous Deployment are described in Section 11.

1 Introduction

This document presents the requirements for firmware and its implementation for the NSW Trigger Processors. Information about the whole New Small Wheel detector system can be found in the New Small Wheel Technical Design Report [1]. Documentation of the Trigger Processor can be found in the Preliminary Design Review documents [2–4] and the NSW Trigger Processor documents available in the NSW Electronics SharePoint http://www.cern.ch/ATLAS-NSW-ELX. The documentation includes a description of the interfaces to the two NSW trigger detectors (Micromegas, MM, and small Thin Gap Chambers, sTGC), of the interface to the muon end-cap Sector Logic and of the ancillary functionality of the Trigger Processor. This document presents the requirements and the implementation of the firmware for the Trigger Processor for both detector technologies (sTGC and MM), including the trigger algorithms, monitoring and other ancillary functions.

1.1 Overview of the NSW trigger

The main goal of the NSW trigger is to provide additional information to the muon Level-1 trigger (L1) in the endcap region $1.0 < |\eta| < 2.4$, in order to dramatically reduce fake triggers arising from particles that are not high-p_{T} muons originating in the interaction point (IP). A major source of fake triggers is low energy particles, mainly protons, generated in the material located between the Small Wheel and the end-cap middle station, the Big Wheel (BW). As shown in Figure 1, these particles can cross the end-cap trigger chambers at an angle similar to that of real high p_{T} muons. The NSW trigger signal is based on track segments produced online by the small Thin Gap Chambers (sTGC) and Micromegas chambers (MM) comprising the NSW detectors. These candidate track segments are sent to the new Sector Logic that uses them to corroborate trigger candidates from the BW TGC chambers. The Sector Logic sends Level-1 trigger candidates to the ATLAS Muon Central Trigger system.

1.2 System granularity and terminology

The endcap trigger system operates independently on each detector endcap (side A and side C). Several other factors define the granularity of the endcap trigger system. The following recalls the terminology and boundary conditions of the overall endcap trigger system:

- A **New Small Wheel sector** comprises $1/16^{\mathrm{th}}$ sector of a wheel (i.e. one endcap). Each wheel has eight large and eight small sectors. The detector and trigger sectors are coincidental.
- The **Big Wheel trigger detector** (BW-TGC) has 12 detector sectors per wheel. Each sector is further divided into trigger sectors, four trigger sections per detector sector in the region at larger radius (endcap region) and two trigger sectors in the region at smaller radius (forward region). This segmentation results in a total of 48 trigger sectors in the endcap (larger radius) region and 24 trigger sectors in the forward region.
- A **Sector Logic board** serves two adjacent BW-TGC trigger sectors. For each side (A/C), there are 24 Sector Logic boards for the BW-TGC endcap region and 12 Sector Logic boards for the BW-TGC forward region.
- A Sector Logic board receives data from up to six NSW sectors.

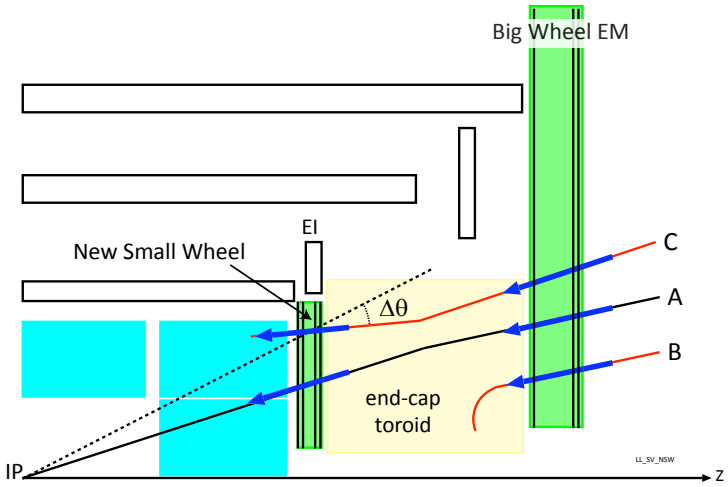


Figure 1: Schematic of the muon endcap trigger. The existing Big Wheel trigger accepts all three tracks shown. With the addition of the NSW to the muon end-cap trigger, only track 'A', the desired track, which is confirmed by both the Big Wheel and the NSW, will be accepted. Track 'B' will be rejected because the NSW does not find a track coming from the interaction that matches the Big Wheel candidate. Track 'C' will be rejected because the NSW track does not point to the interaction point (IP). The NSW logic restricts $\Delta\theta$ to a value consistent with the track to have originated from the IP.

- A **NSW Trigger Processor ATCA carrier board** (corresponding to two NSW sectors) holds two mezzanine cards to serve a NSW octant.
- In the NSW Trigger Processor, there is one **FPGA** per detector technology (MM or sTGC) per NSW sector. (MM) and sTGC information is processed by separate algorithms in separate FPGA's. The two FPGA's are located on the same mezzanine card. Low latency LVDS lines allow communication between the MM and sTGC FPGA's. The track segments found by each algorithm must be merged, i.e. duplicates removed and sent as a single stream to the Sector Logic. The MM algorithm finishes first, so it sends its results to the sTGC FPGA for merging.
- One NSW trigger sector needs to deliver data to up to seven Sector Logic boards. The maximum fan-out of seven is needed for large NSW sectors, due to the overlap with the Big Wheel trigger sectors when multiple scattering, misalignments and magnetic field dispersion are taken into account.

1.3 Use Cases

The Use Cases for the Trigger Processor are:

1. Search for MM and sTGC segments on every bunch crossing.
2. Merge segments into one list for the New Sector Logic.
3. On Level-1 Accept, send Trigger Processor inputs and outputs for the relevant Bunch Crossings to the ROD via FELIX.
4. Download simulated input data to the FPGA and perform the trigger algorithm.

5. Configure the algorithm and readout parameters, and the FPGA's themselves, from the DAQ configuration database.
6. Accumulate and send performance statistics to the monitoring system, e.g. segments generated per BC, segments resulting in L1A, R-ϕ, θ distribution of segments, number of valid planes per BC. . .
7. Monitor the efficiency and purity of the Trigger Processor's segment finding and matching with the Big Wheel by forcing a Level-1 Accept at a low rate (e.g. a few 10's of Hz) for bunch crossings in which a segment has been found. This is done by setting an "NSW monitor" bit in the packet sent to the Sector Logic on every bunch crossing.
8. Send sector data from any bunch crossing that has a "potential" segment to a monitoring point. This enables understanding algorithms, input integrity or BC alignment. For sTGC "potential" can mean, for example, a Pad Trigger for the sector. For MM, this corresponds to strip data in a roads.
9. Monitor board and crate voltages and temperatures and send them to DCS. This may done by the IPMI shelf manager.

1.4 Requirements and Limitations of the trigger

The main requirement for the NSW trigger is to provide track segments from the NSW detectors to be matched to track segments from the Big Wheel. The angular resolution on the NSW track segments should be 1 mrad. In the data-taking period immediately after the installation of the NSW, during the Long Shutdown 2 (LS2), the Big Wheel trigger granularity is limited to an angular resolution of 3 mrad or larger. During the Long Shutdown 3 (LS3), new BW trigger electronics and a new MDT Level-1 trigger will be deployed, allowing for 1 mrad angular resolution in the BW.

Matching resolution requirements The NSW measures the radial coordinates in two planes, the azimuthal coordinate, ϕ, and the angle, $\Delta\theta$, of track segments inside the wheel, i.e. before the end-cap toroid. $\Delta\theta$ is the angle of the segment with respect to an 'infinite momentum track', i.e. a line from the IP to the segment's radial position in the NSW. The radial coordinate is measured by high-precision strips in both detectors. For the sTGC, ϕ is determined by the triggering tower of sTGC pads, and for the MM, by small-angle stereo strips. The angle $\Delta\theta$ is to be measured to an accuracy close to 1 mrad. The NSW trigger logic will reject track segments with $\Delta\theta > \pm(7-15)$ mrad (the final, configurable, value will be determined from future studies). The angle $\Delta\theta$ is passed to the Sector Logic, but is not used in the Phase-1 trigger decision. The corroboration with the BW trigger is done by projecting the 'infinite momentum track' through the $R-\phi$ point of the segment in the NSW onto the plane at the nominal Z of the most downstream sTGC detector's wire plane. This matching requires the NSW trigger segments to have:

- ϕ-resolution of 20 mrad
- η-resolution of 0.005

The sTGC Trigger Processor can find up to four segments. This limit arises from the fact that the Router has four output fibers for transferring strip charges from up to four TDS ASICs. In one bunch crossing, at 4.8 Gb/s, a TDS ASIC can transmit data for only one band of strips. In the case where a band spans the strips handled by two TDS ASICs (about 10% of the bands), two fibers are needed to transmit the band data. Simulations for Phase-2 indicate that bunch crossings with more than three active bands will be less than one per mil. The MM is limited to sending up to eight segments. The Sector Logic requires

the removal of duplicates and can accept at most eight segments from the NSW Trigger Processor. The algorithm to remove duplicates from the sTGC and MM is described in Section 3.2.

1.5 Description of the Trigger Processor hardware

Figure 2 shows an overview of the New Small Wheel trigger and readout paths. Figure 3 shows a block diagram of the Trigger Processor mezzanine card where most of the firmware resides. This includes inputs form the sTGC and MM detectors, the Trigger Processor algorithms, and the output to the Sector Logic. Figure 4 shows the input and output connections of the Trigger Processor.

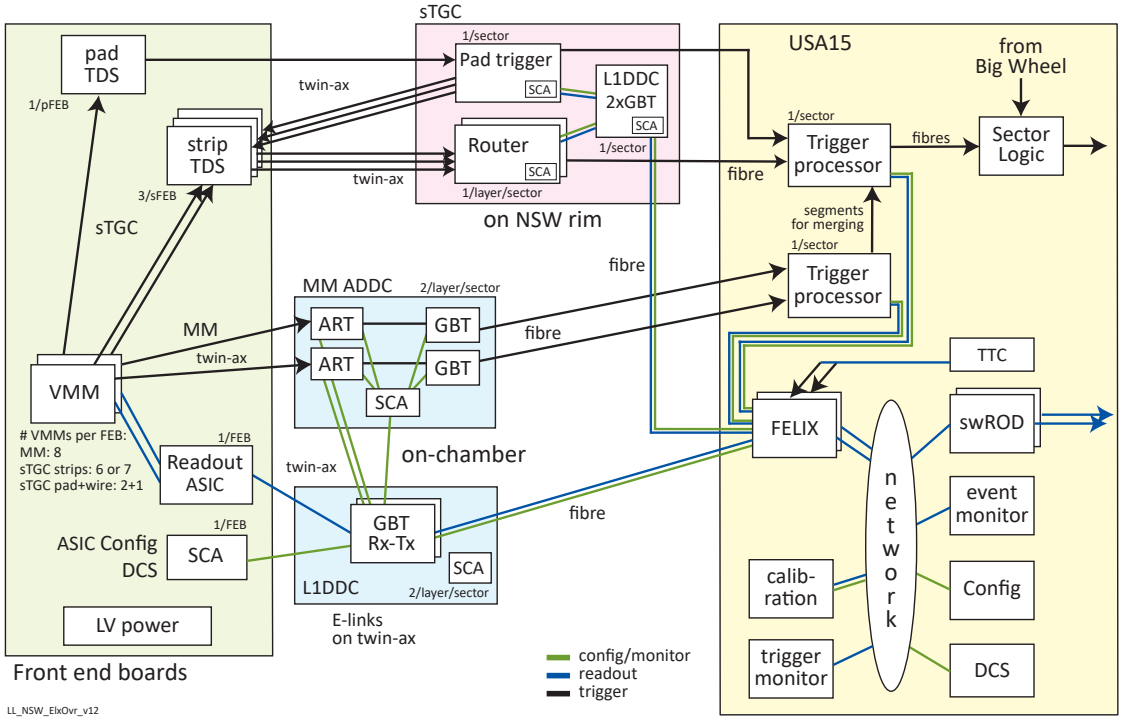


Figure 2: Block diagram of the NSW electronics showing both trigger and readout paths from the VMM front end ASIC. The readout path passes through the L1DDC's to FELIX. The sTGC trigger path passes through the Pad TDS, Pad Trigger, strip TDS and Router to the Trigger Processor. The MM trigger path passes through the ADDC to the Trigger Processor.

The Trigger Processor hardware consists of an AdvancedTCA (ATCA) carrier board and two mezzanine cards. The boards are an evolution of the SRS boards designed within the RD51 collaboration [3, 4]. Each mezzanine has two FPGA's, each with 36 bi-directional fiber optic links (microPODs). The connector to the carrier card is **not** the ATCA standard Advanced Mezzanine Card (AMC) connector. Each mezzanine is intended to serve one sector – one FPGA for MM and one for sTGC. There are 68 fast, low latency LVDS connections between the two FPGA's on a mezzanine. Within one bunch crossing, eight segments can be communicated between the two algorithms. The segments found by the detector that finishes first are sent via these low latency links to the other for merging; up to eight segments can be sent to the Sector Logic. The FPGA's are Xilinx Virtex 7 XC7VX690T-FFG1158 FPGA's with GTH transceivers.

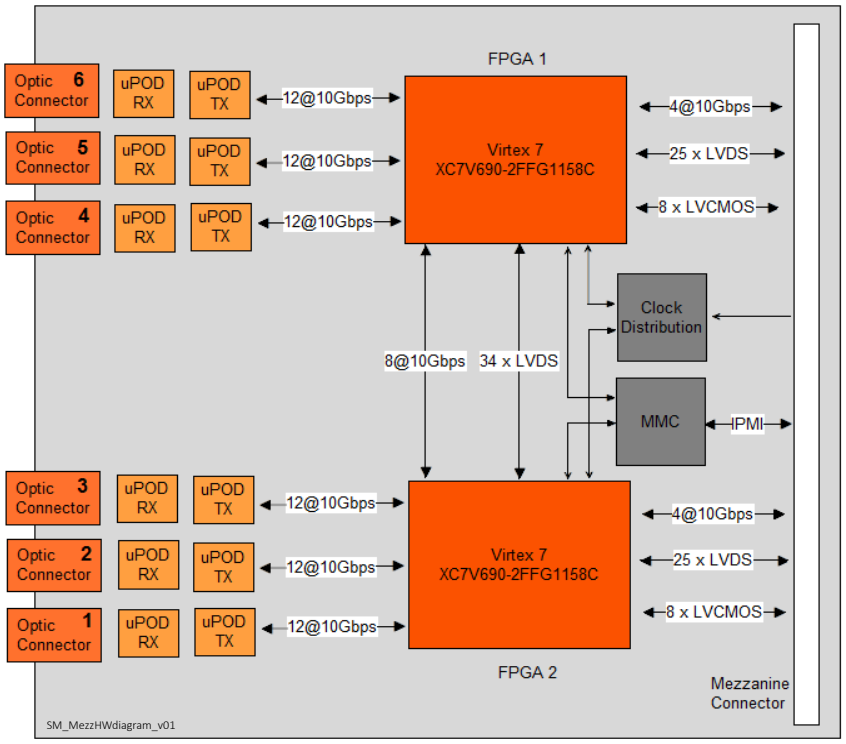


Figure 3: Trigger Processor mezzanine card. Note all LVDS, transceiver and CMOS links are bidirectional and so include twice as many signals as labeled in the diagram.

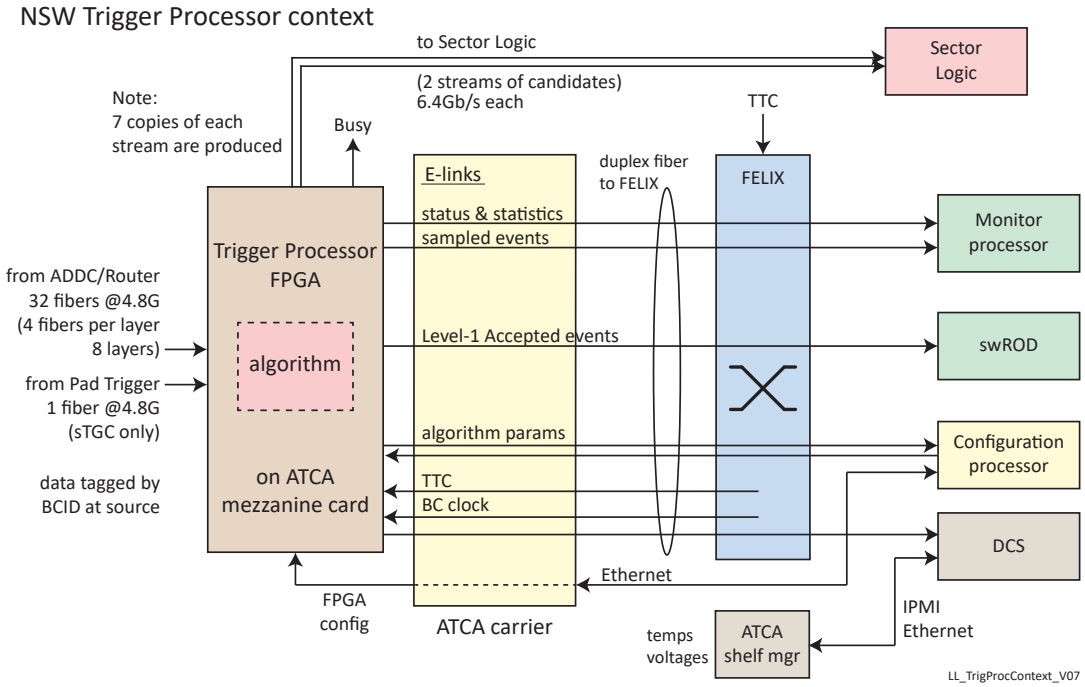


Figure 4: Trigger Processor context diagram

Figure 5 shows a block diagram of the Trigger Processor carrier board [5]. The FPGA's are Xilinx Kintex Ultrascale XCKU060, the board manager is a Zynq XC7Z015 and the gigabit Ethernet switch used has external connections to the ATCA Base interface and the Rear Transition Module (RTM), as well as internal connections to the IPMC, Zynq and FPGA's. The FPGA's on the carrier board support the trigger algorithms running on the mezzanine FPGA's. It can distribute either a Bunch Crossing (BC) clock (40.079 MHz) recovered from FELIX or a local oscillator, and monitor and configure the trigger algorithm. In addition, data from the mezzanine FPGA's sent to FELIX pass through the carrier FPGA's (Level-1 Accept and monitoring).

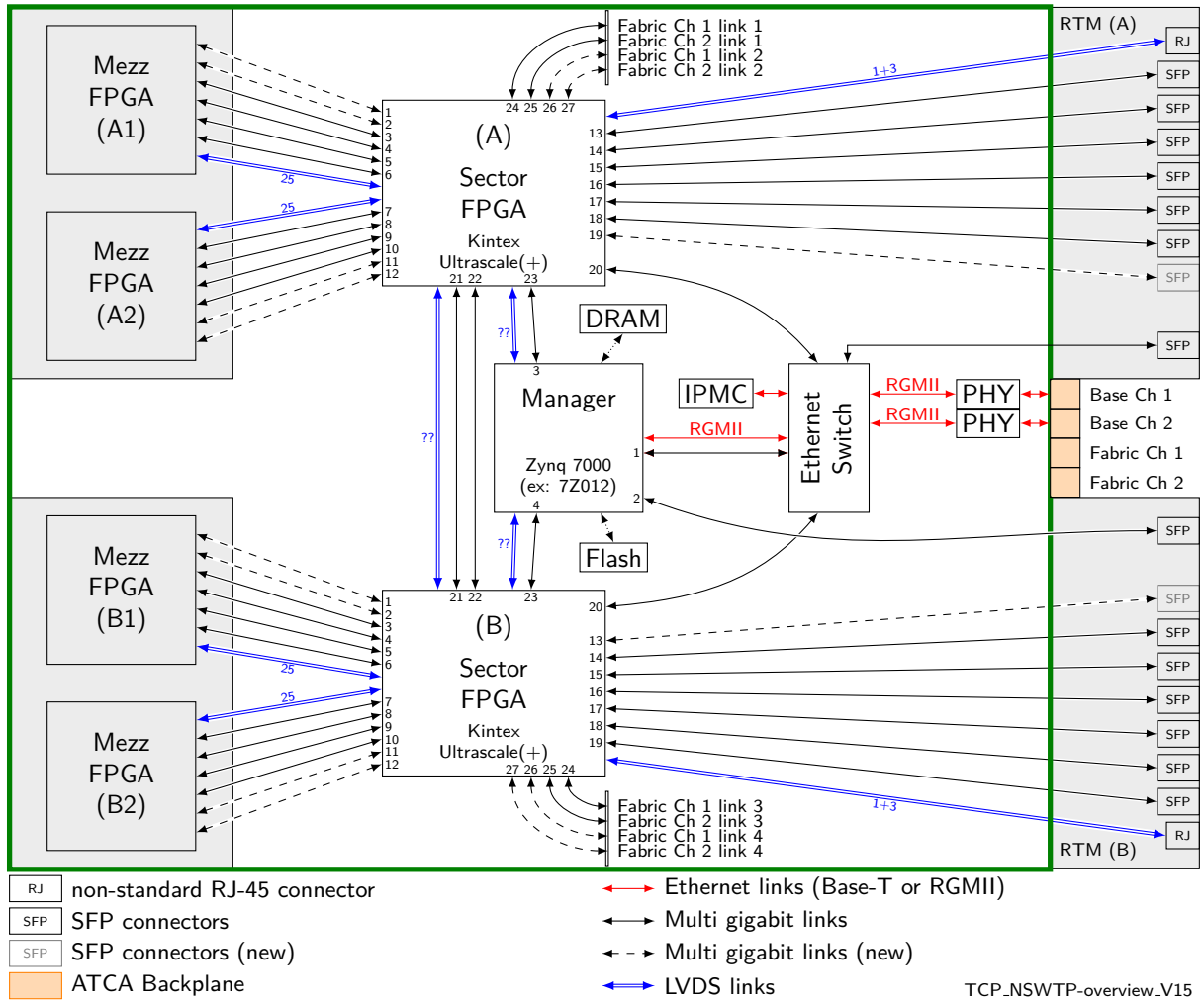


Figure 5: Trigger Processor carrier card highlighting the interconnections.

2 Algorithms

The output of the Trigger Processors to the Sector Logic is up to eight unique segments that point to Big Wheel (BW) Regions-of-Interest (RoIs). Table 2 shows the η boundaries of the RoIs. The ϕ RoIs are 15°/4 in size in the Endcap ($|\eta| < 1.92$) and 7.5°/4 in the Forward part ($|\eta| > 1.92$). The list of segments must be transmitted every bunch crossing. To relax the precision requirements for construction and installation of the NSW, first order alignment corrections must be done by the Trigger Processor. Since R and ϕ RoI indices and not R-ϕ coordinates in the NSW coordinate system are sent to the Sector Logic, the final look-up table from coordinates to indices should take into account misalignments of the Big Wheel.

Table 2: η boundaries of the Big Wheel RoIs

Forward		**Endcap**			
η	**tan**	η	**tan**	η	**tan**
2.4148	0.18022	1.9172	0.30053	1.4622	0.48973
2.3777	0.18714	1.8932	0.30817	1.4428	0.50049
2.3439	0.19370	1.8679	0.31643	1.4227	0.51187
2.3100	0.20050	1.8420	0.32518	1.4015	0.52424
2.2755	0.20768	1.8153	0.33443	1.3783	0.53822
2.2417	0.21497	1.7882	0.34417	1.3528	0.55405
2.2070	0.22277	1.7608	0.35427	1.3257	0.57157
2.1734	0.23056	1.7328	0.36500	1.2972	0.59068
2.1411	0.23835	1.7050	0.37597	1.2681	0.61114
2.1088	0.24639	1.6781	0.38694	1.2384	0.63284
2.0781	0.25431	1.6524	0.39779	1.2086	0.65570
2.0476	0.26247	1.6274	0.40864	1.1798	0.67883
2.0166	0.27101	1.6038	0.41923	1.1528	0.70146
1.9855	0.27991	1.5810	0.42974	1.1275	0.72361
1.9537	0.28931	1.5591	0.44013	1.1036	0.74534
1.9214	0.29921	1.5385	0.45015	1.0807	0.76704
1.9163	0.30082	1.5187	0.46005	1.0587	0.78873
		1.4997	0.46982	1.0375	0.81043
		1.4811	0.47959	1.0334	0.81474

2.1 sTGC algorithm

The sTGC algorithm is based on computing the radial position and segment direction from centroids of strips in two groups of four detector layers. Section 3.2 of the Preliminary Design Review [2] details the sTGC trigger algorithm, including that for alignment corrections. Figure 6 shows a block diagram of the sTGC Trigger Processor algorithm firmware.

The requirements for the sTGC algorithm are:

- Up to four bands of strips in each of eight layers must be capable of being processed to find up to four segments. This limit arises from the four fibers from each Router to the Trigger Processor, one band of strips per fiber per bunch crossing.

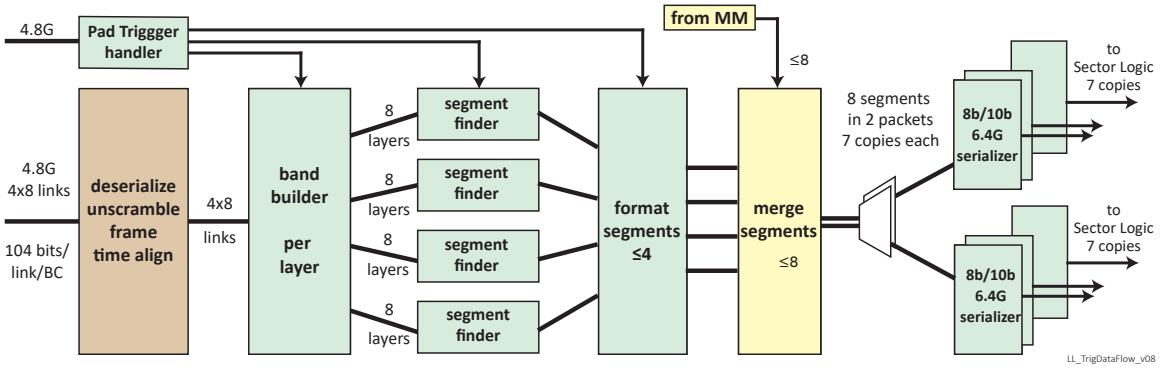


Figure 6: Block diagram of the sTGC trigger algorithm firmware. The data is captured and frame aligned with a 240 MHz clock from the deserializer. The distributor, segment finders and downstream blocks run with a 320 MHz clock. FIFO's between the clock domains must be avoided because of their long latency.

- The ϕ resolution is determined by the pad geometry. See Table 3. A ϕ-id is passed from the Pad Trigger to the Trigger Processor, unchanged, via the strip TDS and Router as well as directly from the Pad Trigger.
- Cuts and thresholds are required to reject δ-rays (a wide cluster) and neutrons (very large charges per strip)
- Due to dead areas (frames) some tracks can pass through at most four active detector regions. A 4-layer algorithm, with reduced resolution, must be provided.
- First order alignment corrections must be made to reduce the needed precision in construction and mounting the sTGC wedges.

The requirements for a valid centroid for a cluster are:

- From two up to seven strips can form a cluster. (configurable)
- The strips must be adjacent.
- One hole (a strip below threshold) will be allowed. (Not allowed in the current algorithm)
- The strips of a cluster must be within the data sent by the TDS in one BC. (The TDS can include strips in a configurable window up to 25 nsec beyond the current BC.)
- Some (≈10%) bands of strips span the boundary between two TDS ASIC's. In this case, the two parts of a band are sent independently from the two TDS ASIC's through the Router and then via two fibers to the Trigger Processor where they must be concatenated.

Table 3: ϕ segmentation of the sTGC

Small Sector						
			One pad		One tower	
	η max	η min	Width	Nmax	Width	Nmax
Quadruplet 1	2.414	1.917	5°	4	1.25°[1]	16
Transition	1.917	1.853	[2]		[3]	
Quadruplets 2&3	1.853	1.281	7.5°	3	1.875°[1]	12
Large Sector						
			One pad		One tower	
	η max	η min	Width	Nmax	Width	Nmax
Quadruplet 1	2.452	1.961	5°	7	1.25°[1]	28
Transition	1.961	1.900	[2]		[3]	
Quadruplets 2&3	1.900	1.298	7.5°	5	1.875°[1]	20

[1] 1.25° = 22 mrad; 1.875° = 33 mrad
[2] Either Quadruplet 1 or Quadruplets 2 & 3
[3] The handling in the Pad Trigger of the transition between Quadruplet 1 and Quadruplets 2 & 3 is not totally defined yet.

2.1.1 sTGC input capture

The trigger input data arrives from the 32 Routers and the Pad Trigger at 4.8 Gb/s, i.e. 120 bits per bunch crossing. Table 4 shows the format of the sTGC input data from the Router. Figure 7 is a block diagram of the sTGC signal capture logic. From the Router, the data arrives either as one 30-bit "null" frame or, for trigger data, as a four 30-bit data frame sequence. 26 bits of each frame are scrambled. 30-bit frame alignment is done by searching for a stable header pattern, either 0b1010 or 0b0110. Should unexpected data be found in the header position, the searching is repeated. Note that the output from the serial transceivers is 20 bits at 240 MHz, whereas the descrambler works with 30-bit words, with output at 160 MHz. Therefore three 20-bit words must be captured to provide two 30-bit words for the descrambler. In fact, a fourth 20-bit word must be captured to allow for the arbitrary bit alignment of the incoming data. If the 30-bit frame is a null, its data is written to output registers. A 30-bit data frame which immediately follows a null frame is assumed to be the first of a 4-frame data sequence. This 120-bit alignment is assumed until another null frame is received. Different links may arrive in different 240 MHz clock periods. They are aligned to a shared 240 MHz clock. The BCID is used to confirm that all 32 inputs belong to the same bunch crossing.

From the Pad Trigger, data arrives from the transceiver, already 8b/10b decoded, as 16-bits at 240 MHz. The K28.5 comma character is used to align the 120-bit data frames. The 16-bit data is written to a width-matching FIFO at 240 MHz. The 96-bit data word is read from the FIFO by the Trigger Processor Distributor with a 320 MHz clock. The data format is shown in Table 5.

Table 4: Format of the sTGC 120-bit input data from the Router. The data payload is 104 bits.

Data	**Frame 0**		**Frame 1**		**Frame 2**		**Frame 3**	
	header	data	header	data	header	data	header	data
bits	4	26	4	26	4	26	4	26
	1010	xxxx	1010	xxxx	1010	xxxx	1010	xxxx

Null	**Frame 0**		
	header	ID	undef
bits	4	10	16
	0110		xxxx

data	Band-ID	Hi/Lo	Strip charges	ϕ-ID	BCID
bits	8	1	14x 6= 84	5	6

ID	Fibre	Sector	Layer	Endcap
bits	2	4	3	1

Table 5: The data record sent every bunch crossing by the Pad Trigger to the Trigger Processor. Each Band-id is 8 bits; Each phi-id, 6 bits. Missing bands have band-id =0xAA. The high flag bit indicates a "monitor this BC" request to be passed to the CTP via the Sector Logic. The BCID is 12 bits. The 16-bit coincidence field shows, for each band for each wedge, the of number of layers in coincidence: 4 × 2 × 2-bits. The 12 bytes are 8b/10b encoded; the first is a K28.5 comma symbol.

–	0–1	2–5	6–8	9–10
comma	flags | BCID	band-id[3..0]	ϕ-id[3..0]	coincidence

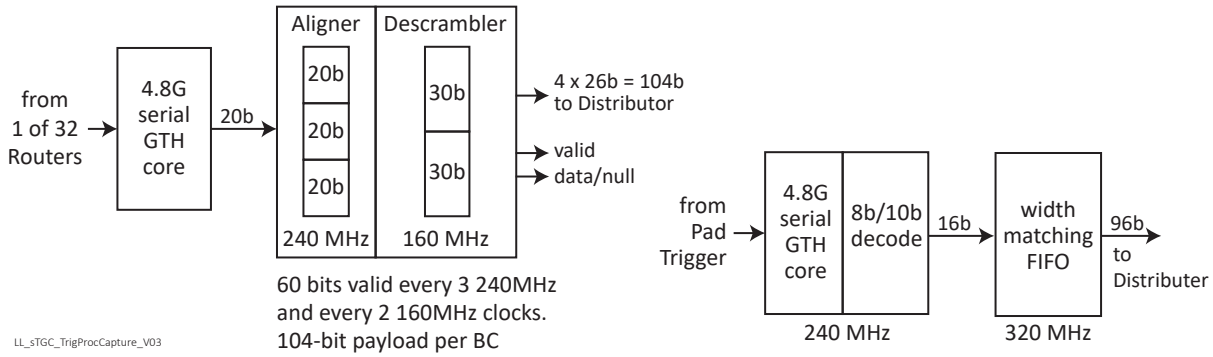


Figure 7: Block diagram of the sTGC signal capture logic. The clocks shown are derived from the 240 MHz deserializer RX clock via a MMCM.

2.1.2 Distributor

The Distributor of the Trigger Processor must pre-process the information sent by the Routers to the Trigger Processor to assemble a message per strip-band (containing the information from all detector layers), and route the assembled messages to the centroid finders. The assembled messages must contain: the strip-band ID, the high/low indicator for the group of strips, the strip data, the ϕ-ID, radial offset of the starting point of the band, and the BCID. The Distributor must arrange strip information from the detector layers in increasing Z, according to a LUT mapping the serializer channel to Z position. The Router channel ID, which identifies the detector layer, is obtained from the ID in the NULL frames sent by the Router when it has no active data, e.g. before the Pad Trigger is enabled.

Table 6: Format of the Distributor output, for one layer, to the centroid finders. A band is actually 17 strips; the Hi/Lo bit indicates if the 14 6-bit charges transmitted are for the upper or lower 14 strips in the band. The centroid algorithm must take the implied offset into account.

Field	Strip data (layers 1,...,8)	band-ID	Hi/Lo	ϕ-ID	Data valid
bits	$(6 \times 14 \times 8)$	8	1	5	1

The strip information is delivered to the Trigger Processor through the 32 fibers connecting the eight Routers (one per detector layer) to the Trigger Processor. In each of these fibres, the Router sends information from a single TDS ASIC. A complication arises from the fact that strip bands may span two TDS ASICs. In this case, each TDS ASIC transmits the strip data that it has, and the Router then receives two active inputs for a single band. For simplicity, and to reduce latency, the Router does not merge the two streams to a single output fiber, but just passes them on two fibers to the Trigger Processor. In this case, the most significant bit within the 8-bit band-ID information is to be used to indicate which of part of the strip band is transmitted from each of the two TDS ASICs. The Distributor then must route the appropriate strip charges from the two strip records to their correct location in the 14-strip record for the detector layer that corresponds to these TDS ASICs. To make this routing possible, a LUT describing the correspondence between band-IDs and the correct location within the 14-strip record is to be provided per detector layer. The merging is implemented using a barrel shifter (6-bit x 14) to route the strips to their correct location. The barrel shifter is controlled by a 4-bit value extracted from a LUT addressed by the 8-bit band-ID. Following the barrel shifter, a 1-to-2 de-mux (84-bit) is used to route the strip data to the complementary strip info if merging is necessary. The de-mux is controlled based on the Pad Trigger data as explained below. The de-mux is followed by a 84-bit x 2 OR that merges the hi and lo strip in the band. Figure 8 illustrates this merging.

Because it does not pass via the TDS ASICs and Router, the information sent by the Pad Trigger arrives ~5 BCs before the TDS data. It is used for two purposes. First, it is used to configure the routing, so complementary TDS messages (belonging to the same band and layer) are concatenated. Second, it is used to configure the routing of the strip information so that for a given band, all its layers are sent to the same sTGC algorithm instance. The Pad Trigger data consists of four flag bits, the BCID, and for each of four bands, the band-id, ϕ-id, and for each wedge of the band, the number of active layers. The 11 bytes of data are shown in Table 5.

2.1.3 FPGA resources

The current mezzanine FPGA design includes the blocks shown in Figure 6. It excludes the calculation of the alignment corrections, but includes the I2C block for configuring the external jitter cleaners and a

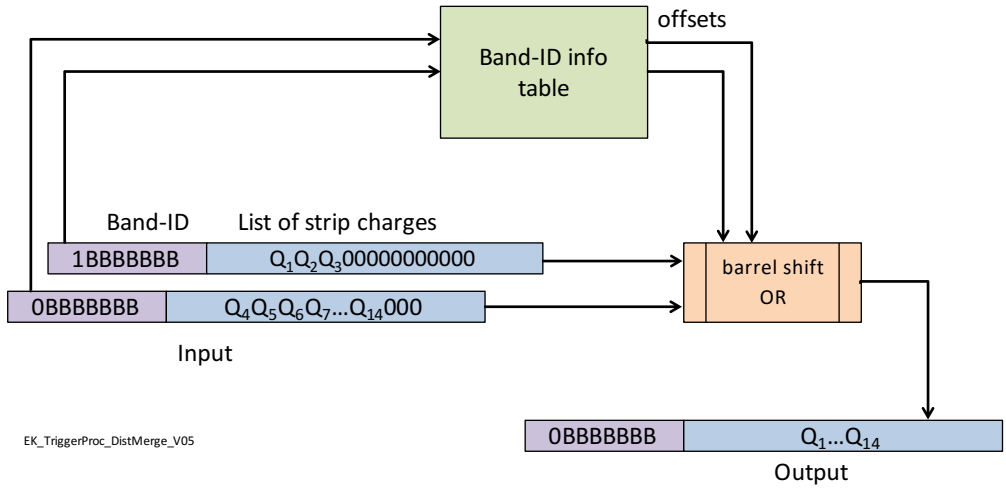


Figure 8: Merging of strip information from a strip band spanning two TDS ASICs – strips 1 to 3 in one, 4 to 14 in the second. There is a separate look-up table for each layer.

sniffer for debugging that was used to output UDP packets to Ethernet via the carrier. The FPGA resource consumption is as follows:

LUT's: 20%; Block RAM: 42%; DSP blocks: 7%, MMCMs (clock blocks): 20%. To be updated

2.2 Micromegas algorithm

This section describes the Micromegas Trigger Processor (MMTP) introduced in [6] and its implementation, as well as necessary modifications to correct for detector misalignments. It also includes details of simulation, and testing on hardware. Further details of the algorithm performance and alignment corrections can be found in [7].

2.2.1 Description

The MMTP algorithm has been described in detail in [6]. Here, its main features are summarized. The main components of the algorithm are shown in Figure 9. In order to handle the detector hit data throughput, there will be 16 copies of the algorithm operating in parallel. Each algorithm will share information with its neighbor to avoid boundary issues. The clock alignment stage is handled in deserializer block, all logic following this stage is clocked using the same 320 MHz clock. This clock is derived from the 40 MHz bunch crossing clock at the FPGA input with a 0-phase offset to maintain a known phase relationship with other components in the system.

Decoder Incoming strip hit addresses are decoded into global slope values using a multiplication with a constant. A strip's stored slope value is defined as the orthogonal distance between a given strip and the beam line divided by the z location of the relevant detection plane. It is pre-computed taking into account a strip offset and a z position stored for each of the 8 planes and 16 radial segments of each wedge. The slope range is divided into 512 slope-roads which will be used to form a track candidate.

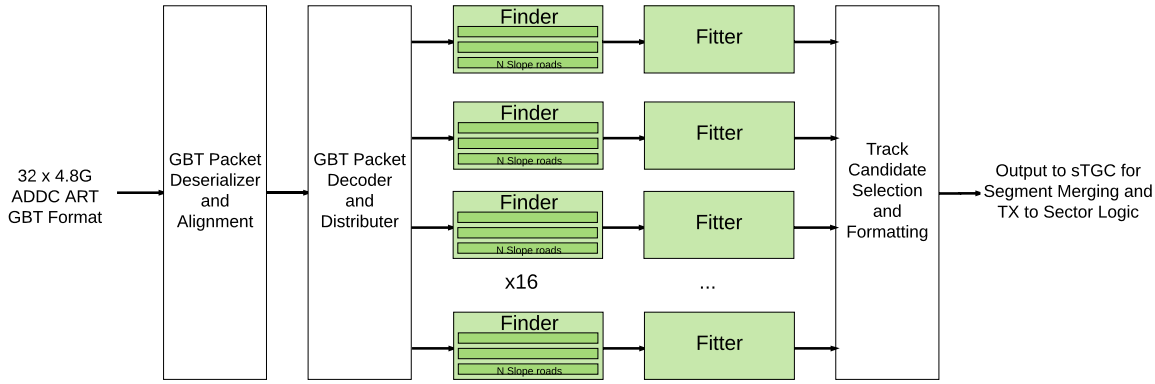


Figure 9: Block diagram of the MM algorithm

Finder The track finding algorithm is slope-road based with the wedge divided from bottom to top into approximately 1000 roads and corresponds to about eight VMM strips. Hit data from the decoder is routed to the corresponding road and that road is marked as being hit. Details of the hit that can't be calculated from the road itself is stored. Each slope-road can hold a single hit for each plane and this hit expires after the hit integration time. The current implementation uses a hit integration time of eight bunch crossings. The slope roads will overlap each other by 1/2 to accommodate tracks on the road boundaries. Each slope-road is checked once per bunch crossing to determine if a coincidence threshold has been met. Coincidence requires a minimum number of planes to be hit and the oldest hit of the track to be expiring. Coincidence identification is accomplished using combinatorial logic and a priority encoder. The strip number and slope for each hit are calculated and passed to the track fitting algorithm.

In an effort to reduce the impact of background hits, the Finder algorithm has been rewritten to separate the X-plane and UV-plane coincidences. The fitter will first identify a coincidence trigger on the X planes and then scan from left to right all of the UV-roads that overlap the triggered X-road. The overlapping area of UV-roads are "diamonds". For each X trigger there can be as many as 57 UV-road diamonds that are searched. The number of diamonds is configured for each algorithm region to match the length of the Micromegas detector strips. Every bunch crossing each of the 16 algorithm regions can process two independent X-road coincidences. For every X-road coincidence the Finder can process three UV-road coincidences.

The Finder implementation focuses on minimizing the resources used in each slope road. This component is multiplied by the number of slope roads and number of planes, thus, uses the most resources in the total design.

Fitter In the fit, individual hit slopes in a slope-road are used to calculate global slopes associated with each plane type, which are averages (e.g. M_X^g for the average slope of horizontal planes). These in turn are used to calculate the three composite slopes: slopes associated with the horizontal (m_x) and vertical coordinates (m_y) and the local slope of hits in the horizontal planes (M_X^l), all of which are shown in Equation 1. In Equation 1, θ_{st} is the stereo angle of 1.5°; the sums are over relevant planes; $\bar{z}$ is the average position in z of the horizontal planes; and y_i and z_i in the local slope expression refer to the y and

z coordinates of hits in X planes.

$$m_x = \frac{1}{2}\cot\theta_{st}\left(M_U^g - M_V^g\right),\ m_y = M_X^g,\ M_X^l = \frac{\bar{z}}{\sum_i z_i^2 - 1/n\left(\sum_i z_i\right)^2}\sum_i y_i\left(\frac{z_i}{\bar{z}} - 1\right) \tag{1}$$

From these composite slopes, the familiar expressions for the fit quantities θ (the zenith), ϕ (the azimuth[1]), and $\Delta\theta$ (the difference in θ between the direction of the segment extrapolated back to the interaction point and its direction when entering the detector region; the following is an approximation) may be calculated, as noted in [6]:

$$\theta = \arctan\left(\sqrt{m_x^2 + m_y^2}\right),\ \phi = \arctan\left(\frac{m_x}{m_y}\right),\ \Delta\theta = \frac{M_X^l - M_X^g}{1 + M_X^l M_X^g} \tag{2}$$

The fitter implementation is accomplished primarily by taking advantage of cascading DSP48E1 slices to minimize resources and latency. The fitter is implemented as a pipeline and can accept one new track from the finder every 320 MHz clock cycle.

2.2.2 MM input format

The ART (Address in Real Time) data from an entire sector will be transmitted to a single Trigger Processor via 32 ADDCs. Each ADDC will transmit its data on a single fibre optic link. The Trigger Processor therefore uses 32 fibres to receive the ART data from one sector. The MM and sTGC Trigger Processor will share the same ATCA carrier card and each carrier will support two sectors.

2.2.3 ART Data Protocol

The ART Data from the ADDC will be transmitted using the GigaBit Transceiver (GBT) architecture and transmission protocol in a low-latency widebus mode at a rate of 4.8 Gb/s. The Trigger Processor will take advantage of the GBT firmware developed by the GBT Project and optimized for latency by Kai Chen, to implement the receivers.

The GBT packet in widebus mode will provide 112 data bits and arrives once every bunch crossing. One ADDC will service 32 VMMs and each packet can contain ART data from a maximum of eight triggered VMMs. Each VMM will be uniquely identified to determine which MM strip on the sector is hit.

The GBT data will be encoded in the “hitmap” format where the triggered VMMs are represented by asserting a bit in a 32-bit hit list as shown in Figure 10. The triggered VMM strip number is provided in an encoded list as shown in Table 7.

[1] Defined with respect to the center (y) axis and *not* the axis of the strips (x) as is sometimes typical, so a hit along the center of the wedge has $\phi = 0$

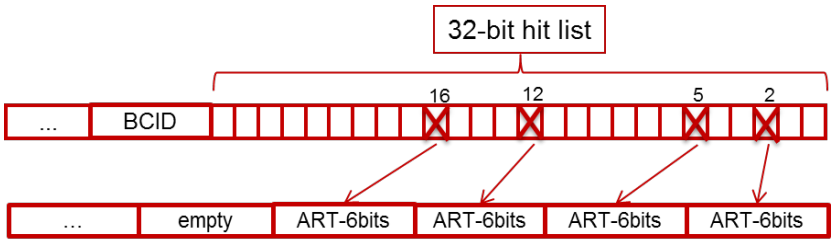


Figure 10: Option 1 VMM ID encoding using 32-bit hit list.

0b1010	BCID(12)	ERR_FLAGS(8)	HIT_LIST(32)	ARTDATA_PARITY(8)	8xART_DATA(6)

Table 7: Option 1 ADDC GBT packet format.

Where:

BCID: 12-bit bunch crossing ID

ERR_FLAGS: ADDC error indicators from the eight binary encoders attached to each priority encoder in the hit selection logic

HIT_LIST: 32-bit list of flags corresponding to each of the 32 VMMs. 0: no hit, 1: hit. A register controls if this is a filtered (i.e. 8 hits max) or an unfiltered copy of the VMM flags registered in a particular BC.

ARTDATA_PARITY: 8-bit parity the ART data computed by each of the 32 ART de-serializer units. Each bit corresponds to one of the ART data field selected by the priority unit

ART_DATA: 6-bit triggered VMM strip number

2.2.4 Input data alignment and clock domain crossing

The ADDC data from 32 fibers arrives via GBT FPGA on a 40 MHz clock that is aligned to each receiver's RX out-clock. In order to align each fiber to a single recovered BC Clock, the data is registered twice using a clock that is phase aligned to the recovered BC clock. The first register uses a 40 MHz clock with settable phase adjustment in 45° increments specific to each fiber. (See Figure 11.) This register will be used to account for any individual fiber length differences. The second register will also be a 40 MHz clock with settable phase adjustment but the phase will be a single constant set for all fibers. This register will align the data from all fibers and provide a fixed latency.

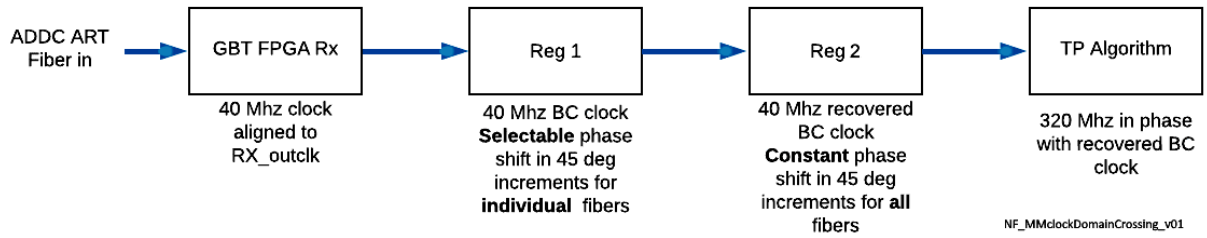


Figure 11: Block diagram of the Micromegas clock domain crossing

2.2.5 FPGA Resources

A full wedge design has been implemented. This includes 16 parallel algorithms, Finder and Fitter, along with the logic to handle tracks on the algorithm boundaries. A block diagram of the algorithm implementation is shown in Figure 9. The implementation uses a nominal configuration based on algorithm performance studies, Ref. [7] with the addition of a new Finder design using fine slope-roads and UV-road diamond triggering. The full algorithm resources required are ~65% of the Xilinx XC7VX690T FPGA. The basic alignment correction functionality involving plane shifts and tilt have been implemented. Alignment corrections involving twist and rotate have not yet been implemented. The implementation for the full wedge includes monitoring, L1A packet generation, and the lateral transfer of track segments to the sTGC FPGA. The removal of duplicate track segments has not yet been implemented.

Vivado 2018.3 is used in non-project mode to enable scripts to control the implementation process. This has greatly aided in timing closure. The implementation requires the use of placement blocks to meet timing.

2.2.6 Simulation

A software model of the algorithm has been developed to evaluate its performance using Athena generated VMM ART hit data. The firmware design is simulated using the same Athena data and the results are compared between the software model and firmware simulation for validation. Early testing of the algorithm used both behavioral and timing simulations. Recent algorithm validation focuses on hardware testing with behavioral simulations as a diagnostic tool. Modelsim is used for firmware simulations.

2.2.7 Hardware Testing

The MMTP design is being tested using two separate hardware platforms, the Xilinx VC707 Development board, and the ATCA HORX hardware. The HORX hardware is being tested with an ADDC emulator to generate the GBT packets containing Athena generated ART data. These packets are transmitted and received with a fiber in a loop-back scheme allowing the MMTP to receive data as if it were sent from an actual ADDC. The MMTP hardware results are compared with the results of the software simulation for verification.

The primary method of algorithm development has been in a cosmic ray test stand. This system includes an entire ART data chain consisting of a Micromegas octuplet, eight MMFE8s, two ADDC prototypes, and one MMTP running on a Xilinx VC707 development board. This system allows us to move beyond simulations and exercise the MMTP using muon tracks from a Micromegas detector. Over the past months, MMTP diagnostic data from millions of cosmic ray muon triggered events has been collected. A full description with test results can be found in [8].

During past integration weeks at CERN the HORX MMTP was successfully integrated with the ADDC V2 in the ART data chain. The ART data from pulsed MMFE8s were successfully received by the MMTP and generating track candidates. All of the boards were synchronized to a clock that was sourced by FELIX. FELIX via RTM was used for all of the communication during tests utilizing SCAX for algorithm configuration, monitoring and L1A functionality to receive algorithm and segment data.

2.2.8 Micromegas trigger alignment constants

The expected tolerance for installation of the Micromegas chambers is roughly 2 mm, although it possible to have larger excursions. In order to retain optimal performance of the trigger, alignment corrections need to be performed. The corrections need to be performed rapidly to minimize latency, and also to have a small number of constants to avoid large look-up tables. Figure 12 shows the categories of possible misalignments.

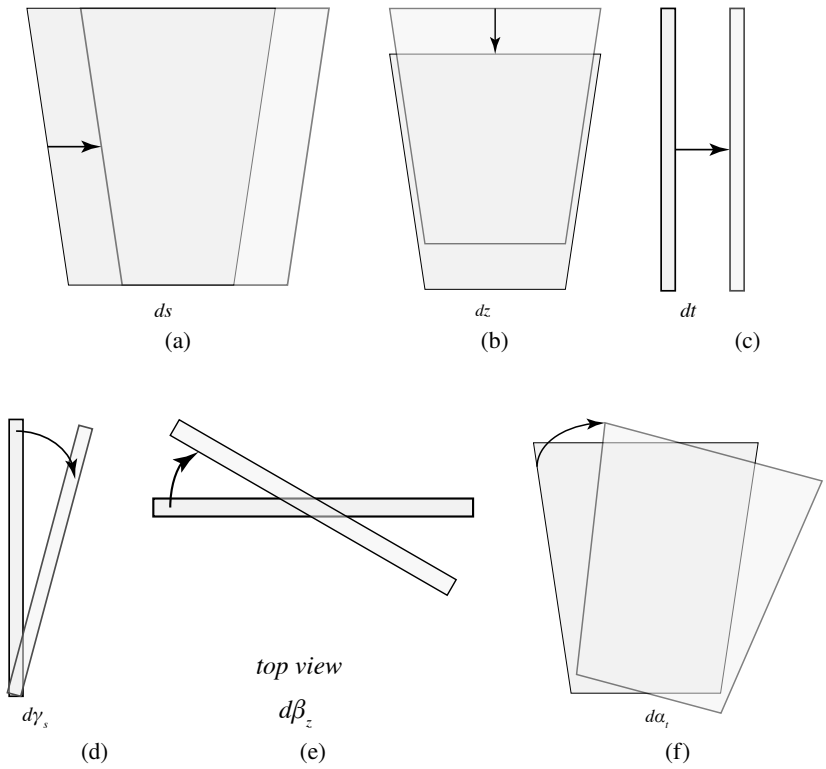


Figure 12: The different misalignment cases as defined in the AMDB manual.

Translations Translations were studied up to 5 mm out of perfect alignment, and showed that the corrections gave acceptable performance.

a Translation along horizontal strip direction (s): This only affects small angle stereo. It is calculated ahead prior to a run, stored in a small look up table that has 11 constants in memory, one lookup, one addition – two operations.

b Translation perpendicular to beam and strip direction (z): This only requires a change in the calculated constant ybase at the start of the run with no additional parameters.

c Translation along beamline (t): As with the translation in z, this requires only a change in an initializing constant at the start of the run with no additional parameters.

Rotations Rotations of modules were studied up to ±1.5 mrad, and it was found that acceptable corrections can be made with a manageable number of parameters.

d Chamber rotations toward and away IP (gamma): A two-dimension lookup table is used to parameterize the wedge into an 8 x 8 grid, and with one offset, this yields 56 parameters (64-8) and one operation.

e Chamber rotations about wedge vertical (beta): This rotation creates a negligible effect for any conceivable misalignment. No correction is performed.

f Chamber rotations in the azimuthal direction (alpha): This rotation can be corrected for with a look-up table that partitions the wedge into discrete subsections. It requires two operations per hit in the x strip (non-stereo strip), and requires 400 constants in memory.

A summary of correction methods, including resources necessary for the individual analytic cases, is shown in Table 8.

Table 8: A summary of corrections with additional constants/operations (written as n_{const}c/n_{ops}op; n_X is the number of X hits in a fit) necessary for analytic corrections. Yes means a correction exists but might not entirely remove misalignment effects, while yes+ means a quality of correction is only limited by knowledge of misalignment and memory.

	Δs	Δz	Δt	γ_s	β_z	α_t
Analytic Resources	yes+ 11c/2op	yes+ 0c/0op	yes+ 0c/0op	yes 56c/1op	no —	yes 400c/2$n_X$op, 32c/12$n_X$op
Simulation	yes+	no	no	no	yes+	yes+

2.3 Clocking

The Trigger Processor has no direct connection to the TTC. The BC clock and TTC bits are received via the FELIX fibre connection. The BC clock is recovered with fixed latency by the fixed latency version of the CERN GBT-FPGA firmware [9, 10] running in the Carrier. Figures 13 and 14 show the details of the fixed latency BC clock recovery from the 240 MHz user bus clock from the 4.8 Gb/s deserializer. The 120-bit frame transferred every BC clock consists of a 4-bit fixed header (Idle=0b0110 or Data=0b0101) and a 116-bit scrambled payload. The fixed latency of the recovered BC clock is attained by shifting its rising edge so that it is aligned to the start of the fixed header pattern.

All the needed design clocks, 40, 80, 160, 240 and 320 MHz, are generated from the recovered bunch crossing clock, each with an edge aligned to the BC clock edge. The deserializer and serializer reference clocks are generated from the BC clock by "zero-delay" jitter cleaners that align their output to the input BC clock. Since the various design clocks preserve fixed phase with respect to the BC clock, clock domains can be crossed without FIFO's. This avoids the latency inherent in FIFO's. Figures 14 and 16 show the Micromegas and sTGC, respectively, clock schemes.

Note that Micromegas Trigger Processor does not need to transmit its segment output to the sTGC Trigger Processor with fixed latency. The Micromegas segment output is passed to the sTGC clock domain where it is buffered in a FIFO until the sTGC algorithm completes.

See also Section 3, Output to the Sector Logic.

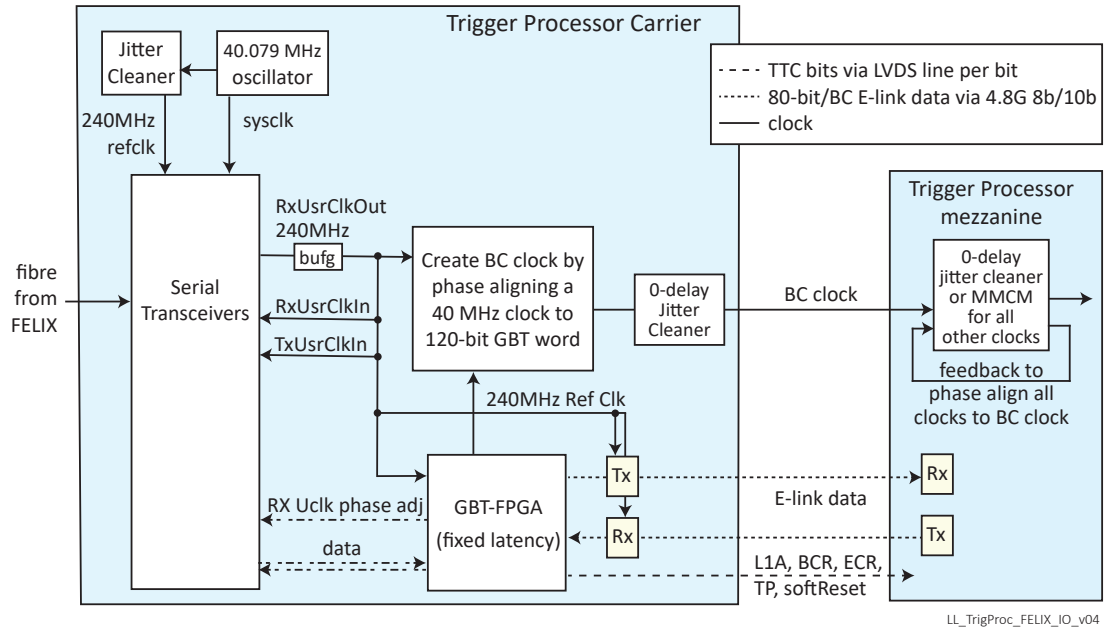


Figure 13: Details of the fixed latency BC clock recovery from FELIX. A 120-bit packet is received every bunch crossing.

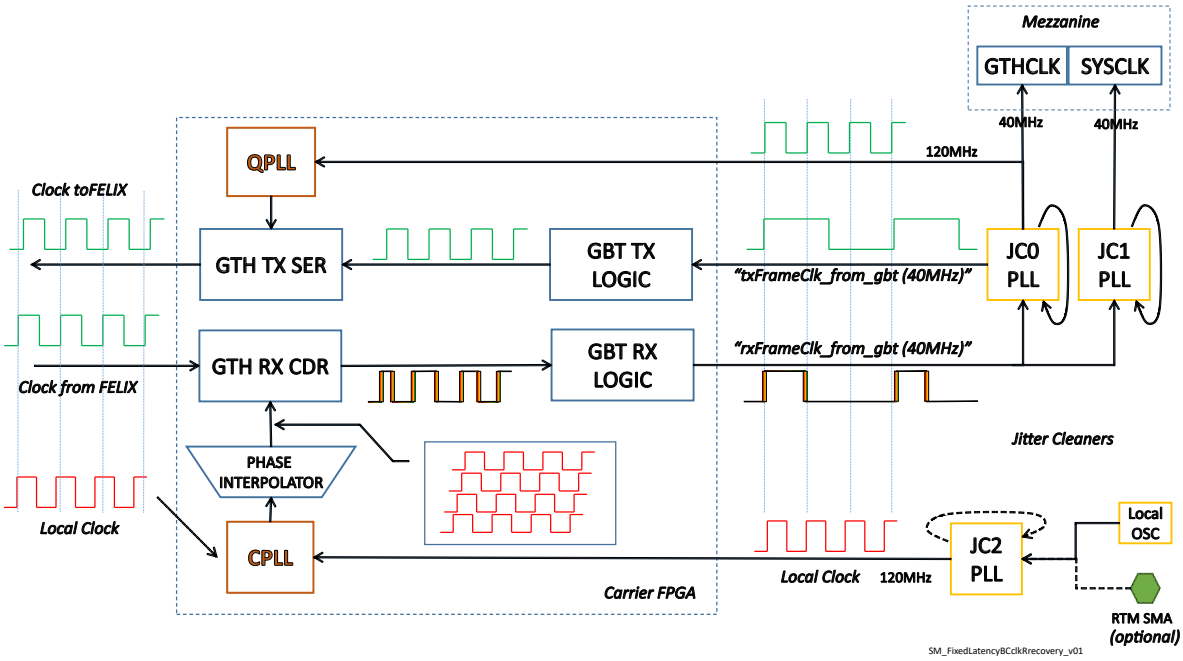


Figure 14: Fixed latency BC clock recovery – details

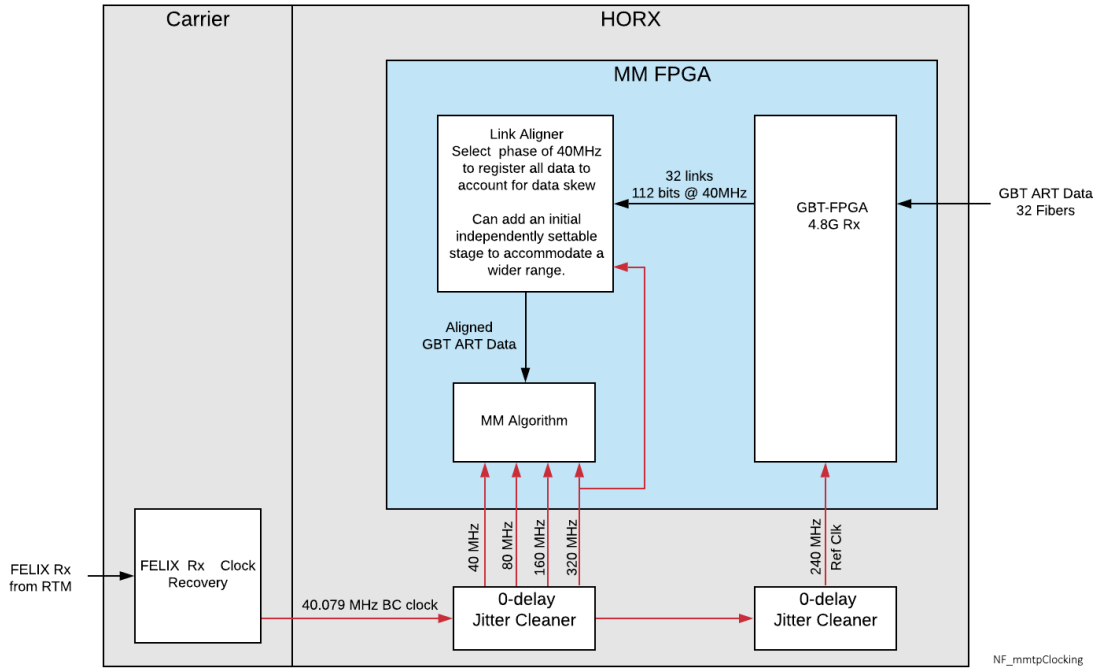


Figure 15: Micromegas trigger clocking

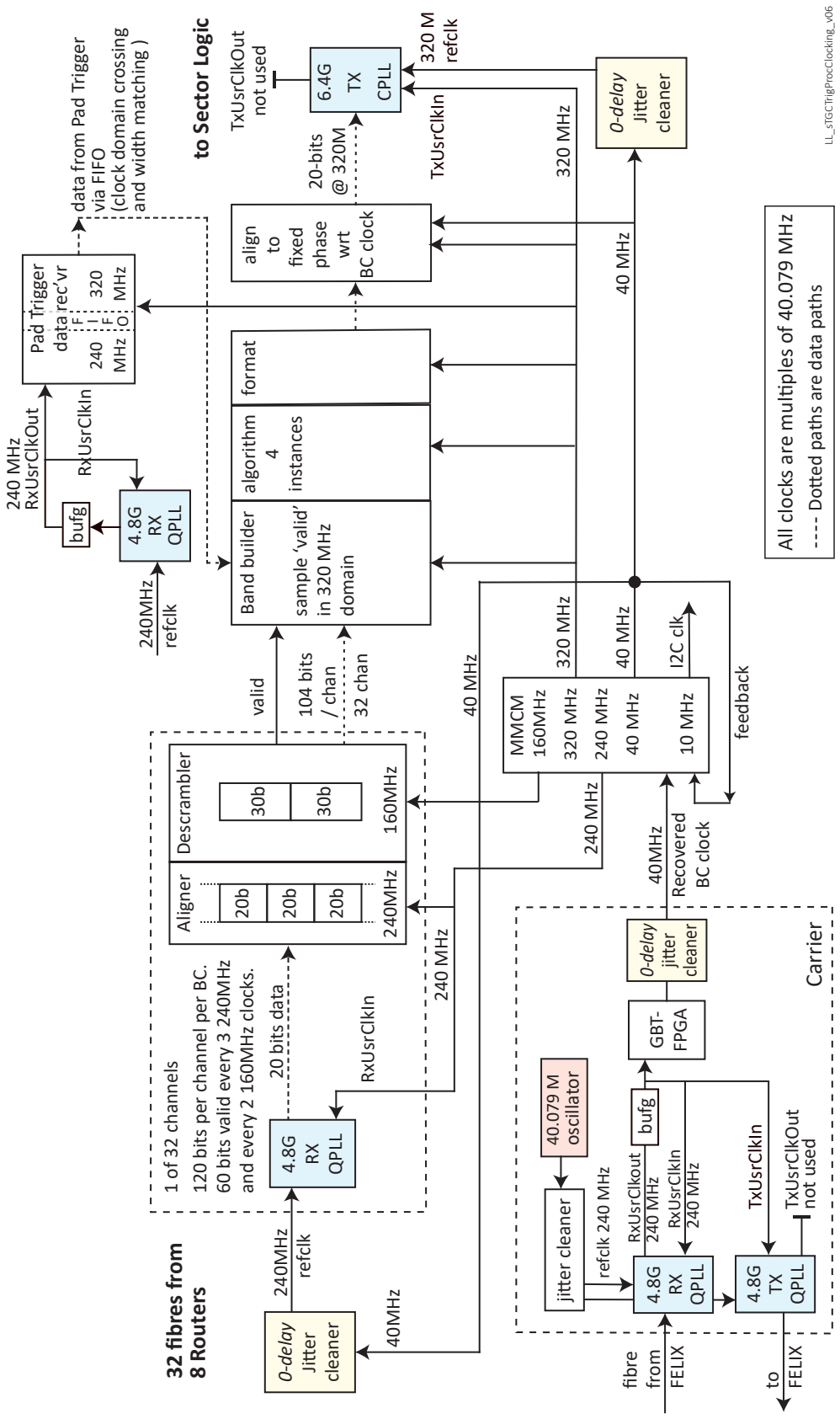


Figure 16: sTGC trigger clocking

3 Output to the Sector Logic

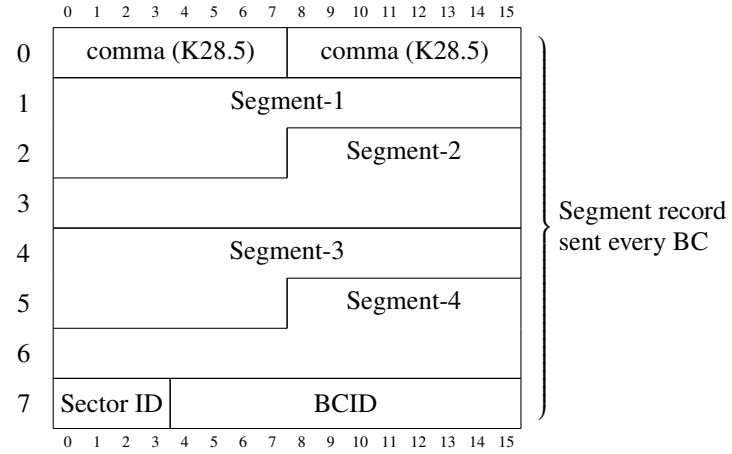
The output data format of a track segment from the NSW Trigger Processor is shown in Table 9. One track segment is represented in a 24-bit field, that includes $\Delta\theta$, ϕ and R position information, and three flags. Required resolutions (1 bit) are approximately 1 mrad (±15 mrad full scale) for $\Delta\theta$, 20 mrad for ϕ and 0.005 in pseudo-rapidity η. $\Delta\theta$ is stored as a 1's-complement signed value, with "-0" (0x1F) indicating that there is no segment present in that segment field. The ϕ resolution flag when set, indicates that the segment is from the sTGC whose ϕ resolution is one bit less than the MM. In this case, the low bit of the ϕ index is zero. The low resolution flag is set when only one quadruplet had a 3-out-of-4 coincidence and therefore $\Delta\theta$ has poor resolution. If data only from bunch crossings that are accepted by the Level-1 trigger are recorded, we cannot know if missing triggers are due to improper functioning of the NSW trigger. The Monitor flag is set when this bunch crossing has been sampled as "interesting" by the NSW Trigger, e.g. a 3-out-of-4 coincidence. When the Monitor flag is set, the Sector Logic forwards it to the MUCTPI and similarly the MUCTPI to the CTP. The CTP generates a Level-1 Accept with trigger type "NSWMON trigger". This allows a monitoring process to see all the NSW front end, Sector Logic and MUCTPI data for that bunch crossing. The CTP ensures that the rate of these events does not exceed a few 10's of Hertz.

Table 9: Format of the 24-bit track segment sent to the Sector Logic ("Monitor" is the *hi-order* bit.)

Field:	Monitor	Spare	Low res.	ϕ resol.	$\Delta\theta$ (mrad)	ϕ index	R index
Num of bits:	1	2	1	1	5	6	8

Up to eight track segments are transmitted to the Sector Logic in each LHC bunch crossing via two optical fibres, four candidates per fiber. Table 10 shows the format of the transmitted data. 8b/10b encoding is used. The segment frame is comprised of two comma codes for frame alignment, up to four track segments (three bytes of data each), the NSW Sector ID (4 bits) and a BCID number (12 bits). A missing segment is indicated by a $\Delta\theta$ of "-0" (0b10000). Note that the End cap is not identified. The 16 bytes, 160 bits after 8b/10b encoding, are sent in a single bunch crossing at 6.4 Gb/s. The segments found by MM and sTGC must be merged into a single list. Duplicates (i.e. equal $\Delta\theta$, ϕ index and R index) must be removed, giving priority to the higher quality segment.

Table 10: Data format from the NSW Trigger Processor to the Sector Logic. The data is transmitted with 8b/10b encoding in one bunch crossing (16 bytes at 6.4 Gb/s). The format of the 24-bit segment is shown in Table 9. The comma character (K28.5) is an 8b/10b out-of-band control code for symbol alignment and packet framing. The NSW Sector ID is the *low-order* 4 bits and the BCID is 12 bits.



3.1 Transfer of track candidate segments from MM FPGA to sTGC

Figure 17, shows a block diagram of how the track candidate segments are sent from the MM FPGA to the sTGC FPGA where they will be merged. The process starts by collecting the segments from the 16 algorithm regions. Each region can generate eight segments every bunch crossing. A priority encoder will be used to select eight total segments each bunch crossing from all regions. The selected segments will be transferred to the sTGC FPGA using 16 640 Mb/s LVDS signals. The receiver in the sTGC uses independently delayed versions of the same signal to set the sampling point to the center of the data eye and provide segment data to the sTGC logic that will be aligned to the sTGC clock with no additional clock domain crossing necessary.

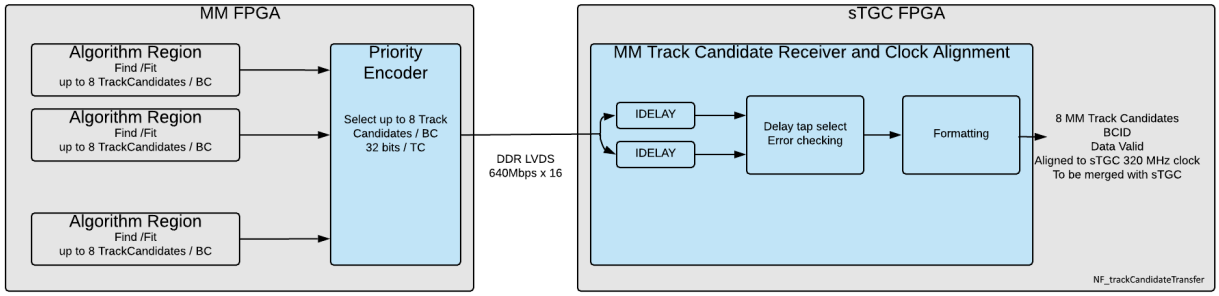


Figure 17: Transfer of MM track segments from the MM FPGA to the sTGC FPGA on the HORX mezzanine card

3.2 Removal of "duplicate" segments

Figures 18, 19 illustrate the two step process of removing duplicate segments so that at most eight segments will be passed to the Sector Logic from at most four sTGC and at most eight MM segments. The first step is to mark all "duplicate" segments so that only one is potentially chosen by the second step. Duplicates are segments that have equal $\Delta\theta$, ϕ index and R index. The fuzziness of the match, i.e. the number of significant bits to be compared, is configurable and will be determined from experience. Optionally, the ϕ-index of the MM, which has better ϕ resolution, can be taken instead of from the sTGC. In addition, lower quality segments from four-layer versus eight-layer coincidences will be disfavoured.

The second step is to gather up to eight segments to the output registers. This is done by using the vector of 12 chosen bits to address a look-up table whose outputs control multiplexers that route the segment data to the eight output segment registers. The look-up-table was prepared by software that chooses, for each of the 2^{12} possible vectors of chosen valid segments, which of the chosen segments (or null segment) is to be routed to each of the eight output segment registers. If there are more than eight segments after duplicate removal, it is currently undefined which eight will be chosen.

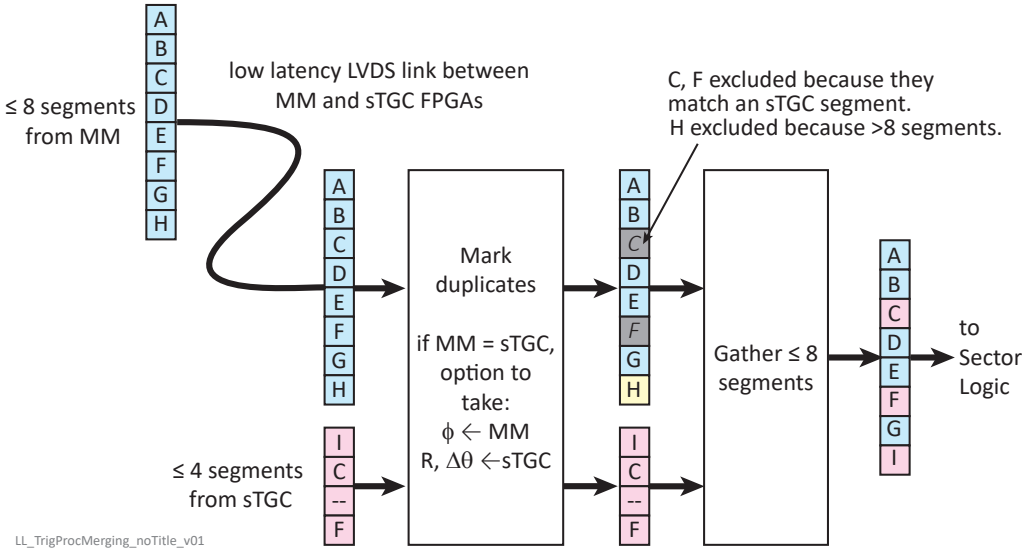


Figure 18: Merging of MM and sTGC segments for the Sector Logic by removal of duplicate segments

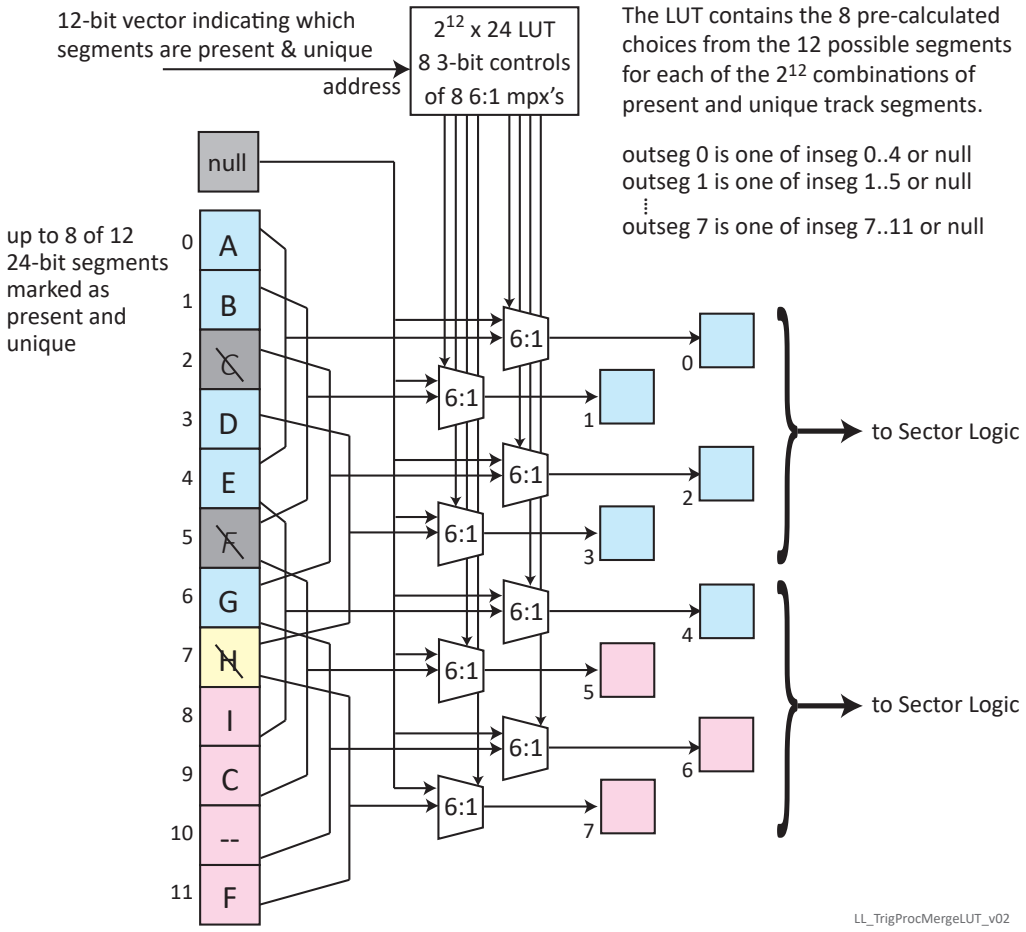


Figure 19: Gathering the chosen segments for output to the Sector Logic

3.3 Fixed latency output to the Sector Logic

The sTGC Trigger Processor runs with clocks that are multiples of the BC clock recovered with fixed latency by the carrier FPGA. The fixed latency version of the CERN GBT-FPGA firmware [9, 10] is used. The 320 MHz FPGA design clock is edge-aligned to the 40 MHz recovered BC clock by an MMCM. A zero-delay jitter cleaner guarantees that the output serializer reference clock is also edge-aligned to the BC clock.

The start of the output from the GTH serializer transmitting to the Sector Logic that was initiated from the 320 MHz design domain is therefore guaranteed to have fixed phase with respect to the 320 MHz design clock. It remains to start at a defined one of the eight 320 MHz clock edges in a BC to guarantee fixed phase with respect to the BC clock.

Although the Trigger Processor algorithm pipeline has fixed latency, its start time is not fixed across power cycles and resets. The delay between the arrival of a bit to the deserializer input and the write to the parallel deserializer output userbus varies by ± 1 receiver userbus clock between power cycles and resets. Since the pipeline starts as soon as all the input data is ready (after configurable delays compensate for differences in fibre length), there will therefore be a variation (between power cycles) in the time the output segments are ready to transmit to the Sector Logic.

Fixed latency is attained by measuring, over several power cycles of the whole trigger path, the latest 320 MHz phase within a BC at which the segment data is ready and saving that value as a configuration parameter, N. The output to the Sector Logic is then always begun at the 320 MHz clock with phase N+1 (modulo 8), that is after the longest possible variation in the latency. The phase counter is reset by BCR. See Figure 20.

In this way, the latency from collision at the interaction point to the start of output to the Sector Logic is fixed across power cycle and resets.

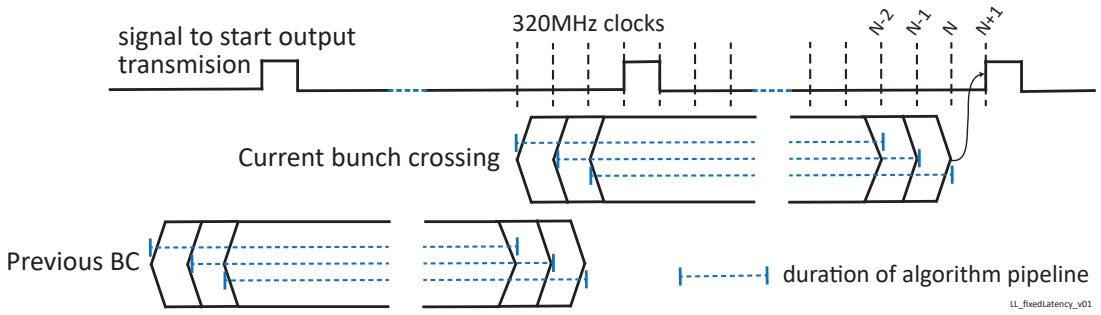


Figure 20: Timing diagram showing the (fixed) duration of the algorithm pipeline for three different power-ups (dotted lines). To guarantee fixed latency of the output, transmission to the Sector Logic is begun one clock after the latest possible completion of the pipeline. “N”, between 0 and 7, is one of the eight phase 320 MHz phases of the BC clock at which the latest completion of the pipeline occurs. “N” is found by observing several power cycles of the complete trigger path.

The fixed latency across power cycles of the Trigger Processor of the output transmission has been verified by measurements with an oscilloscope. The time delay between the edge of BCR from the TTC module (used as a stand-in for the collision) and a signal when the segment data was sent to the output serializer was measured and found not to change more than 1 ns after power cycles and resets. A test point on the Trigger Processor PCB was used to extract this signal.

4 Configuration, control and status of algorithm and operating parameters

Table 11 shows the status and control registers common to both sTGC and MM, specific to the sTGC and specific to the MM.

For configuration and status, there is no logical distinction between the Trigger Processor and the Front end devices which are configured via FELIX and the SCA ASIC. For the back-end software, uniformity with the SCA path is desirable. The Trigger Processor already requires a FELIX connection for readout on Level-1 Accept. These considerations point to using the SCA protocol between the NSW configuration process and the Trigger Processor FPGA's.

The SCA protocol uses an OPC client to interface to both the DCS and the NSW configuration process, and an OPC server to communicate to FELIX. On the Front end boards, an SCA is connected to an E-link of the same GBTx ASIC that handles the TTC and Level-1 Accept readout E-links. The Trigger Processor implements firmware using the SCA protocol so that its configuration register banks emulate SCA I2C devices. Each of up to 16 I2C masters can address up to 1024 32-bit registers. The SCA protocol is a very simple command–response, stateless protocol. Figure 21 shows the data formats and the communication path from the NSW configuration process to the status and control blocks of each FPGA on each mezzanine card of the Trigger Processors. The implementation is called SCA eXtension (SCAX) and uses 8b/10b encoding instead of the HDLC bit-stuffing used by the SCA ASIC. Detailed documentation can be found at [12–14].

Each of the 16 SCAX I2C masters accesses the FPGA logic's registers via an associated register file, which acts as a demultiplexer for write commands, or as a multiplexer for read commands. Upon a user's write request, the corresponding I2C master drives its register file's demultiplexer with the register address and propagates the data-to-be-written to the desired register. Upon a read request, the register address is used to drive the corresponding register contents to the I2C master. The data that were read are then forwarded to the back-end software.

Important extensions are: 1) A register address can point to a RAM and the FPGA generates read or write enables and automatically increments the RAM address on every read or write; 2) A register address can point to a FIFO and the FPGA generates read or write enables on every read or write. This is used to configure look-up tables and to read out debug info captured in a FIFO.

A high-level user interface has been developed called `NSWSCAXRegisters` which creates vhdl, C++ header files, xml for OPC UA, JSON for the configuration software, and documentation tables. The required registers and their properties are described in a CSV format. Scripts then build a YAML database of these registers and the package `wuppercodegen` is used in conjunction with a series of code templates written using the `JINJA2` standard. In addition, a Continuous Integration setup runs such that user interaction is confined to CSV updates, and running of the scripts and production of the various sources are automatically done for the user. This integration also ensures full versioning of the register configuration.

Table 11: Status and control registers. For a complete list, see the CSV files [11]. The NSWSCAXRegisters package generates vhdl, C++ header files, xml for OPC UA, JSON for configuration, and documentation based on the CSV input.

Status registers	**Control registers**
FELIX RX link OK	Endcap/Sector ID
Jitter cleaner status (2)	$\Delta\theta$ cut
Firmware version / time stamp	L1A BC window
	Z (IP to layer distance)
	1/Z
	monitoring BC window
	BCID offset
	Level-1 ID
	XVC IP low byte
	HORX PLL reset

sTGC status registers	**sTGC control registers**
Input QPLL lock (3)	enable MM segments
Input CPLL lock (36)	enable output to Sector Logic
input link OK (36)	max cluster size
SL output CPLL lock (14)	min cluster size
	charge threshold for determining cluster size
	charge threshold for determining centroid
	strip direction (+ / -) list
	radial offset (per gas gap)
	params for alignment corrections
	for each 1/4 pad in each layer of a sector
	which of the 8 320 MHz phases of BC clock to start output to SL

MM status registers	**MM control registers**
GBT link status	MM strip zero offset
PLL lock status	List of inverted MMFE8s
GBT packet alignment error	Cotangent 1.5°
FIFO overflow errors	Cosecant 1.5°
Communication errors	Hit integration BC window width
	Physical arrangement of x u v planes
	Cartesian ROI map
	parameters for M_X^l calculation
	parameters for $\Delta\theta$ calculation
	List of connected ADDCs
	GBT link reset
	GBT PLL reset
	GBT pkt alignment control
	Diagnostic FIFO enable
	ART data enable

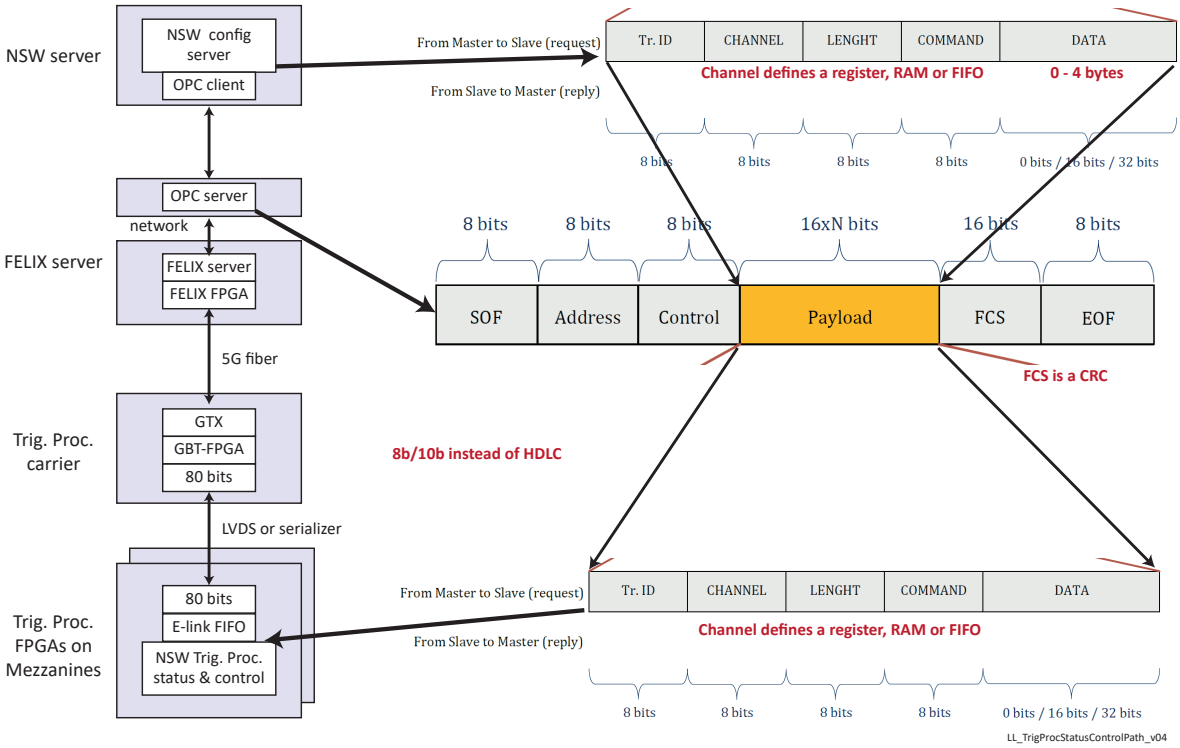


Figure 21: Configuration and control path using the SCA OPC-UA protocol.

5 Latency estimate

The latency budget for the NSW from bunch crossing to input of the segment information to the Sector Logic is 1075 ns. See [15] for a detailed breakdown of the latencies along sTGC trigger path and fiber routing. The latency of the current design of the NSW trigger path leaves little contingency. It is therefore imperative to keep the latency as low as possible at every step of the chain, including in the Trigger Processor. The Trigger Processor latency estimation is given in Table 12 for both detector technologies. The accounting includes the input/output serializers, the trigger algorithm, the algorithm that merges the MM and sTGC segments and the transmission time to the Sector Logic. The Trigger Processor start times are based on individual latencies of the upstream elements which are a mixture of measurements, simulations and estimates. Work is ongoing to improve the estimates and some measurements will be made in the NSW Vertical Slice. Consequently the total latency from bunch crossing to input to the Sector Logic is still uncertain. Since the sTGC is the critical path, segment merging is planned to be done in the sTGC FPGA.

Micromegas Trigger Processor latency The latency from VMM output to Trigger Processor track segment coincidence trigger (finder) have been measured during a hardware integration workshop. Timing measurements for the Micromegas track segment fitter components are taken from behavioral simulations. The total algorithm latency from GBT fiber input to segments arriving in the sTGC is 267 ns. This estimate assumes a three bunch crossing hit integration window. Recent studies suggest the hit integration window will need to be increased to four or more bunch crossings. Each additional bunch crossing will increase the latency by 25 ns.

Table 12: Trigger Processor latency for the MM and sTGC streams. The time required to merge the segments from the two streams is assigned only to the sTGC, since the merging is done in the sTGC FPGA.

	sTGC (ns)	MM (ns)	
Start time	795	637	Arrival of input data
Input deserializer (Rx)	77	44	
Clock alignment & distribution	31	44	
BC integration window		75	3 BCs for MM
Trigger algorithm	56	25	320 MHz clock
MM Track Segment Fitter		41	dTheta calculation critical path
Format & Tx MM segments to sTGC		38	up to 8 pipelined segments
End	959	904	MM finishes first
Segment merging algorithm driving output serializer	13		Assigned to sTGC
Output to Sector Logic serializer (Tx only)	21		to 1st bit out (Deserialization is in Sector Logic latency budget)
Fiber to Sector Logic	25		5 m fiber @ 5 ns/m
End	1018		

sTGC Trigger Processor latency The sTGC Trigger Processor latency has not yet been measured. This section will be updated with results from the simulation.

6 Monitoring

Monitoring enables validation of the data processing in the Trigger Processor by providing data for debugging and fast online analysis. Monitoring covers L1A events, a sampling of bunch crossings of interest, as well as additional dedicated monitoring data to validate the trigger algorithms and data flow. The monitoring data is intended to identify algorithm failures that may arise from occurrences such as:

- Misalignments
- Fake events from backgrounds and noise
- True events that are missed due a range of effects including non-functional detector elements or timing errors

Figure 22 shows a block diagram of the NSW Trigger Processor monitoring path. While the details of the data used for monitoring depend on the specific detector technology, sTGC or MM, the common aspects and infrastructure will be described first here.

Monitoring data is buffered on the Trigger Processor pending a Send Monitoring Data request for monitoring, which could correspond to a L1A, a random signal or an internally generated signal. The monitoring data is sent from the Mezzanine cards through the carrier card FPGAs to the Rear Transition Module (RTM) behind the carrier card. The baseline, as shown in Figure 22, is to buffer on the Mezzanine card FPGAs,

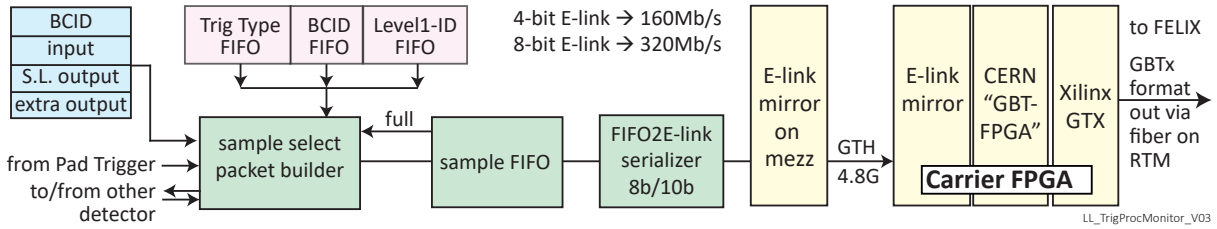


Figure 22: Block diagram of the NSW Trigger Processor monitoring output. The "from Pad Trigger" and "to/from other detector" signals enable choosing the same BCID to be monitored by all three components.

though an alternative is to buffer on the carrier card FPGAs (for example to reduce resources used in the Mezzanine FPGAs).

Complete data from a sample of bunch crossings are collected for sending to a monitoring process. Example selection criteria are: a random BC, a BC with at least one segment found, a BC with segments outside the $\Delta\theta$ cut, a Level-1 accept BC, etc. The data buffered includes all the input data and the output segment data that is sent to the Sector Logic for a configurable window around that BC.

Example diagnostics that the monitoring can provide include: the number and distribution in $R - \phi$ of bunch-crossings that have candidates but are not accepted by Level-1, multiplicity and distribution of input VMM ART signals and sTGC bands per BC, multiplicity and distribution of slope coincidences, θ, ϕ and $\Delta\theta$ distributions, types of track candidates (number of coincidence layers, etc.)

In addition to sending data to a monitoring process, the SCAX system allows readout of counters and statistics accumulated in the FPGA that can be transferred to the DAQ Information System (IS). For example: the number of segments accepted and rejected per bunch-crossing, the count of link transmission errors and the volume of monitor data.

The monitoring data streams are transferred via E-links on GBT links connected to FELIX as part of the ATLAS TDAQ system. The GBTx ASIC is emulated by the CERN GBT-FPGA firmware and the Xilinx GTH serializer/deserializer. Interfacing the data streams to the GBT protocol as E-links is the baseline choice due to its simplicity and because this link already is required for the Level-1 Accept data. The monitoring data is then transferred to FELIX for communication via Ethernet to processes on PCs. FELIX routes the various logical data streams to the appropriate network end-points, including Monitor processes.

Two LVDS lines between the sTGC and MM FPGAs exchange the Send Monitoring Data signal across the two technologies so that the same BCID from both detectors can be monitored. The sTGC Trigger Processor also receives requests to monitor signals from the Pad Trigger based on 3-out-of-4 coincidences in a pad tower.

Monitoring data is buffered at several stages:

- The monitoring data is first stored in a circular buffer that continuously keeps a moving window of data samples (typically comprised of the raw hit information and the corresponding trigger information available for each BC). There is a dedicated buffer for each readout element (typically strip or pad, or layer). When a Send Monitor Data signal is received, a contiguous block of data samples for the BC range desired (typically 5 to 10, depending on the type of monitoring data that is being sent).

- The monitoring data is then packaged into a second stage FIFO in preparation for transmission.
- Once the second stage FIFO has received all the data following the Send Monitoring Data signal, the results are merged, the header information included and data are sent out sequentially in E-link packets.

6.1 Monitoring Data

The data available to be included in the MM monitoring data stream is collected at various points in the algorithm pipeline. Data that is included in the monitoring stream will be selected as needed due to the throughput limitations. Monitoring data can contain two general types of data. This is shown in Figure 23.

1. ART GBT data packet as sent from the ADDC. Only a small subset of the total ART data being received can be selected to be included in the monitoring data. ART Data, is selectable at the fiber level. All non-zero GBT packets on the enabled fibers will be output through the monitoring path.
2. Coincidence Trigger Data. The Coincidence trigger data will contain ART hit data that forms a coincidence trigger along with algorithm debug values and track fit results.

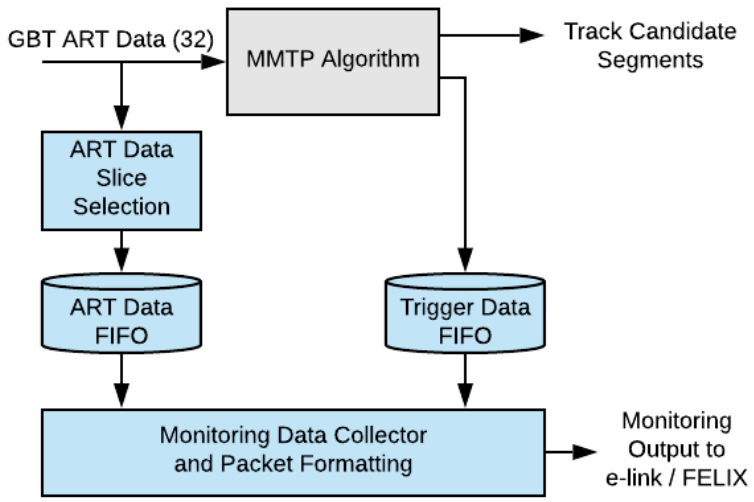


Figure 23: Block diagram showing the monitoring data flow for the Micromegas algorithm.

Figure 24 shows the current debug monitoring interfaces for the sTGC. The output is via E-links to FELIX using the FPGA-GBT framework shown in Figure 22.

6.2 Correlated MM– sTGC monitoring

In order to perform correlated MM – sTGC monitoring, each Trigger Processor can signal the other that the current bunch crossing should be monitored by both. In this case both will send monitoring data via FELIX to the NSW monitor processor. The Pad Trigger can also signal the sTGC Trigger Processor to monitor the

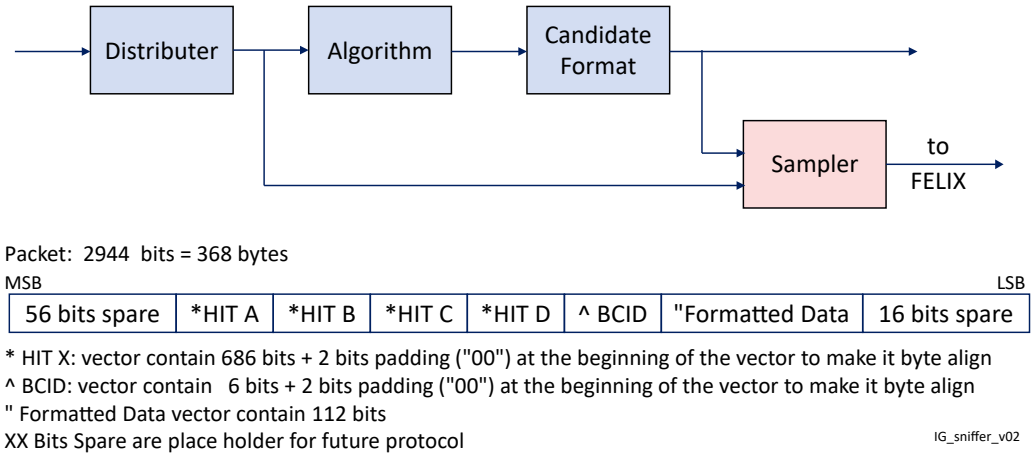


Figure 24: Block diagram showing the monitoring points for the sTGC and the output format for debugging.

current bunch crossing. The signaling is via two of the LVDS lines connecting the two mezzanine FPGAs. Since the two Trigger Processor trigger pipelines do not operate synchronously with respect to each other, they will be required to buffer their data and handle the monitor request in a manner similar to how they handle Level-1 Accept.

7 TTC and Level-1 Accept

For each bunch-crossing, the input data (raw and hit), the output segment data that is sent to the Sector Logic, and error/status information along with the BCID for later matching to the BCID of a Level-1 Accept, is stored for the duration of the Level-1 latency. Those bunch-crossings that have Level-1 Accepts (and usually some of those preceding and following) are transferred to output buffers and combined into a Level-1 packet that is sent to FELIX. The output bandwidth should be sufficient for the fixed input and output data lengths at the full Level-1 rate of 100 kHz for Phase-1 (1 MHz for Phase-2). A BUSY output to the RODBUSY system, must be sent if the output derandomizer buffer becomes close to full. BUSY symbols can be inserted anytime in the link carrying the Level-1 data to FELIX. FELIX will then assert RODBUSY to the ATLAS Central Trigger Processor.

Requirements:

Size of the BC window read out: up to eight BC's (not necessarily symmetric around the BC of the L1A) can be read out.

Depth of the latency buffer: For a 2.5 μsec trigger latency, data from 100 bunch crossings must be saved. For sTGC, the input is 104 bits from each of the four possible TDS ASICs in each of eight sTGC layers. Saving all the input data for three bunch crossings would require a modest buffer of about 150 kB. For most bunch crossings there is at most one band to process and much of the data is zero, so a huge zero-suppression is possible.

BC offset register: The value loaded into the BCID register on BCR, since the arrival time of BCR is different from its arrival time to the front end.

Level-1 ID register: For "stop-less recovery" it is necessary to set the L1ID.

Sector-ID register: The sector id of this instance of the Trigger Processor.

Output bandwidth The bandwidth needed is expected to be larger than that provided by a single 320 Mb/s E-link, partly because more than three bunch crossings are likely to be read out for MM. The software ROD supports transmitting L1A events in round-robin over several E-links. Ten such E-links can be carried by a GBT normal-mode fibre, of which at least eight would be available for readout of L1A data.

For sTGC, the case with zero-suppression of the input by only a factor of two and no suppression of the output requires 1/2 × 32 inputs × 104 bits = 208 bytes for the input data and 8 segments × 16 bytes = 128 bytes for the output data, for a total of 336 bytes per Level-1 Accept. At 100 kHz and including 8b/10b encoding, two 320 Mb/s E-links are sufficient.

7.1 TTC data

The eight TTC bits that are received on every bunch crossing are:

`L1A, L0A, BCR, ECR, EC0R, test_pulse, soft_rst, SCA_rst`

L0A, EC0R are required only in a two-level trigger. In a single level trigger, L0A is sent along with L1A. The current Trigger Processor firmware does not support a two-level trigger. Soft reset is used to flush all data in the data path, synchronously over the whole NSW data path.

7.2 Level-1 Accept Algorithm

The Level-1 Accept algorithm concept may be described as a matrix, as shown in Figure 25. Each row except the first one handles one data stream and each column handles an L1A. The first row stores the header information that is common to all processing cells in a column. Note each data stream may contain different types of information (raw data, segments, etc.).

L1A requests are buffered until a processing column is available. Once all the data has been gathered for one L1A request, the information is sent to the packet builder module and the column is released for use by another L1A. The packet builder sweeps the columns for data to be sent and the data is sent in the same order that the L1A were received.

7.3 Level-1 Accept Implementation

Figure 26 shows a block diagram for the data path for the Level-1 Accept data. The input data is buffered for the duration of the trigger latency (latency pipeline). Once the L1A signal is issued, the data associated to the L1A (e.g. BCID, Orbit count, Level-1 ID) is collected (derandomizer) in a processing column. Once all the data associated to the L1A is collected, the L1A packet builder retrieves all the data in the desired format and sends it to FELIX. The Level-1 Accept implementation is located on the mezzanine FPGA's.

The clock used is 320 MHz (8× BC clock). The data input interface is a variable length payload associated with a BCID and a write strobe to validate the payload. Each processing row has a data input interface. The payload length is a multiple of 16 bits in order to interface to the FELIX E-link module.

In this implementation, the data flows continuously from the algorithms to the processing cells, where the data is matched to the L1A BCID. The matching is performed in a configurable (and possibly asymmetric)

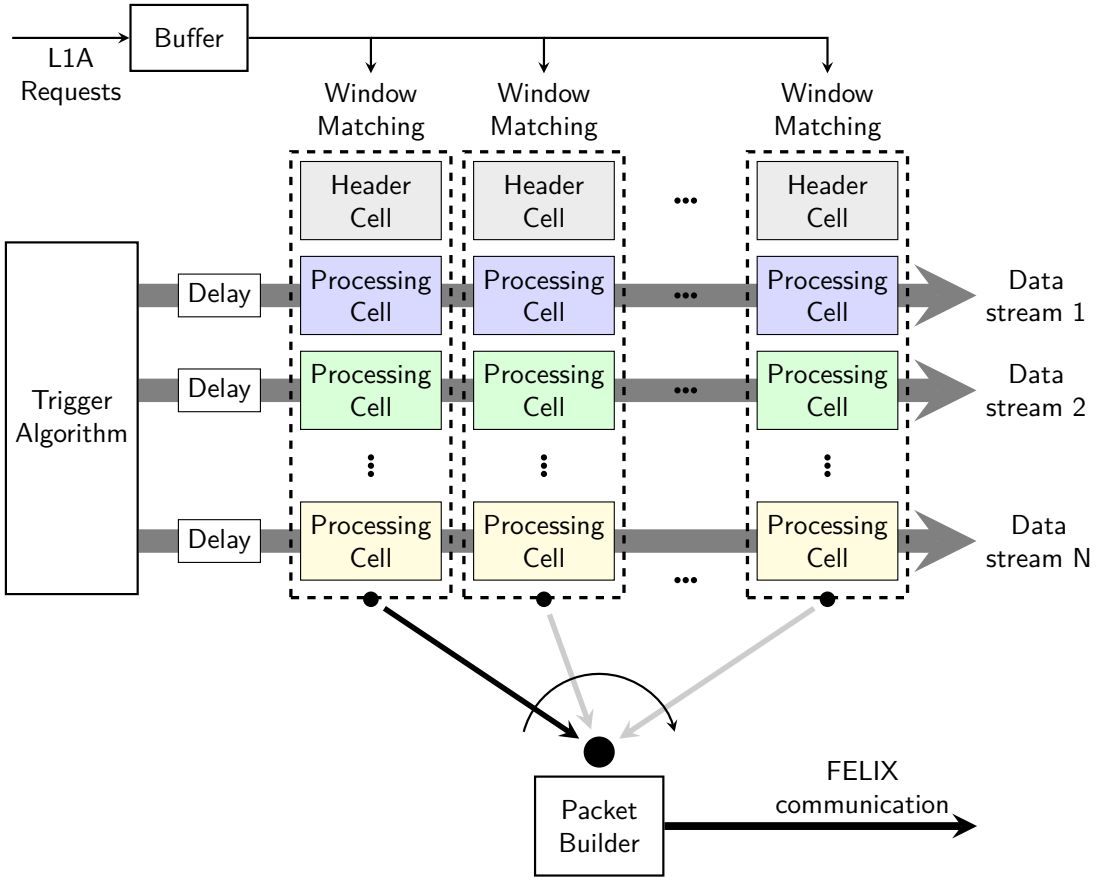


Figure 25: Conceptual design of the Level-1 Accept algorithm

window around the BCID of interest. The data formats for the data input to the L1A firmware are defined below. The input buffers are connected to all concurrent processing cells.

The L1A Manager receives a single boolean signal indicating a L1A for a BC and calculates a range of BCID's associated to the L1A. The range is defined according to a configurable offset. The L1A manager then configures and activates the first processing column that is available with the BCID of interest. Note that multiple columns can be active simultaneously. The L1A manager watches for the end of the process and instructs the L1A packet builder to drain the corresponding column.

Each processing cell stores the data in its FIFO. The data format used is the same as the input information, with the exception of headers that are added for subsequent packet handling downstream. This implementation allows for flexible input data streams, accommodating both sTGC and MM, as well as different data types (raw data, segments, etc.). The header is included by the header cells and is common for all packet types.

The output of the L1A packet builder is a "FIFO to E-link" module provided by FELIX. The L1A packets are sent via FELIX to the swROD and monitoring processes for further processing.

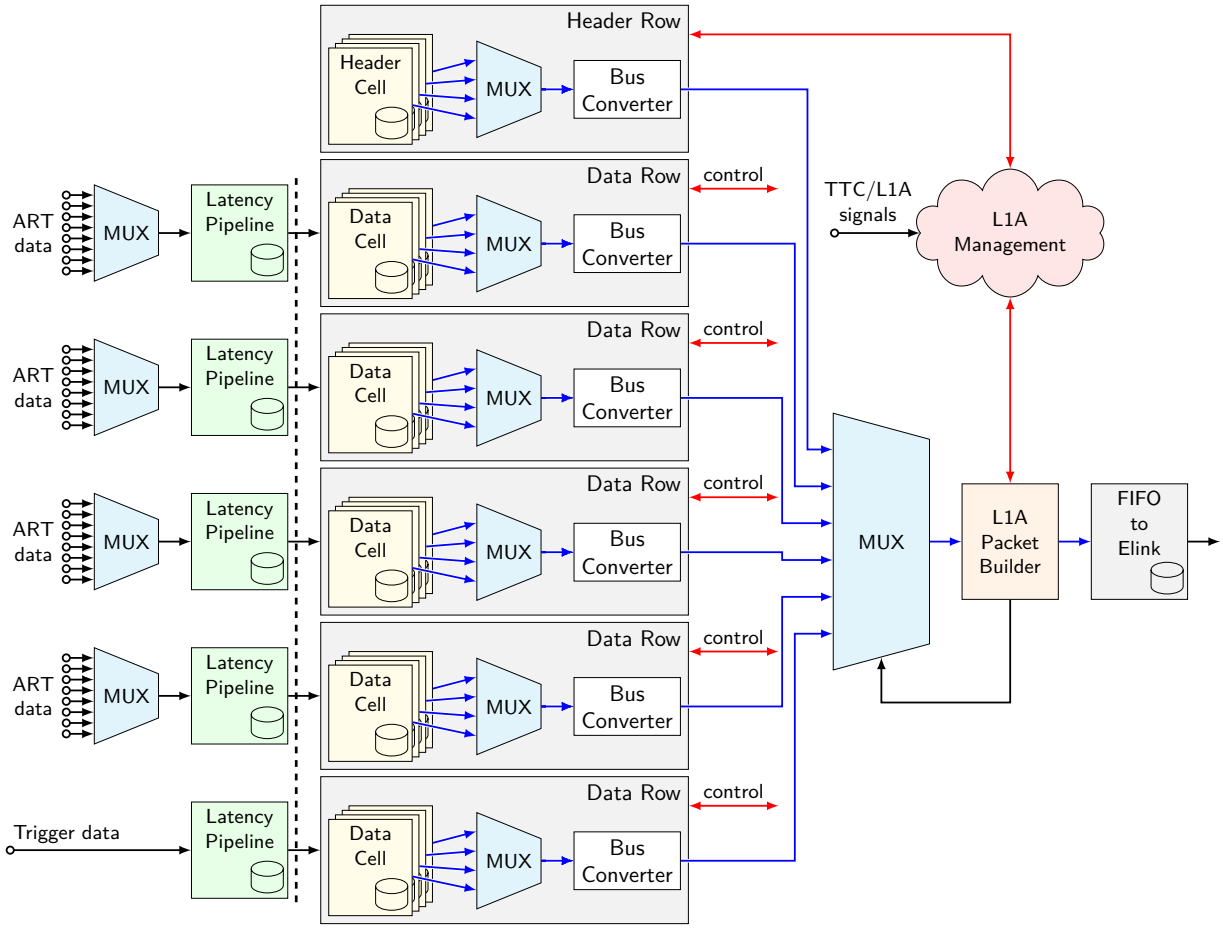


Figure 26: Level-1 Accept data flow

7.4 Read out data format

The format of the Trigger Processor data fragments sent to the swROD include a packet header followed by the specific information provided by the corresponding trigger algorithm (sTGC or MM, raw data, trigger segments, etc.). The overall format is shown in Figure 27 and the specific header is shown in Figure 28. The data formats listed below for sTGC and MM correspond to the detector-specific data streams included in the full packet.

7.4.1 Micromegas read out data

For the MM, three kinds of information are foreseen to be output by the L1A processor: raw data, hit data and fitter data. The data formats are: (See also Figure 29.)

- *MM Raw data:* comprised of 256-bit packets sent over 16 links → 16 links ×1 packet/BC corresponds to 4k bits. If the data extended over a ±8 BC window, this corresponds to 64k bits. The proposed implementation uses a 512-bit bus with the 320 MHz clock, such that one BC of raw data can be read per BC.

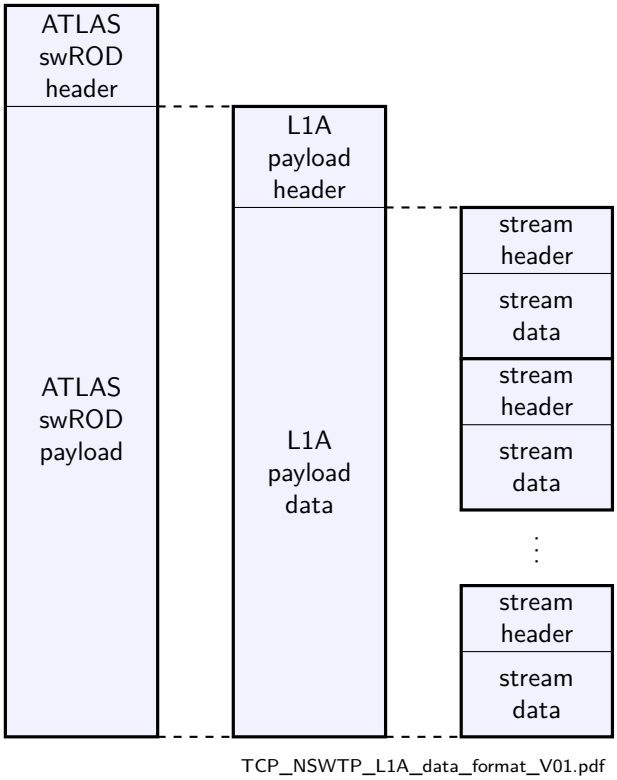


Figure 27: Level-1 Accept data format

Standard Header for input to the swROD and monitor																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
spare		orbit		BCID												flags[6..0]							EC	sector id				frag id		
Level-1 ID																															

Flags for Trigger Processor output to swROD										MM VMM hit data	frag ID = 0
6	5	4	3	2	1	0				sTGC VMM hit data	frag ID = 1
Mon				# segments						Pad Trigger input data	frag ID = 2
										Pad Trigger output data	frag ID = 3
Mon:	Monitoring of this BC was forced									MM trigger processor input data	frag ID = 4
										MM trigger processor segment data	frag ID = 5
										sTGC trigger processor input data	frag ID = 6
										sTGC trigger processor segment data	frag ID = 7
										output data to the Sector Logic	frag ID = 8

Figure 28: Level-1 Accept data header

- *MM Hit data* comprised of 416-bit packets. It is potentially larger than raw data if there are multiple packets per BC for a given region. The proposed implementation uses a 576-bit bus with the 320 MHz clock, such that one BC of hit data can be read per BC.
- *MM Fitter data:* comprised of 416-bit packets. This data is only generated in case of a coincidence.

Figure 29: Format of the MM data sent for a Level-1 Accept to the swROD, excluding raw data

	name	bits	description
Segment 0			
	... Fitter output		
	ROI	16	
	DTheta	16	
	mx	16	
	MX local	16	
	MV Global	16	
	MU Global	16	
	my/MX Global	16	
	Event Counter	16	
		16	bytes / segment header
	... for 8 layers		
	Hit information on track	16	
		16	bytes / 8 layers/ segment
	for one segment	32	bytes / segment
... for up to 8 segments		256	bytes for max 8 segments

7.4.2 sTGC read out data

The format of the sTGC Trigger Processor output segment data fragment sent to the swROD is shown in Figure 30. Also output is a fragment containing the input data with format shown in Table 4.

Figure 30: Format of the sTGC segment data sent for a Level-1 Accept to the swROD

	name	bits	description
Segment 0	band-id	8	
	Phi-id	8	
	Delta theta	8	
	R index	8	
	map of valid layers	8	
	number of valid layers	8	
	spare	16	
		8	bytes / segment header
	... for 8 layers		
	layer Centroid	16	
	layer num hits in centroid	4	
	layer error flags	12	why layer was rejected
		32	bytes / 8 layers/ segment
	for one segment	**40**	bytes / segment
... for up to 4 segments		**160**	bytes for max 4 segments

7.4.3 Read out bandwidths

Micromegas Level-1 Accept data payload The L1A data packet consists of RAW ADDC hit data and coincidence triggered segment data. The ADDC data is in the form of raw un-decoded ADDC GBT packet, rather than the individual hits. This means that each GBT packet containing one or more hits will contribute 96 bits to the L1A packet. Packets with no hits are suppressed.

Using an average strip hit rate of 8 kHz for $\mathcal{L} = 2.8\times10^{34}$ (from a plot produced by Alex Tuna at $\mathcal{L} = 7\times10^{34}$ giving 20 kHz), the probability of an ART GBT packet containing at least one hit is

$$\lambda = 8\,\text{kHz} \times 2048\,\text{strips/GBT} \times 25\,\text{ns} = 0.41$$
$$1\text{-}e^{-\lambda} = 34\%$$

The ART data contribution for a 5 BC window for 32 ART fibers would be

$$34\% \times 96\,\text{bits/BC} \times 5\,\text{BC's} \times 32\,\text{ART's} = 5.2\,\text{Kb}$$

For the coincidence triggered segment data, we currently collect 304 bits. Assuming we have five segments in a L1A window this will contribute 1.5 Kb. The total data size is then 6.7 Kb; a L1A rate of 100 kHz yields a required data rate capacity of 670 Mb/s. Six E-links, each providing 250 Mb/s after 8b/10b encoding, are available, providing a total of 1500 Mb/s. Of the 96 bits in each ART GBT packet, many will be zeroes. Data compression should be able to significantly reduce the bandwidth.

sTGC Level-1 Accept data payload For $\mathcal{L} = 5\times10^{34}$, the Table 13 shows that the expected number of 32 Mb/s E-links needed for the sTGC Trigger Processor is 1.36. Scaling to the expected Run 3 luminosity of 3×10^{34}, we require 80% of a 320 Mb/s E-link. Note that six such E-links are available.

Table 13: Expected number of 32 Mb/s E-links needed for the sTGC Trigger Processor

16	bits for header	
104	bits per input TDS	104 means no compression
8	layers	
1.33	average no of bands per BC. This is an exaggeration.	
3	BC's per L1A	
0.1	L1A rate, MHz	
348	Mb/s	(435 bytes/L1A)
435	Mb/s with 8b/10b	
1.36	# E-links	at $\mathcal{L} = 5\times10^{34}$

8 Configuration, control and monitoring of the hardware via I2C

Figure 31 shows a simplified diagram of the I2C buses available in the NSW TP mezzanine board. Each FPGA is connected to related set of microPODs and clock and jitter cleaner devices, which allows not only sensor reading, but also parameters configuration. There is also a bus connecting the local microcontroller

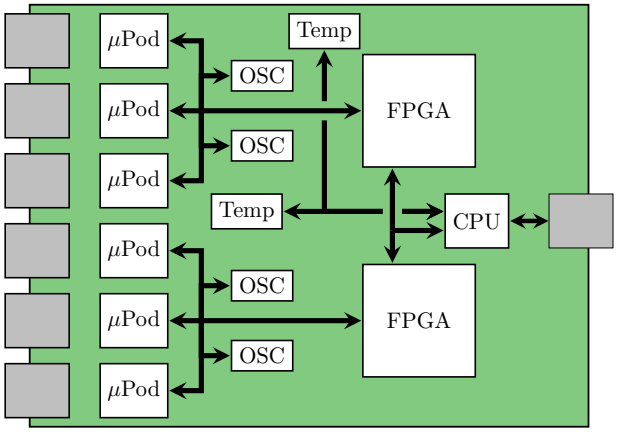


Figure 31: Simplified diagram for the I2C buses in the NSW TP mezzanine board.

to the FPGA devices, making read and write operations possible. There is another bus connecting the microcontroller to discrete temperature sensors for environment sensing.

Figure 32 shows the path from the FPGA internal parameter table to the DCS. The MMC microcontroller has a user I2C port connected to the FPGA's on the mezzanine. This port provides IPMI access to the FPGA internal parameter table. The parameters include the FPGA parameters (temperature, internal voltage) and the optical link status registers from the micropods (RX power, link-loss, TX fault, temperature, etc.). This path can also be used to monitor application flags (Trigger Processor status, error flags, etc.).

Alternatively, instead of the I2C slave, the SCAX machinery can be used to read out the parameter memory in Figure 32. This path goes to DCS via FELIX instead of via the IPMC and shelf manager.

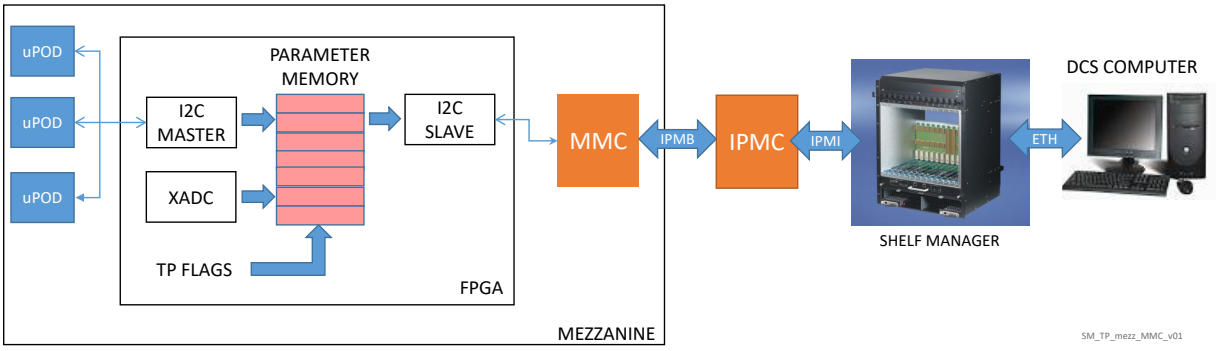


Figure 32: Simplified diagram for the access of the sensor parameters sent to the DCS.

8.1 IPMI monitoring

The management of the hardware temperatures is accomplished by the Intelligent Platform Management Interface (IPMI) protocol, which changes the speed of the fans in the ATCA shelf according to sensor readings and the configurable recommended ranges. Figure 33 shows some levels of managing resources. The Shelf Manager (ShM) in the ATCA crate gathers information of the devices inside the crate and makes decisions. The Intelligent Platform Management Controller (IPMC) is the standard interface for all

the possible sensors in the board and responds directly to the ShM. In addition, Modular Management Controllers (MMC) are used for specific units and purposes (e.g. the Mezzanine card, the RTM, etc.). All the communication ShM-IPMC and IPMC-MMC relies on the I2C protocol. The ranges and descriptions of sensors are stored in the MMC or in the IPMC as Sensor Data Records (SDR) and are promptly available upon system startup.

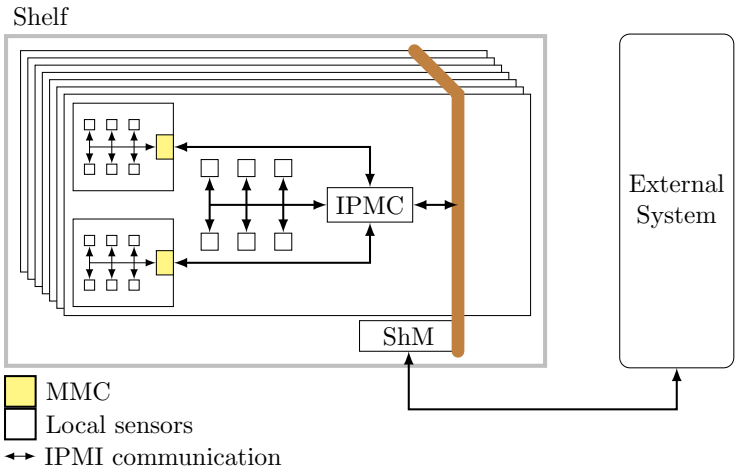


Figure 33: Simplified diagram for the I2C buses in the NSW TP mezzanine board.

Each NSW TP mezzanine board is equipped with an MMC, as shown in Figure 31 (labeled as “CPU”), which is able to read information from sensors for temperature, from the micropods and FPGA’s. The IPMC then collects the information from the MMC’s (mezzanine and RTM) and also for the many components on the carrier board, like FPGA’s, SoC, power supply, etc, as detailed in the diagram in Figure 34. Only the Zynq requires some programmable logic design for this communication but relies on IP Cores from Xilinx and software drivers. None of the additional items involves FPGA designs and the monitoring happens directly at the hardware level.

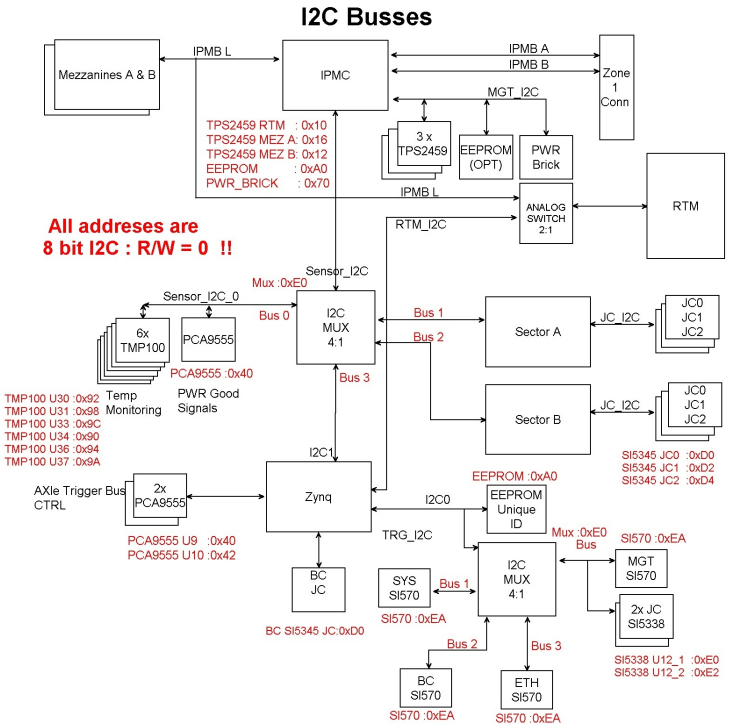


Figure 34: Detailed I2C map for the NSW Trigger Processor carrier board, involving access to mezzanines and RTM buses. Among other functions, the IPMC monitors FPGA's and SoC, transceivers, power supplies, and other sensors (temperature, current and voltages).

9 FPGA configuration

Each Trigger Processor ATCA carrier has seven FPGA's: two Virtex 7's on each mezzanine, two Ultrascale Kintex and the Zynq SoC FPGA on the carrier. In order to program all the components on the board, a scheme to expose the JTAG chains over the network is used. The Xilinx Virtual Cable (XVC) protocol is used to translate between network protocol (TCP/IP) and JTAG signals. This solution, supported by Xilinx development tools, provides access to the reconfigurable devices inside the crate, enabling remote connections.

The NSW Trigger Processor carrier board includes a Zynq System on a Chip (SoC), comprising a Xilinx FPGA and an ARM processor in the same chip. Figure 35 shows the implementation of the TCP/IP stack and XVC service. The Zynq device is in the JTAG chain and provides the FPGA programming capability. However, since the processor itself cannot be reprogrammed once it is running the XVC service, an external flash and boot loader is used, which are directly controlled by the processor and outside the JTAG chain.

In this scheme, the Zynq SoC on the blade provides configuration resources for its own FPGA, as well as for the JTAG chains in its mezzanines. It also provides network access using the fabric interface of the backplane or via RTM, ensuring that all the configuration resources for every blade in the shelf are available through network switches. If the Zynq SoC configuration is disrupted, it can be reprogrammed through the XVC service provided by the CERN IPMC as an alternative path, also available via network connection. As yet another recovery procedure, the carrier Zynq is also equipped with two flash memories. A golden firmware is stored in the secondary one, which can be activated by the IPMC via shelf manager.

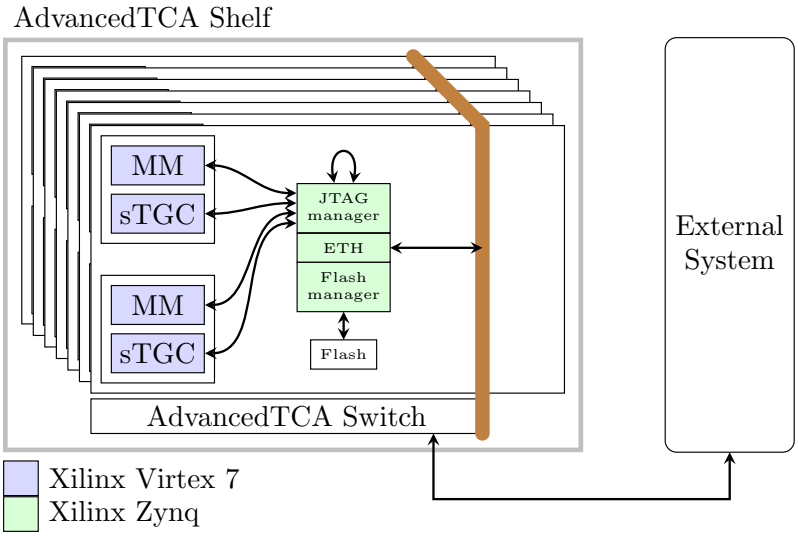


Figure 35: Example of programming scheme using the XVC service via Zynq SoC and network through an ATCA network switch. A similar approach can be done with a regular network switch since each carrier provides network access on the RTM.

10 Playback of test data

In order to test the Trigger Processor it is possible to *play-back* simulated events (Athena). To produce the expected inputs to the Trigger Processor with an approximate description of their time wise features (in particular to emulate the correct time relationship between the MM and sTGC Trigger Processors), events obtained from the Athena simulation are used as inputs to a time-based simulation of the NSW trigger chain. The expected inputs are loaded to the Trigger Processor via the SCAX and then played back (event → Tx → Rx → Trigger Processor algorithms).

The time-based simulation is driven by an event-based scheduler. The first version of the simulation is to be released in the upcoming weeks. Currently the simulation includes approximate models for the following items in the NSW trigger chain.

- MM: MMFE8
- MM: Cabling from MMFE8 to ADDC
- MM: ADDC
- MM: Fibres from ADDC to Trigger processor
- sTGC: sTGC strip adapter boards (to describe which strips are connected to which VMM channel)
- sTGC: sTGC front end boards
  - VMM direct output 6-bit ADC
  - TDS
  - VMM to TDS connections
- sTGC: Pad trigger (using an external coincidence table)
- sTGC: Cabling from Pad Trigger to sTGC strip FE-board
- sTGC: Cabling from strip FE-board to Router
- sTGC: Router (includes mis-match between fiber input ordering and serializer block order)

- sTGC: Fibres from Router to Trigger Processor

11 Firmware organization and continuous integration

The firmware language used in the Trigger Processor collaboration is VHDL. Xilinx Vivado is used for both synthesis and place & route of the firmware. All firmware is stored in the CERN Gitlab group https://gitlab.cern.ch/nsw_trigger_processor_elx.

Multiple institutes are involved with the development of the Trigger Processor firmware. There are firmware blocks specific for sTGC, specific for MM and blocks that are shared. The blocks need to be integrated with each other into the two top-level designs: sTGC and MM. The standard tool flows from Xilinx Vivado or Mentor Graphics were not efficient for such a collaboration. We have adopted the design flow developed by the FELIX team that makes it possible to work together.

By default, Vivado creates a multi-level directory structure containing all the source files. The Vivado project is not included in the source tree under version control, but is generated by sourcing a TCL script from within the Vivado GUI. This script creates a fresh Vivado project by copying all the needed source files from the repositories; no source files are kept inside the Vivado project tree. An additional benefit of this approach is that the huge Vivado project is not included in version control. The project contains many binary and generated files; these files would create many changes in the version control system and would complicate the manual browsing through the repository.

A complete set of firmware needed for the Trigger Processor is versioned together in a so-called *release* repository at https://gitlab.cern.ch/nsw_trigger_processor_elx/nsw-trig-proc. This release repository represents an ensemble of git commits (as git submodules) that represent compatible Trigger Processor firmware, including all dependencies. Major versions of the entire Trigger Processor firmware set are git tagged from this release repository creating a uniform release structure for all relevant firmware. These tags take the form `NSWTrigProc-XX-XX-XX`.

Additionally, the health of all firmware is ensured using a Continuous Integration and Continuous Deployment (CI/CD) setup in CERN GitLab. Using custom CI/CD "runners" (or optionally shared CERN runner resources with the use of a Docker image-based setup), every push event launches a pipeline of jobs on the runners to build the projects and obtain `bit` files, storing them as artifacts for testing. Merge requests into the primary development branches are required to pass these build tests to ensure the primary development branch always functions. Tagging git commits also triggers a CD job to deploy the Vivado output files to a tag area in the EOS file system for preservation of tagged `bit` files as well as quick deployment of stable firmware builds to test sites.

References

[1] ATLAS New Small Wheel collaboration, *New Small Wheel Technical Design Report*, (2013), URL: https://cds.cern.ch/record/1552862 (cit. on p. 6).

[2] ATLAS New Small Wheel Trigger Processor Working Group, *Trigger Processor Design Review Report*, (2015), URL: https://edms.cern.ch/file/1470527/1/NSW_TriggerProcessor_PDR_20150210.pdf (cit. on pp. 6, 12).

[3] eicSys GmbH, *EATCA-100, FPGA based ATCA blade for RD51 SRS system, User Manual*, (2014), URL: https://edms.cern.ch/file/1476256/1/NSW_TriggerProcessor_SRS_ATCA_PDR_20150209.pdf (cit. on pp. 6, 9).

[4] Sorin Martoiu, *High-Density Optical Receiver Mezzanine board for ATCA-SRS*, (2015), URL: https://edms.cern.ch/file/1476256/1/NSW_TriggerProcessor_SRS_Mezzanine_PDR_20150209.pdf (cit. on pp. 6, 9).

[5] ATLAS New Small Wheel Trigger Processor Group, *NSW Trigger ProcessorCarrier Board Version 2 Specification*, (2018), URL: https://espace.cern.ch/ATLAS-NSW-ELX/Shared%20Documents/NSW%20Trigger%20Processor/NSWTP_NewCarrierBoardRequirements.pdf (cit. on p. 11).

[6] B Clark, D Lopez Mateos, N Felt, J Huth, and J Oliver, *An Algorithm for Micromegas Segment Reconstruction in the Level-1 Trigger of the New Small Wheel*, (2014), URL: https://cds.cern.ch/record/1753329/files/ATL-UPGRADE-INT-2014-001.pdf (cit. on pp. 17, 19).

[7] S Chan and et al., *Micromegas Trigger Processor Algorithm Performance in Nominal, Misaligned, and Corrected Misalignment Conditions*, (2015), URL: https://cds.cern.ch/record/2113121/files/ATL-COM-UPGRADE-2015-033.pdf (cit. on pp. 17, 21).

[8] J. Farah et al., *Test of the Micromegas Trigger Processor with Cosmic Ray Muons*, (2015), URL: https://cds.cern.ch/record/2285496/files/ATL-COM-MUON-2017-049.pdf (cit. on p. 21).

[9] M Barros Marin et al., *The GBT-FPGA core: features and challenges*, (2014), URL: https://cds.cern.ch/record/2158963/files/10.1088_1748-0221_10_03_C03021.pdf (cit. on pp. 23, 30).

[10] J Mendez, S Baron, P Leitao, and A Caratelli, *An Algorithm for Micromegas Segment Reconstruction in the Level-1 Trigger of the New Small Wheel*, (2017), URL: https://cds.cern.ch/record/2312287/files/PoS(TWEPP-17)083.pdf (cit. on pp. 23, 30).

[11] ATLAS New Small Wheel collaboration, *NSW Trigger Processor status and control registers*, (2019), URL: https://espace.cern.ch/ATLAS-NSW-ELX/Shared%20Documents/NSW%20Trigger%20Processor/Registers.xlsx (cit. on p. 32).

[12] ATLAS New Small Wheel collaboration, *SCX - Requirements and Implementation of Access to Control and Status Registers in FPGAs via FELIX*, (2018), URL: https://espace.cern.ch/ATLAS-NSW-ELX/Shared%20Documents/NSW%20Trigger%20Processor/SCX_requirements.pdf (cit. on p. 31).

[13] ATLAS New Small Wheel collaboration, *SCA eXtension Deployment User Guide*, (2019), URL: https://espace.cern.ch/ATLAS-NSW-ELX/Shared%20Documents/NSW%20Trigger%20Processor/scx_userGuide.pdf (cit. on p. 31).

[14] C Bakalis et al., *SCA eXtension: a Design for FPGA Parameter Configuration within the ATLAS DAQ Scheme*, (2019), URL: https://cds.cern.ch/record/2711577/files/ATL-COM-DAQ-2020-016.pdf (cit. on p. 31).

[15] S. Bressler, A. Koulouris, A. Lanza, L. Levinson, and S. Zimmermann, *Trigger latency for the sTGC in the New Small Wheel*, (2016), URL: https://edms.cern.ch/file/1604270/1/sTGC_Latency_fiber_routing_final_20160314.pdf (cit. on p. 33).