Nuclear Physics Division
Fast Electronics Group

VTP Manual

Benjamin Raydo
Chris Cuevas
Bryan Moffitt
Dave Abbott

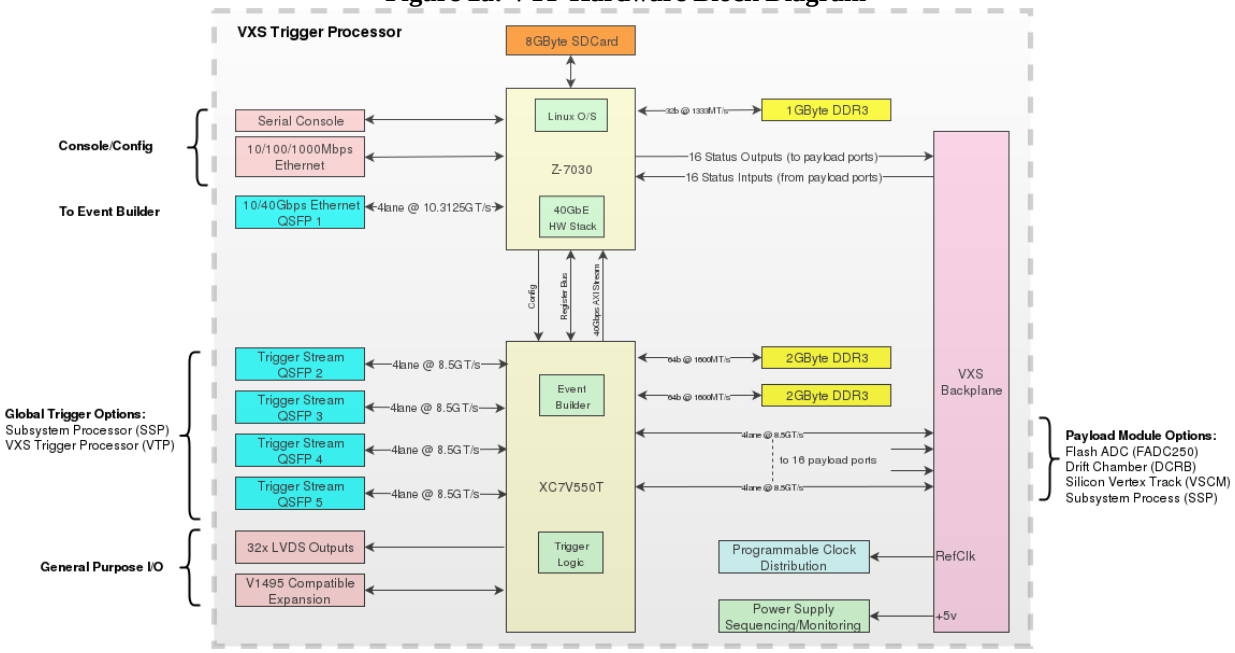
Oct 9, 2023

1 Introduction

The VXS Trigger Processor (VTP) module is a VXS switch card module that participates in the Level 1 trigger in front-end VXS crates (a.k.a. the CTP switch slot) as well as the global-trigger VXS crate (a.k.a. the GTP switch slot). The VTP design supersedes the CTP and GTP designs as it contains more backplane serial links to front-end payload modules, more fiber optics serial links to other crates, and more FPGA resources for trigger logic. Additionally it contains a dual-core 1GHz ARM processor core that runs a CODA ROC on a Linux O/S capable of event building trigger diagnostic information from FPGA trigger logic. The processor utilizes 1Gbps Ethernet connection for configuration, control, and filesystem. Additionally a 10Gbps and 40Gbps Ethernet can be used for the CODA ROC readout in a future firmware/software development. This high speed readout can be used to read from compatible Jlab VXS electronics as an alternative to VME which provides significantly more bandwidth for future experiments that demand this.

2. Functional Description

Figure 2a: VTP Hardware Block Diagram



2.1 XC7V550T FPGA

The XC7V550T FPGA manages all the VXS backplane and optical serial streams. Custom trigger firmware collects information from the front-end (VXS payload) modules using 4 full duplex SerDes lanes. Up to 4 QSFP front panel ports can be used to communicate trigger information of other front-end crates and to the global trigger crate. LVDS (32) outputs can be used to send fixed latency trigger signals to a TI master for local crate triggering or TS (Trigger Supervisor) for global system triggering. A V1495 compatible mezzanine connector exists that allow use of commercially available ECL/TTL/NIM/ADC/DAC expansion modules mainly to eliminate the need for external level translators. Two DDR3 memories interfaces exists (each with 100Gbps bandwidth) that can be used for large event data buffers in future event building applications, debug/trace buffers, trigger logic dictionaries, histograms, etc. Dual 16bit AXI streaming busses providing a total of 40Gbps connect to the Zynq FPGA which can be used to transport high speed event data to the 10/40Gbps Ethernet. For FPGA configuration and register access a 32bit data slave bus exists which connects to the Zynq FPGA who acts as the bus master.

Summary of Features for the XC7V550T

- 16x VXS payload interfaces: 4 full duplex lanes @ up to 8.5Gbps each
- 4x QSFP interfaces: 4 full duplex lanes @ up to 8.5Gbps each
- 2x DDR3 interfaces: 64bits each @ 1600MT/s
- Dual 16bit AXI streaming bus (up to 40Gbps) transmitter to XC7Z7030
- TRIG1, TRIG2, SYNC from VXS

- 32bit LVDS output (>250Mbps per bit)
- 32bit V1495 daughtercard expansion (A395x)
- Virtex7 FPGA (364000LUT, 692800FF, 5MB BRAM, 2880DSP)

Summary of Responsibilities for the XC7V550T

- Manage 80SerDes links on the 16 VXS payload and 4 QSFP interfaces
- Run detector/experiment specific level 1 trigger algorithms
- Provide scalers and configuration registers to monitor/manage level 1 trigger
- Receive readout trigger and build experiment specific event data, stream to ROC in ZYNQ
- (Future) receive event data from 16 VXS payload and stream to ROC/EB in/through ZYNQ

2.2 XC7Z7030 FPGA

The XC7Z7030 is an FPGA and processor contained in a single chip. The processor is a dual ARM Cortex-A9 which runs Linux and the FPGA provides reasonably large resources to deal with hardware interfaces.

Summary of Features for the XC7Z7030

- Dual ARM Cortex A9 1GHz processor, 1GB DDR3 RAM 32bit @ 1333MT/s
- Bootloader in microSD card
- RS232 Console (115200bps, 8b, np) – used for boot configuration and debug terminal
- 10/100/1000Gbps Ethernet – used by Linux O/S
- 32bit data/address bus master which memory maps register space of XC7V550T into processor
- 16bit FPGA configuration bus for programming XC7V550T image
- Dual 16bit AXI streaming bus (up to 40Gbps) receiver from XC7V550T
- 10/40Gbps Ethernet – intended for FPGA accelerated TCP/IP stack for high speed event building
- 1Gbps TI (PP18) full duplex link
- Kintex7 FPGA (78600LUT, 157200FF, 1MB BRAM, 400DSP)

Summary of Responsibilities for the ZYNQ Processor/FPGA

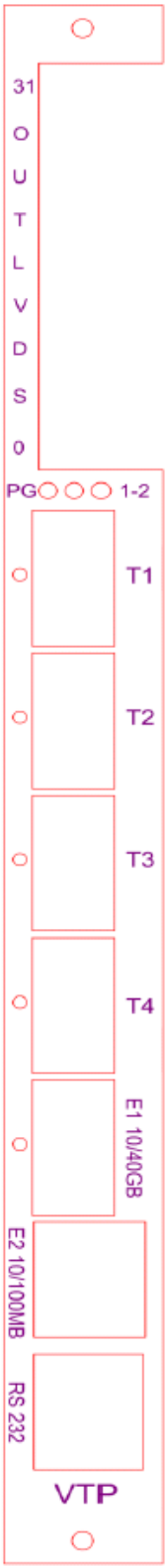
- Execute FSBL and U-Boot bootloaders from microSD card, load centrally managed Linux kernel and filesystem from NFS
- Program XC7Z7030 and XC7V550T FPGA images on boot from network
- Runs CODA ROC, event building data from TI and XC7V550T
- Report various scalers, temperatures/voltages to EPICS IOC
- Provides drivers for all VTP peripherals
- (Future) use 10/40Gbps Ethernet to stream event builder data from partial hardware accelerated ROC

3.1 Specification & I/O Summary

The VXS connection is used to interface to the trigger system without the need for loose cabling. This interface provides the following signals:

Signal	Description	Direction	Signal Type	Interface
Clock	125/250MHz System Synchronous Clock	In	LVPECL	SD
Trig1	L1 accept trigger bit, synchronous to clock	In	LVPECL	SD
Trig2	L1 accept trigger bit, synchronous to clock	In	LVPECL	SD
Sync	L1 synchronization bit, synchronous to clock	In	LVPECL	SD
Busy	Module busy signal	Out	LVDS	TI
GTP_TX	1Gbps VTP->TI TILINK	Out	LVDS	TI
GTP_RX	1Gbps TI->VTP TILINK	In	LVDS	TI
SDA/SCL	TI I2C slave interface	InOut	LVTTL	TI
STATOUT	Payload->VTP status input	In	LVTTL	PP1-16
STATIN	VTP->Payload status output	Out	LVTTL	PP1-16
L1 Trigger	8.5Gbps per lane (4) used to generate L1 trigger	In/Out	CML	PP1-16

3.2 Specification Summary



MECHANICAL

- Single width VITA 41 (VXS) Switch Module

Trigger Interface and Switch B signaling
(Signal Distribution module)

- 250MHz Clock (LVPECL)
- Trig1, Trig2, Sync (LVPECL)
- LINKUP out (LVTTL)
- BUSY out (LVTTL)

GIGABIT DATA STREAMS
From/To 16 Payload Slots (FADC, SSP, VETROC, DCRB, etc.)

- 4 full duplex lanes up to 8.5Gbps
- 544Gbps Aggregate

Outputs:

- 32 LVDS front panel outputs to Trigger Supervisor
- 1x RJ45: 100/1000Mbps Ethernet
- 32 I/O expansion mezzanine (LVDS/ECL/PECL/NIM/Analog)
- 4x QSFP Fiber Transceivers (34Gbps)
- 10/100/1000Mbps Ethernet (RJ45)
- 1x QSFP 10/40Gbps Ethernet
- RS232 console serial port

Indicators: (Front Panel)

- Power – Blue LED
- Trigger – Amber LED
- Alarm – Red LED Programming and Trigger Data Input:
- On board JTAG Port
- Virtex 7 550T; 80GTH Gigabit transceivers
- 1GHz ZYNQ-7030 SoC processor with Linux OS
- Global Trigger equation and processing for up to 16 JLAB SubSystem Processors
- Front End Trigger processing for up to 16 JLAB Flash ADC digitizers
- 8GB Micro SD card support (Linux OS file system + FPGA image)
- 1GB DDR3 SDRAM

Power Requirements:

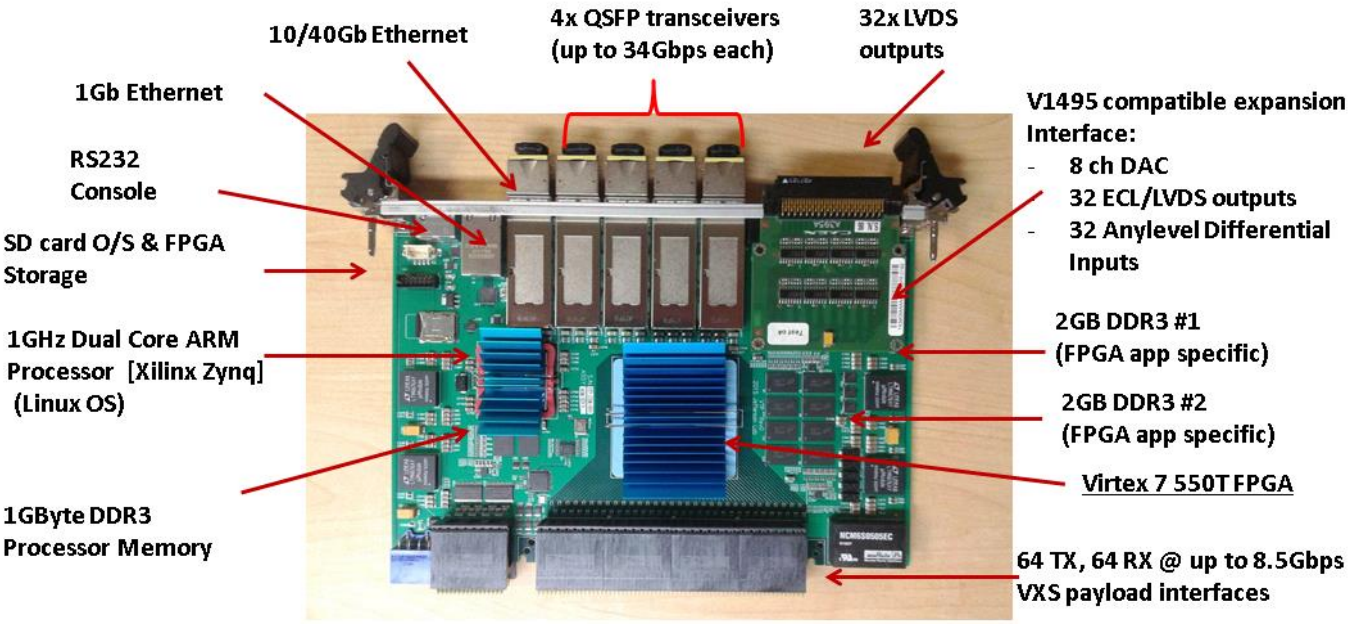
- +5v @ 10 Amps (typ. from Backplane)
- Local regulators for other required voltages

Environment:

- Forced air cooling: Heat sink
- Commercial grade components (85°C max)

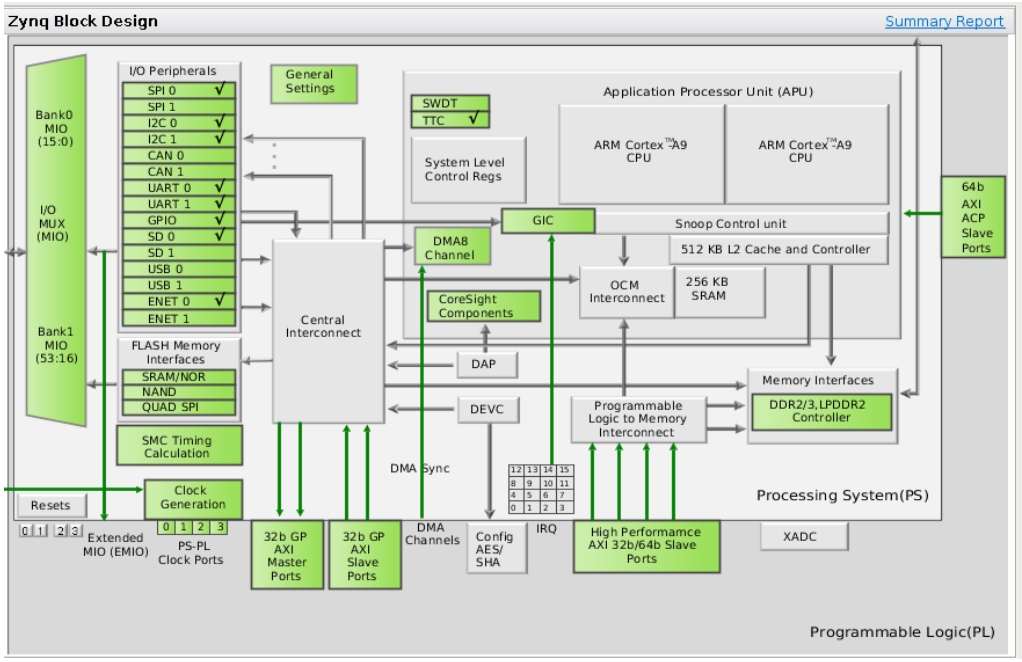
4. PCB Assembly View

The VTP PCB is a 22-layer impedance controlled FR-408HR stackup



5. ARM CPU Configuration

All VTP board registers (whether in Virtex 7 or Zynq 7) are memory mapped into the Zynq7 CPU address space. Many peripherals used in the Zynq7 are part of the PS (Processor System) and the Zynq7 technical reference manual can be referred to for details (ug585-Zynq-7000-TRM.pdf). The Zynq7 processor configuration is summarized in the following diagram:



The following table indicates which PS peripherals are used on the VTP:

PS Peripherals:

Peripheral Name	Description	Address Base
SPI0	Used by: Si5341 clock synthesizer	**0xE0006000**
I2C0	Used by: 10/40Gbps QSFP for module identification & monitoring	**0xE0004000**
I2C1	Used by: LTM4676 smbus power supply monitoring	**0xE0005000**
UART0	U-boot & Linux console	**0xE0000000**
GPIO	Used by: Si5341, LEDs, 10/40Gbps QSFP, VXS STAT IN/OUT	**0xE000A000**
SD0	µSD card is the Zynq7 boot resource (contains FSBL, U-Boot)	**0xE0100000**
ENET0	10/100/1000Mbps Ethernet for U-Boot & Linux	**0xE000B000**
TTC0	U-Boot & Linux OS Timer	**0xF8001000**

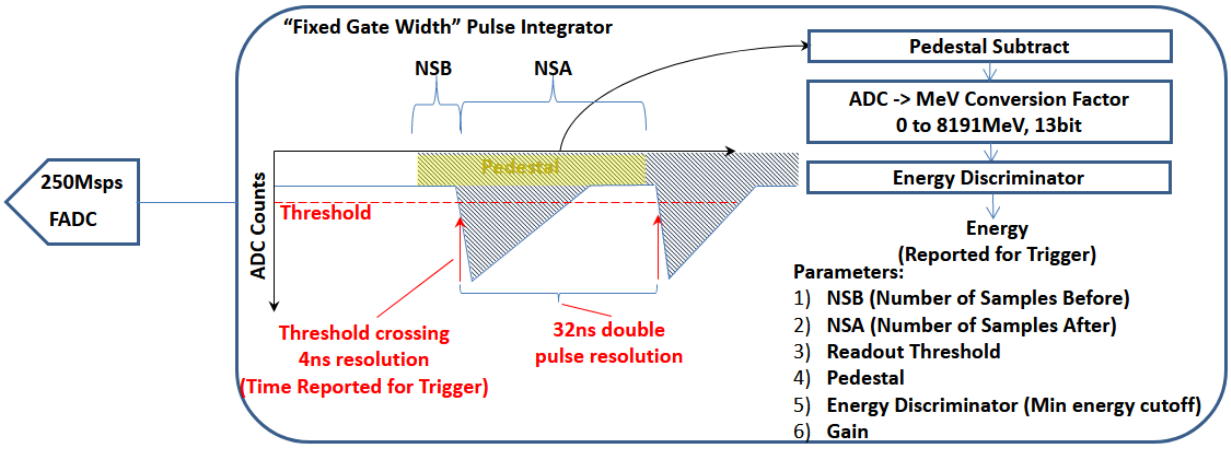
6. Streaming Readout

The VTP was developed as a trigger processor, but the high bandwidth serial connections to the VXS backplane modules as well as optical outputs has provided an opportunity to convert to a streaming DAQ architecture (avoiding the need for the trigger data/processing path).

FADC250 Streaming Firmware

The first streaming DAQ implementation with the VTP and FADC250 used the FADC trigger output as the streaming data source. The trigger output of the FADC250 uses the VXS P0 interface to streaming FADC hits over a 10Gbps interface (4 lanes at 2.5Gbps, 8b10b encoded provides 8Gbps of usable bandwidth). The 8Gbps usable bandwidth was sufficient to allow a simple reporting scheme that allows each channel of the FADC250 to report a 13bit charge and 4ns timestamp pulse hit with a 32ns double pulse resolution (e.g. 16ch * (13bit charge + 3bit fine

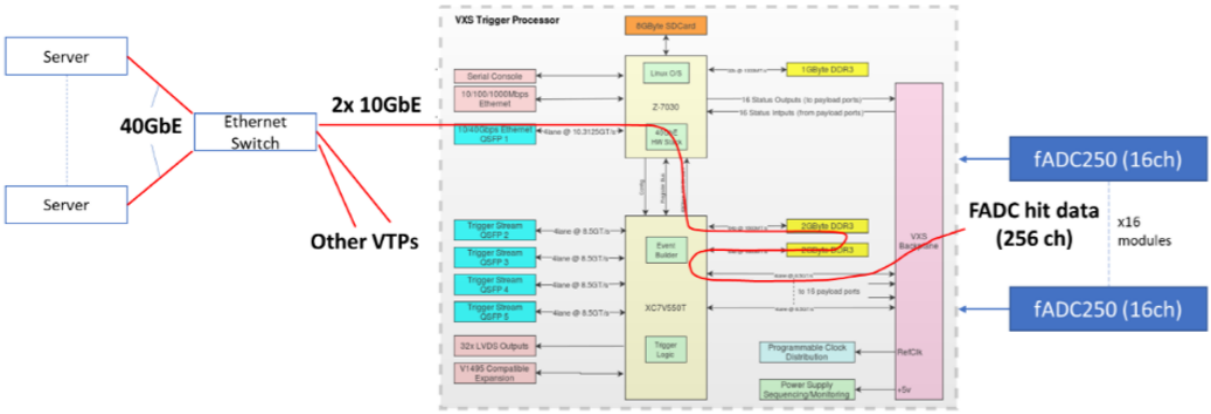
time) / 32ns = 8Gbps). The FADC250 uses a programmable threshold to detect a leading edge of pulses. When the leading edge is found, a programmable number of samples before, NSB, and after, NSA, are summed together to form the pulse integral. A pedestal is subtracted from the integral and a gain is applied to convert the value into the desired units (typical examples: 1MeV, 100keV, Npe). The resulting value is capped to 8191 so that it fits into the 13bit charge field that is reported to the VTP – if the value exceeds 8191 the FADC will report 8191 (so this capped value can be used as an overflow detection if desired). The following figure illustrates this pulse processing logic:



The FADC250 data format can change easily in the future with firmware updates. Plans are already made to change the trigger output to report more information (to more closely match the information it typically reports in VME readout mode), such as time walk corrected timestamping with much higher resolution as well as raw waveform samples. This additional information will reduce the hit rate capability of the streaming system, but the bandwidth is significant this limit will still be extremely high for most detector channels used in experiments at Jlab.

VTP Streaming Firmware

The following image shows the path of the streaming data through the hardware:



FADC Input

FADC250 data is received at 10Gbps (8Gbps of useful data) from each of the 16 payload slots. Data from each FADC250 is buffered into DDR3 memory where each modules has 256MBytes of buffering. This buffer is used to allow significant burst of occupancy and/or handle signficant downstream network delays without loosing any data. If high occupancy persists for too long that exceeds the network bandwidth then data will be dropped by the VTP and details on this will be discussed below.

FADC/VTP Streaming Data Path
Currently, the VTP streaming firmware implements 2 parallel instances of the FADC250 streaming system. Each instance handles 8 slots of FADC250, and has a dedicated 2GByte DDR3 buffer and 10Gbps Ethernet link. A future plan is to make 4 parallel instances in the firmware so that the all 4 lanes of the QSFP are usable for streaming each at 10Gbps (an additional 16 optical links can be used as well, but are limited to 8.5Gbps or if the speed grade of the FPGA is increased then 10Gbps is also achievable). Ethernet was chosen for the streaming readout interface because of its widespread support/compatibility. It is easy to buy additional commercial switches to scale the network as needed. We also already have significant amounts of VTP and VXS crates on site at Jlab, so we can take advantage of the available resources and run standard TCP/IP on the ethernet link to ensure simple and reliable data transfer to nearly any PC.

TCP/IP Interface
The VTP is programmed with the destination IP address and socket for where the 8 FADC slots of streaming data are sent. We can consider supporting additional destinations/sockets if needed to distribute a lesser load to servers, but as computing power is increasing it does appear that a single socket/server per 10Gbps is feasible for data transfers. A hardware accelerated TCP/IP stack implemented in VHDL is used to achieve deterministic and and throughput on the network. The TCP transmit buffer is limited to 64kBytes and have been able to achieve ~8Gbps throughput when streaming to Linux based servers.

Data formatting
Data is built into a TCP data frames that contain a header (containing frame number, timestamps, and other information helpful in ensuring data coherency and timing synchronization) and a payload (the customizable information by the streaming DAQ, currently the FADC hit data). The TCP frames correspond to a programmable span of time, typically 65536ns, for which the reported FADC hits have been collected. When the DAQ starts, the VTP modules also synchronously start their timers that are used to timestamp FADC hits. Each time a frame time has elapsed a TCP data frame is sent containing the hits for that time span. The ‘C’ structure for the data frame is as follows:

```
typedef struct stream_buffer
{
      uint32_t source_id;
      uint32_t total_length;
      uint32_t payload_length;
      uint32_t compressed_length;
      uint32_t magic;
      uint32_t format_version;
      uint32_t flags;
      uint64_t record_counter;
      struct timespec timestamp;
      uint32_t payload[];
} stream_buffer_t;
```

source_id: definable by software on each VTP, intended as a unique identifier
total_length: length (in bytes) of stream_buffer_t (including variable length payload, and excluding source_id)
payload_length: length (in bytes) of payload[] element. Can be zero.

compressed_length: not used, and currently set to payload_length
magic: should always be set to 0xC0DA2019
format_version: used to specify version of this data format (no official versions have been released yet, so kind of meaningless for now)
flags: no flags have been defined, meaningless for now
payload: variable length (0 to payload_length/4-1), front-end defined readout data

Main features are the timestamp (a 1ns resolution absolute timestamp for the FADC hit data frame). This timestamp should be added to the local timestamps of the FADC data hits. The record_counter indicates the frame number and will be received sequentially – if a frame is dropped due to overflowing data the record_counter will show a jump/gap.

The payload[] data should be parsed by looking for a data type (indicated by bit31=1 and type in the bits 30:15). The data type indicates the data to follow. The payload[] element contains the variable length streaming data (e.g. FADC hits). In addition to the FADC hits (data type = 1) there exists an FADC hit pointer structure (data type = 0) that is used to allow the receiving software to know where each FADC slot data exists in the payload and how long it is – the eliminates the need for the receiving end to parse the full payload before distributing parts to different processes.

The FADC pointer structure is as shown below.

```
typedef struct fadc_ptr
{
      uint32_t type;  // set to 0x80000000, bits 30:15=0 (FADC pointer
type)
      uint32_t ptr_len[8];
} fadc_ptr_t;
```

The array of 8 ptr_len elements correspond to the 8 FADC slots reported on this streaming socket. The 32bit ptr_len contains the offset for the data in payload[] in bits 15:0, and the length is contained in bits 30:16.

The FADC hit structure is as shown below.

```
typedef struct fadc_hit
{
  uint32_t type;
  uint32_t hit[];
} fadc_hit_t;
```

```
FADC hit type will have the bit assignment: 31=1, bits 30:15=1 (FADC
Hit type), 14:8=roc_id, 4:0=fadc slot number. Then a variable number
FADC hits will follow and have the following format: bit 31=0, bit
30:17=4ns timestamp, 16:13=channel, 12:0=charge
```

Data loss handling

For the moment the VTP is responsible for dropping streaming data when downstream pipes can't accept a higher input rate for long periods of time. Burst conditions (on the order of ~100msec at 32MHz/channel for all channels) are handled without data loss by the VTP DDR3 memory buffers. When the buffers are full, entire frames (e.g. 65536ns chunks of data) are dropped. These losses should not happen under normal conditions when the network and processing are sufficient

to keep up with the readout data (or if thresholds are too low) - the record_counter in the CODA SRO header can be used to identify dropped data.

7. VTP CODA Software ROC

7.1 Description

The VTP Zynq ARM processor is capable of running the traditional CODA ROC software components. The reason the VTP needs a ROC is mostly for diagnostic event data recording. The VTP performs trigger processing in many applications where some involve making complex decisions where the efficiency & correctness of these decisions needs continuous verification and monitoring throughout experiments. The software ROC is capable of running over 70kHz with event blocking and minimal deadtime. The VTP receives TI event block data using a proprietary 1Gbps serial link. This same serial link provides information on the event blocking level, sync event status, and block acknowledgement. TI and VTP generated event data can be read using single 32bit reads over the AXI bus, or use the AXIS DMA engine to achieve high performance. Readout is sent to the CODA event builder using the Linux OS controlled 1Gbps Ethernet link.

8. VTP CODA hardware accelerated ROC

8.1 Description

The VTP has a serial link to each VXS payload module primarily used for trigger data, but the link speeds are programmable and offer additional bandwidth that can be utilized as an alternate path for event data readout (instead of using the VME bus). The CODA ROC runs partially in software (initialization, asynchronous event injection, and low data rate synchronous event generation), while the large bandwidth event building and transport to the CODA EB are done using the FPGA (including a fully firmware based 10Gbps TCP/IP Ethernet solution).

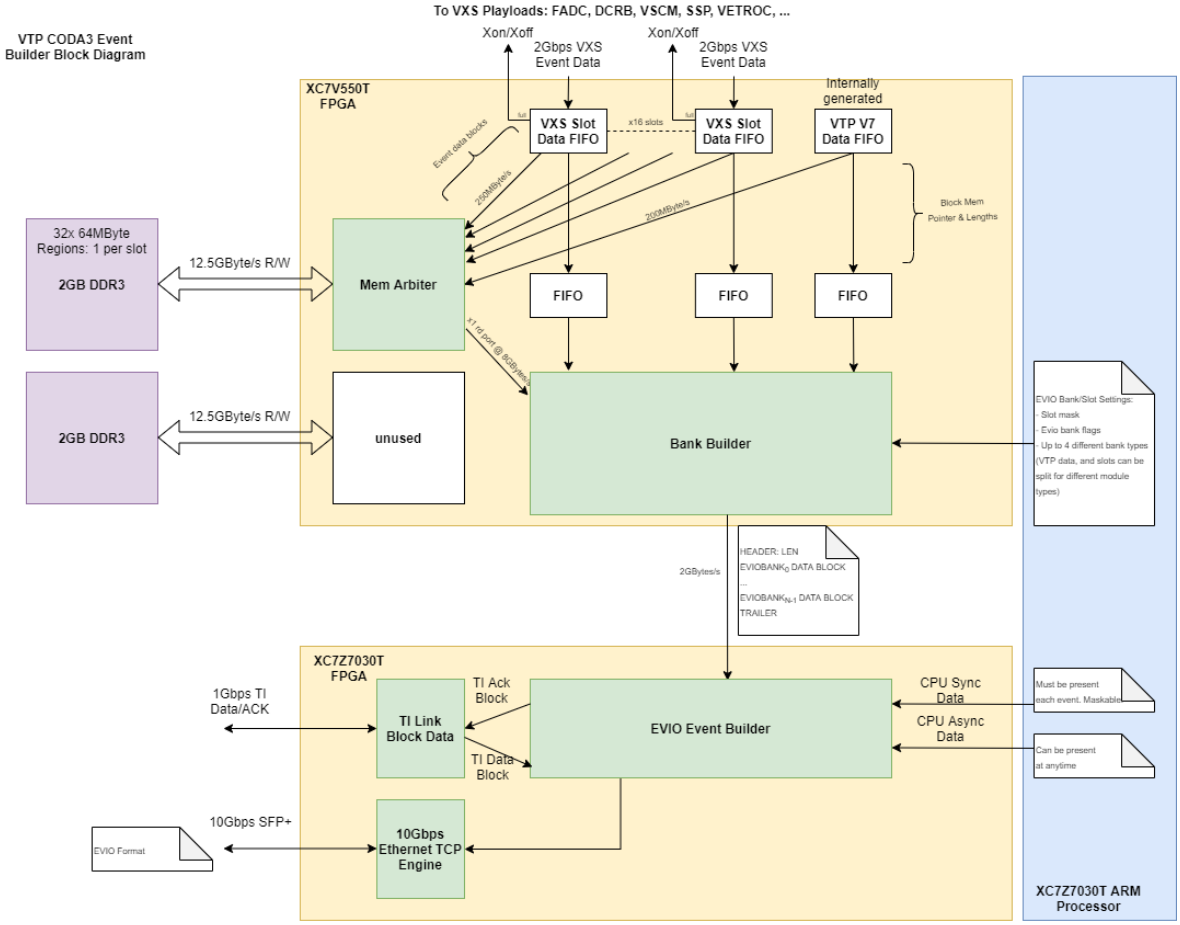
The primary benefits are:
1) payload modules can be readout in parallel (where on VME they were readout sequentially)

2) readout bandwidth from payload modules to the VTP is guaranteed and not shared

3) the VTP readout bandwidth to the CODA event builder is close to 10Gbps (and upgradable to 40Gbps in a future firmware)

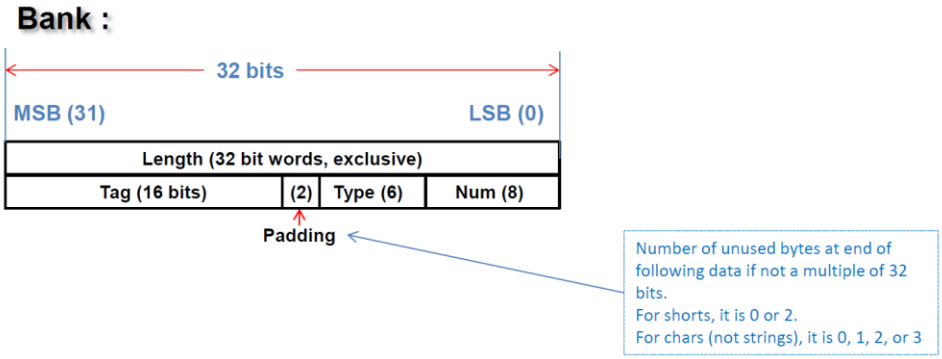
4) setups that already have a VTP installed for triggering purposes can receive this accelerated readout option is basically at no cost (for setups without a VTP, the cost is only the VTP which is often much cheaper than replacing the full front-end hardware)

The following diagram illustrates the hardware/firmware resources used:

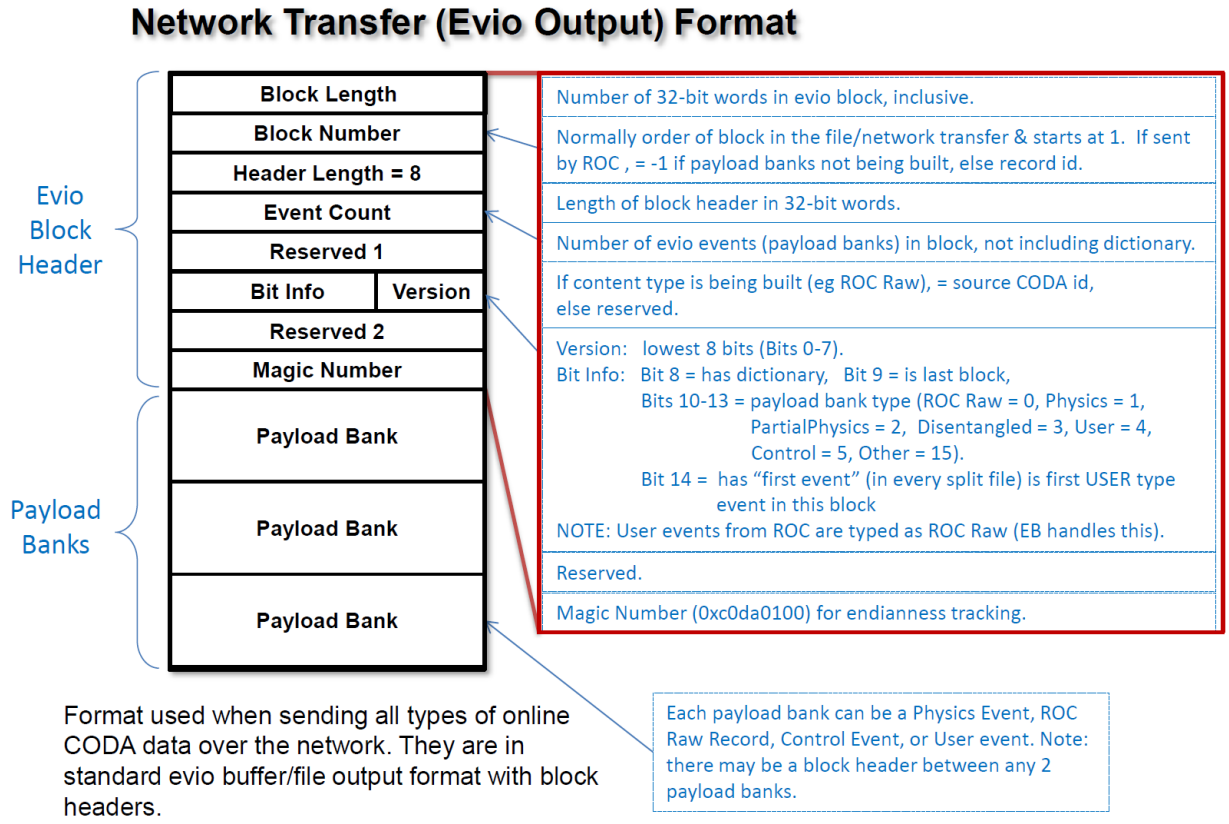


Serialized event data will be separated from the trigger input streams and buffered into one of the 2GByte DDR3 memories. The bandwidth of this memory is ~10GByte/s so is plenty capable of sustaining the traffic to saturate the 10Gbps Ethernet link. Future implementations can take advantage of multiple 10Gbps links or possibly use a single 40Gbps Ethernet link. Event data from the payload modules are buffered into

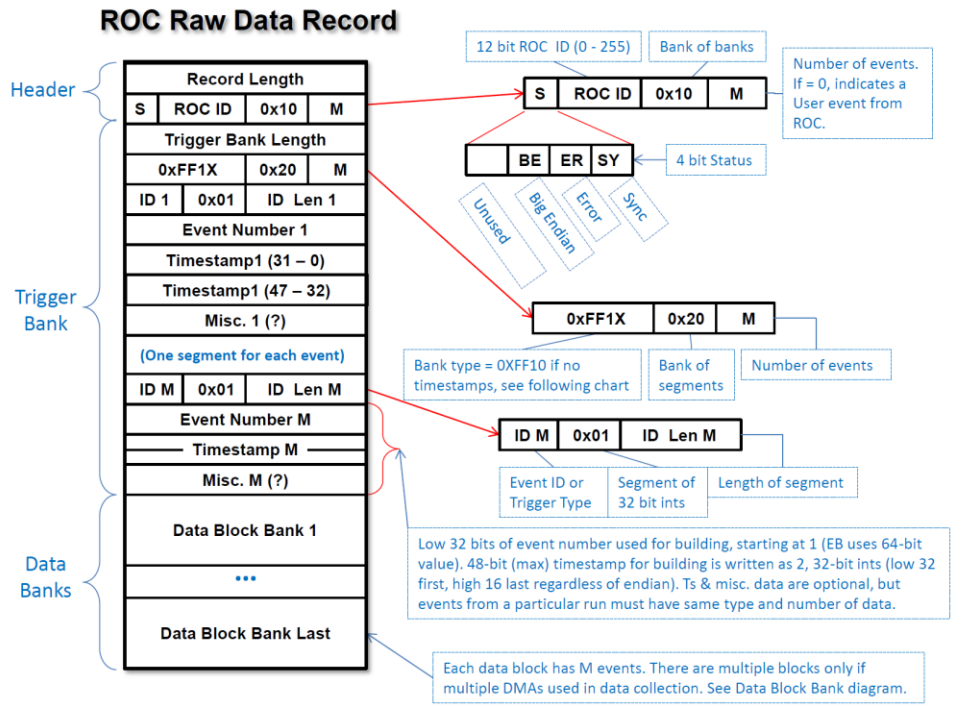
the DDR3 memory. The Virtex 7 FPGA can arrange the payload module data from each module to be assembled into 1 of 3 definable EVIO banks. All 16 payload modules can go into a single EVIO bank, or they can be split up into any of the 3 banks. This is useful in the case where multiple different module types exist in the crate and if it is desired their data lives in different EVIO banks. The Virtex 7 builds a single EVIO bank and computes the length, which contains up to 3 EVIO banks containing the payload data (so we're builing an EVIO bank of banks). The structure used is as follows, with type = 0x10 for the main EVIO bank, and the 1 to 3 banks inside use type = 0x1 for 32bit unsigned types):



This Virtex 7 EVIO bank header information is presented to the Zynq 7 FPGA where the final EVIO structure is built intended to communicate directly with CODA event builders. This final format adheres to the following structure:



The Zynq 7 FPGA will assemble the Payload Bank which contains the TI block data, and optionally an ARM CPU generated Data Block, and the Virtex 7 Data Block following this format:

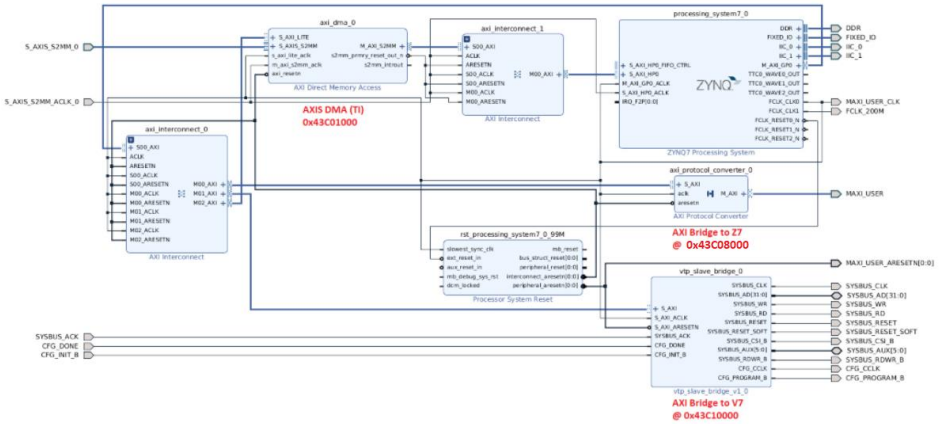


The VTP HW CODA ROC will only build physics events in firmware. CODA control events that are not physics events must use the VTP ARM Processor asynchronous event data interface. This asynchronous interface can be used to generate the required control events at the beginning and end of runs (but it can also be used to send asynchronous event data in the middle of ongoing runs for things like scalers, config data, or whatever else may be useful to record).

8.2 ARM CPU Peripheral Configuration

The ARM CPU/FPGA peripherals are as follows (and will be made standard on all VTP CODA ROC firmware versions):

1) AXIS DMA (TI): allows DMA transfer of TI event blocks into ARM system memory
https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf
2) AXI Bridge to Z7: allows CPU AXI-lite transactions to Zynq FPGA peripherals
3) AXI Bridge to V7: allows CPU AXI-line transactions to Virtex 7 FPGA peripherals (and also the same bus used to load the Virtex 7 FPGA firmware)



8.3 Zynq 7 FPGA Peripheral Configuration

Zynq 7 FPGA peripherals exist on the “AXI Bridge to Z7” component. The peripheral address offsets are described here (and are relative to the “AXI Bridge to Z7”):
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_z7/vtp_z7_pkg.vhd

Peripherals used on the VTP CODA ROC are as follows (and will be made standard on all VTP CODA ROC firmware versions):

PER_ID_TI
1Gbps link between VTP and TI used to transfer block of TI event data and control/acknowledge signals between these modules.
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_z7/z7_ti_per/z7_ti_per.vhd

PER_ID_CLK
Z7 FPGA clocks, resets, firmware version & timestamp.
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_z7/z7clk_per/z7clk_per.vhd

PER_ID_CODA_ROC
HW CODA ROC, also allows asynchronous and synchronous event data injection.
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_vxs_readout_z7/coda_roc_per/coda_roc_per.vhd

PER_ID_10GBE_TCPIP_CLIENT0
HW 10Gbps TCP/IP Ethernet stack. The VTP CODA ROC uses this interface as a TCP client (e.g. it makes the connection to the TCP server)
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_z7/z7_10GbE_tcpip_per/z7_10GbE_tcpip_per.vhd

PER_ID_EBIORX0
V7->Z7 16Gbps data bus receiver where the V7 FPGA CODA EVIO bank is received (which contains data from the VXS payload modules)
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_vxs_readout_z7/ebio_rx_per/ebio_rx_per.vhd

8.4 Virtex 7 FPGA Peripheral Configuration for FADC readout

Virtex 7 FPGA peripherals exist on the "AXI Bridge to V7" component. The peripheral address offsets are described here (and are relative to the "AXI Bridge to V7"):
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_v7/vtp_v7_pkg.vhd

PER_ID_CLK
V7 FPGA clocks, resets, firmware version & timestamp.
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_v7/v7clk_per/v7clk_per.vhd

PER_ID_QSFP0..3
QSFP transceiver and data links
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_v7/v7aurora8b10b_per/v7aurora8b10b_per.vhd

PER_ID_VXS0..15
VXS transceiver data links. Actual payload ID is +1 to the peripheral index.
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_v7/v7aurora8b10b_frm_per/v7aurora8b10b_frm_per.vhd

PER_ID_SD
Signal distribution management peripheral.
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_v7/v7sd_per/v7sd_per.vhd

PER_ID_EVT_BUILDER
V7 payload module EVIO bank configuration
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_vxs_readout_v7/vtp_evt_builder_per/vtp_evt_builder_per.vhd

PER_ID_MIGPER_R
V7 DDR3 memory configuration
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_v7/v7mig_per/v7mig_per.vhd

PER_ID_EBIO_TX0

V7->Z7 16Gbps data bus transmitter where the V7 FPGA CODA EVIO bank is received (which contains data from the VXS payload modules)

https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_vxs_readout_v7/ebio_per/ebio_tx_per.vhd

8.5 Virtex 7 FPGA Peripheral Configuration for MPD readout

MPD readout is very similar to the standard FADC readout except that each payload port may have up to 4 MPD connections. Currently only 32 MPD may be readout using VME slots 3-10 (this may be expanded in the future if needed). The MPD also has a unique processing block to perform zero suppression, common-mode subtraction, and remote register access so that MPD readout will be done using a dedicated V7 FPGA image/type. Virtex 7 FPGA peripherals exist on the "AXI Bridge to V7" component. The peripheral address offsets are described here (and are relative to the "AXI Bridge to V7"):

https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_v7/vtp_v7_pkg.vhd

PER_ID_CLK

V7 FPGA clocks, resets, firmware version & timestamp.

https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_v7/v7clk_per/v7clk_per.vhd

PER_ID_MPDFIBER0..31

MPD fiber control and processing interface. Fiber 0,1,2,3 correspond to the serial lanes VME slot 3. Fiber 4,5,6,7 correspond to the serial lanes of VME slot 4, etc. The VME slots are requires to have a VXS to QSFP payload module that converts the serial link to optics that interface the MPD.

https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_mpd_readout_v7/mpd_fiber_per/mpd_fiber_per.vhd

PER_ID_MPDREGS

This peripheral is used to access registers on the remote MPD.

https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_mpd_readout_v7/mpd_register_per/mpd_register_per.vhd

PER_ID_SD

Signal distribution management peripheral.

https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_v7/v7sd_per/v7sd_per.vhd

PER_ID_EVT_BUILDER

V7 payload module EVIO bank configuration

https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_vxs_readout_v7/vtp_evt_builder_per/vtp_evt_builder_per.vhd

PER_ID_MIGPER_R

V7 DDR3 memory configuration

https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_v7/v7mig_per/v7mig_per.vhd

PER_ID_EBIO_TX0

V7->Z7 16Gbps data bus transmitter where the V7 FPGA CODA EVIO bank is received (which contains data from the VXS payload modules)

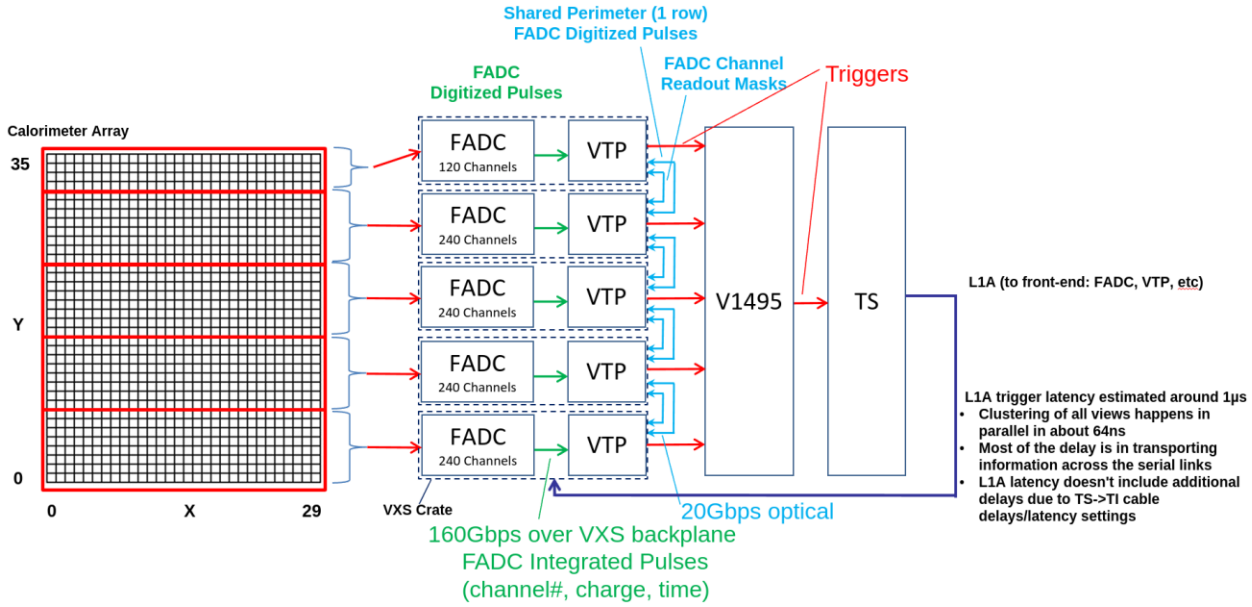
https://github.com/JeffersonLab/fe_fw/blob/devel/Firmware/Source/vtp_vxs_readout_v7/ebio_per/ebio_tx_per.vhd

9. NPS Cluster Trigger

9.1 Description

The NPS calorimeter is comprised of an array of 30x36 $PbWO_4$ blocks with a PMT and analog amplifier for each block. The amplifiers connect to the Jlab FADC250 modules for creating a trigger and readout. The purpose of the NPS cluster trigger is to find clusters in the calorimeter with high efficiency and minimal bias. When a cluster is found over threshold a trigger signal will be sent to the counting where it can trigger the DAQ directly or be used in coincidence with other detectors.

The segmentation of the calorimeter into VXS crates & FADCs was done such that each crate looked the same: same number of channels and the local mapping/view of the calorimeter was the same. This allows the clustering firmware in the VTP to be the same in all crates and only a Y coordinate offset is needed to distinguish each crate. The top of the calorimeter also works this way even though it has fewer channels. The following figure illustrates the segmentation and DAQ connections:



Another requirement of the NPS cluster trigger is to support sparsification of the FADC250 readout channels and only readout FADC channels centered on found clusters. This allows the FADC250 readout to report the full raw waveforms for event data, which is data volume intensive, but the sparsification will reduce the number of reporting channels from 1080 to a much smaller fraction. This significantly reduces the data volume and at the same time allows readout of waveforms for very small signals around the cluster center without having to apply an individual channel threshold online.

9.2 Trigger data flow

9.2.1 FADC pulse detection

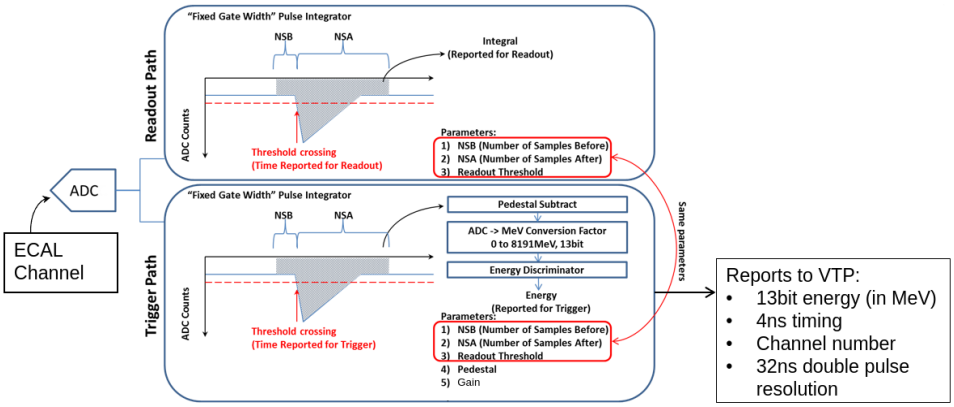
The trigger processing begins in the FADC250 with identifying and measuring the total charge of individual pulses on each channel. A software configurable threshold, **FADC250_TET**, which sets a common threshold for one or more FADC250 modules. These units are in FADC units: for NPS the FADC250 are in 1V dynamic range mode, so each FADC sample unit is 1V/4096 = 244µV. The threshold is relative to the software defined pedestal, **FADC250_ALLCH_PED**, which is a floating-point number specifying how many FADC units should be removed from each sample when performing a pulse integral. When the FADC input samples see a threshold crossing (previous sample <= PED+TET, next sample > TET) the pulse sum is formed around the threshold crossing: NSB (number of samples before the threshold crossing) and NSA (number of samples after the threshold crossing) samples are summed to compute the pulse integral. Next the pedestal is subtracted, and the gain is applied. The resulting pulse integral can be summarized by (where Sample[] is the array of FADC samples and N is the sample where the threshold crossing occurs):

$$PulseIntegral = GAIN \cdot \sum_{n=N-NSB}^{N+NSA-1} (Sample(n) - PED)$$

Integrated pulses can overlap, the requirements needed to ensure a pulse integration occurs is:

1. A threshold crossing is seen
2. No threshold crossing occurred in the past 7 clock cycles (last 28ns)

The second requirement is needed to ensure no more than 1 pulse every 32ns can be reported (this is a bandwidth limit on the communication link from FADC -> VTP). It also helps to suppress multiple threshold crossings due to pulse ringing or noise. This also means the FADC trigger path can miss a legitimate pulse if too close to a previous one. The pulse integration is summarized by the lower "Trigger Path" part of the following figure:

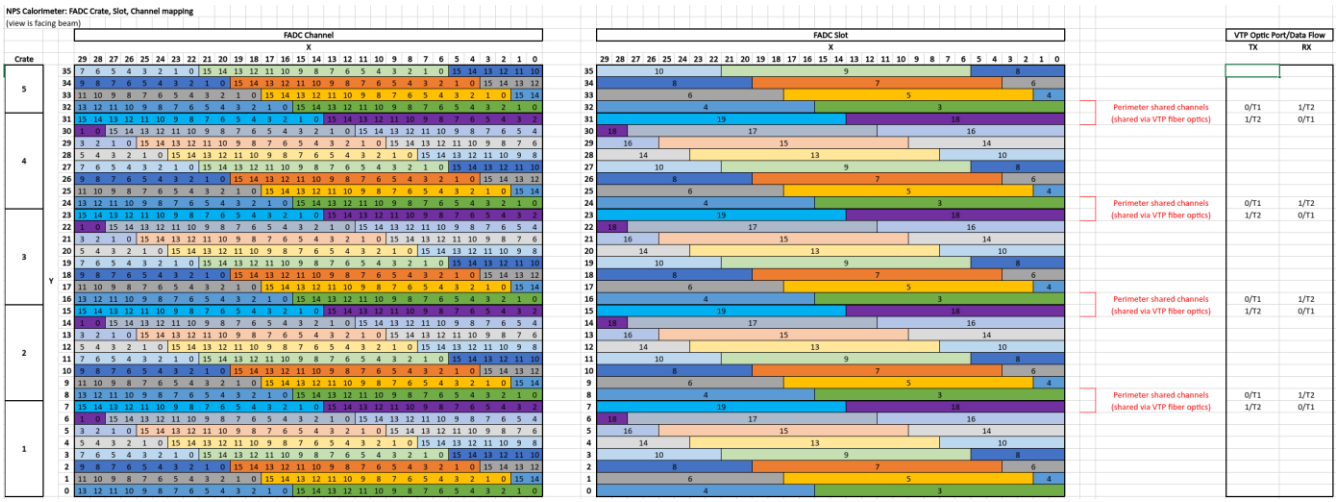


The resulting pulse integral must also be constrained to 13bits due to bandwidth limitations. So it is important that the FADC GAIN parameter is set so that the desired pulse integral dynamic range is scaled to fit in the 0 to 8191 range set by the 13bit limitation. Normally the GAIN is setup such that the resulting gained integral units are in MeV. After applying the gain, any values less than 0 saturate at 0 and any values greater than 8191 saturates at 8191.

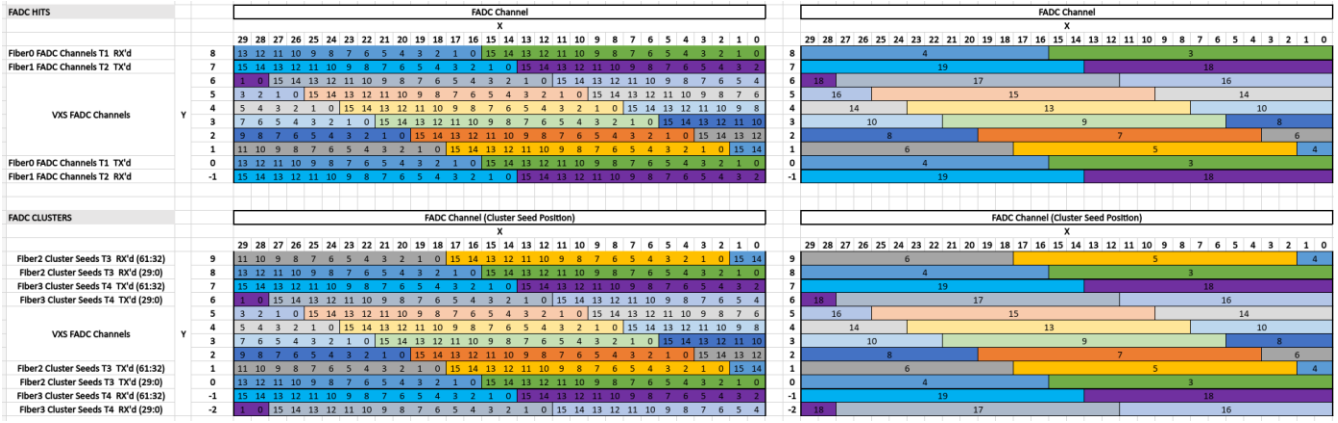
9.2.2 Streaming FADC hits to VTP

Each FADC250 communicates with the VTP using 4 full duplex dedicated links per FADC250. Each link runs at 3.125Gbps with 8b10b encoding. This results in 10Gbps of bandwidth availble between each FADC250 to the VTP. The Xilinx Aurora framing protocol is used as a lightweight interface, which defines all the necessary control characters, synchronization, lane-bonding, and framing parameters to make it easy to develop on top of. The FADC sees a 64bit data path running at 156.25MHz and it must fit 31.25MHz * 16 channels * 16bits of pulse integral/timing information. The necessary bandwidth for this trigger path is 8Gbps, where the FADC can effectively send 13bit energy, 3bit timestamp for 16 channels every 32ns. The 3bit timestamp is needed to identify the pulse time to 4ns resolution within the 32ns window the data is sent. The VTP receives this stream and decodes it such it sees FADC channel pulse integrals for all 256 channels with 4ns timing resolution – the generic basis from which a variety of applications can take this data to build custom algorithms.

Clustering also has to work seamlessly across the VXS crate boundaries. To do this the FADC pulses are the crate borders are exchanged with the adjacent crate. The segmentation of the calorimeter channels was done to ensure the bordering channels were able to fit on a single VTP fiber link (which runs at 20Gbps 8b10b, so 16Gbps usable – exactly enough to send 32 FADC channels over 1 link). These links are bi-directional so a crate sending FADC pulses to the adjacent crate will also receive from that crate so that same crate. The FADC250 channel/slot map is shown along with the channels shared and what VTP fibers are used to do that (shown all the way on the right):



As mentioned before the firmware for each of the VTPs is the same so that locally they see the same channel map (the only parameter the VTPs need set is the unique Y offset set by software). It is as follows:



The final resulting FADC pulse stream seen by each VTP consist of the local FADC250 pulses reported over the VXS backplane and the remote FADC250 pulses received over fiber shared from adjacent VTPs. The data from these different sources show up at different times due to different path delays. To remove this skew, the VTP is configured to know which FADC and Fiber ports are active for the trigger processing. Once the TS releases SYNC, the FADC250 stream integrated pulses to the VTPs, the latency is around 300ns for a small NSA parameter (e.g. NSA ~36ns). The VTP forwards certain FADC channels over fiber to the adjacent VTPs. The FADC pulses are all stored in a FIFO that are not read until all enabled FADC and fiber sources have data ready. Once all FIFOs have data ready the VTP can begin reading and this provides the full FADC pulse information that is aligned in time that the trigger logic can begin to process. The overall latency of this is about 1800ns (time for data from FADC to be read by VTP trigger logic that is aligned in time across all slots & fibers).

9.3 Clustering logic

Clustering is performed on FADC pulses in both space and time using all possible overlapping 3x3 views of the calorimeter. This results in 1080 possible views and is how many cluster processing units running in parallel to quickly perform this task (in about 96ns). A single cluster processing unit needs to see only the 3x3 view of FADC channels it will cluster. Within the cluster processor, a cluster will be found/formed using the following rules:

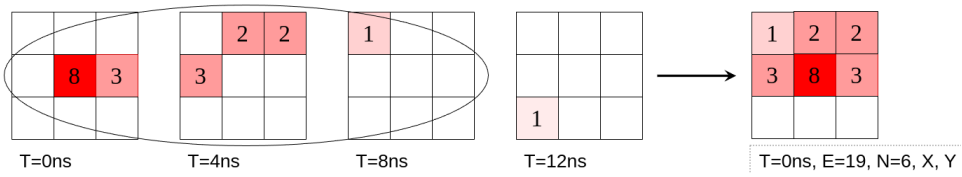
a. Central block has energy >= software defined cluster seed threshold defined by the config **VTP_NPS_ECALCLUSTER_SEED_THR**

b. Central block energy is >= all immediate surrounding block energies in 3x3 view (i.e. it is a local maximum) that are withing the timing coincidence window defined by **VTP_NPS_ECALCLUSTER_HIT_DT**

c. The cluster sum is formed by adding up the pulse energy of the central crystal along with the largest pulse energy in the timing coincidence window (defined by **VTP_NPS_ECALCLUSTER_HIT_DT**) from each of the 8 channels surrounding the central block

d. The cluster seed position is the reported (X,Y) position and time of the cluster, along with the cluster sum, and the number of channels that had a hit in the 3x3 window

The following figure shows a single 3x3 cluster processor view and the cluster it forms given a seed threshold of 2 and hit_dt = +/-8ns



Finding clusters in both space and time prevent multiple clusters from being found from adjacent 3x3 views so that 1 cluster is reported for each physical cluster in the calorimeter (exceptions to this can be if the channels have poorly calibrated gains, thresholds, pedestals, timing offsets, noise, and/or cable reflections).

All of these found clusters are send to various trigger processing stages (cluster singles and cluster pair processing). These clusters are stored in an 8µs, similar to FADC samples, so that when the system is triggered the VTP will record all found clusters in its readout window (useful for verification and monitoring clusters that the FADC may not readout due to sparisfication cuts).

9.4 Trigger Types

The following sections will describe the conditions needed to meet the trigger bit requirements that generate the VTP trigger bit pulses. See section "VTP Trigger Bits & V1495 Trigger Bits" for trigger bit mapping details.

9.4.1 Cluster singles trigger

Cluster singles trigger is the primary NPS production data trigger. The conditions to create a trigger are as follows (where a single cluster must satisfy all conditions):

Cluster.Energy >= **VTP_NPS_ECALCLUSTER_CLUSTER_TRIGGER_THR**
Cluster.NHits >= **VTP_NPS_ECALCLUSTER_NHIT_MIN**

When the condition is satisfied the trigger time is a based on the Cluster.Time, which is the central hit/seed hit time. It is delayed by the software programmed trigger latency parameter:

VTP_NPS_TRIG_LATENCY

9.4.2 Cluster pair trigger

The cluster pair trigger was requested and added late in the design and the original architecture wasn't setup for a global multiplicity or pair processing (e.g. cluster sum cuts). Individual clusters must meet the following condition (note it shares the same config parameter for NHits cut, but a unique on for the cluster threshold):

Cluster.Energy >= **VTP_NPS_ECALCLUSTER_CLUSTER_PAIR_TRIGGER_THR**
Cluster.NHits >= **VTP_NPS_ECALCLUSTER_NHIT_MIN**

Local PAIR1 trigger logic

Cluster that past the above conditions will generate a "PAIR1" trigger pulse trigger bit – this trigger signal there is at least 1 cluster in the local crate that satisfies the pair cluster condition. The V1495 will generate a PAIR trigger if it finds 2 or more VTPs with a "PAIR1" trigger asserted at the same time. The coincidence width the V1495 sees for PAIR1 trigger bit inputs is defined by the VTP trigger bit pulse width **VTP_NPS_TRIG_WIDTH**. The V1495 will generate a 100ns trigger pulse to send to the trigger supervisor (when two or more PAIR1 VTP trigger bits are asserted).

Local PAIR2 trigger logic

A separate trigger bit PAIR2 is generated by each VTP that is asserted when two or more clusters are found in the local crate that both satisfy the above Cluster.Nhits and Cluster.Energy cuts used for the pair trigger. The coincidence window is programmable by the parameter **VTP_NPS_ECALCLUSTER_CLUSTER_PAIR_TRIGGER_WIDTH**. The trigger bit is sent to the V1495 using the trigger bit pulse width define by **VTP_NPS_TRIG_WIDTH**. The V1495 OR's all PAIR2 trigger bits from the VTPs together and this is sent to the trigger superviser.

- Note: the pairs found in the V1495 generate a trigger pulse on the same output that the PAIR2 triggers are OR'd onto. So only one pair trigger bit is provided to the counting house trigger supervisor.

9.4.3 Cosmic trigger - scintillator

There are 4 scintillator bars connected to nps-vme3 FADC250 slot 20:

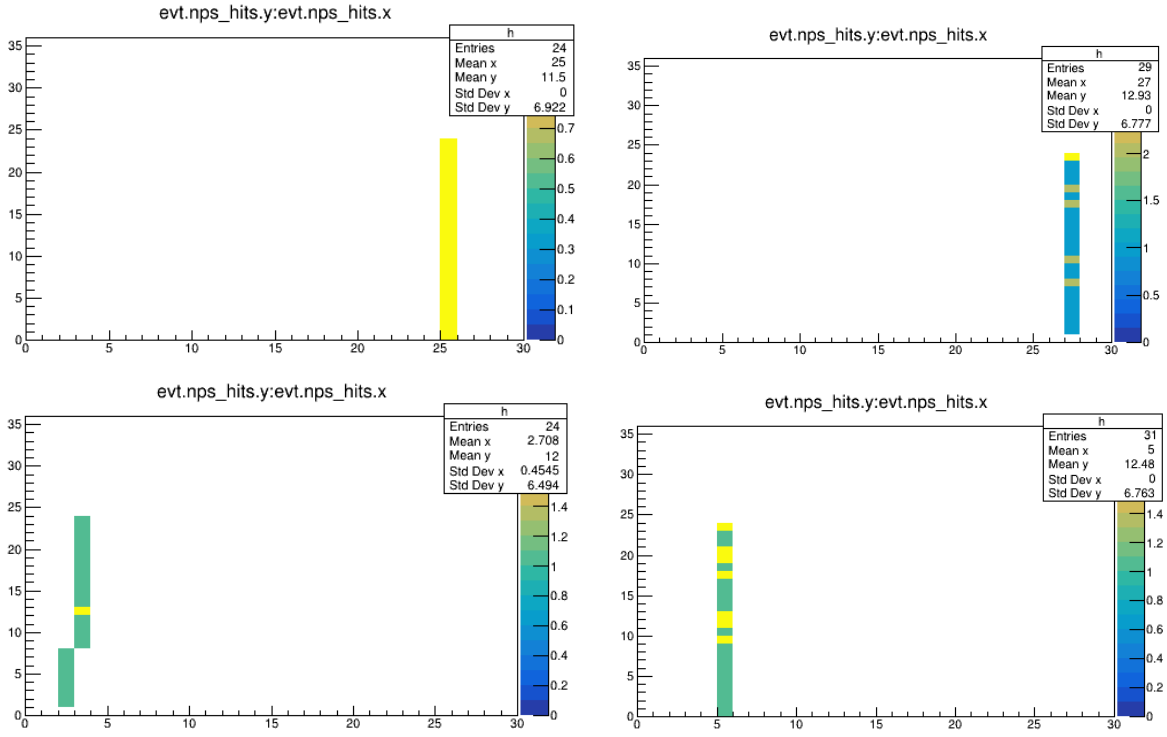
Scintillator top bar 0: FADC slot 20, channel 0
Scintillator top bar 1: FADC slot 20, channel 1
Scintillator bottom bar 0: FADC slot 20, channel 2
Scintillator bottom bar 1: FADC slot 20, channel 3

The cosmic trigger will trigger when any hit in either top scintillator is in timing coincidence with any hit in either bottom scintillator. The trigger time will be timed with respect to when both top and bottom are seen in coincidence. The coincidence time is programmable and determines the coincidence time, **VTP_NPS_COSMIC_SCINT_DT**. A hit on any of these scintillator channels is defined as any FADC250 threshold crossing on the scintillator that has a non-zero integrated energy (integration by FADC250 NSB & NSA, pedestal subtraction, and gain applied => pulse energy).

9.4.4 Cosmic trigger – calorimeter column

This calorimeter column trigger is satisfied by any NPS VTP when it sees FADC channels in a single column of its crate meet the software defined multiplicity in the timing coincidence window. The VTP doesn't apply any threshold cuts, but it does require a non-zero pulse that has crossed the FADC250 channel threshold to see a channel hit. Within each crate, a column is only 8 crystals tall. The multiplicity can be changed from 1 to 8 so conditions can be relaxed if inefficiencies are expected. In addition, there is a veto option that will reject any column trigger if a hit exists anywhere else in another column of the same crate – this is very helpful at rejecting corner clipping tracks. The V1495 can be setup to OR the crates or AND them. When enabling the veto and performing a crate AND at the V1495 almost all events are vertical tracks constrained to a single column of the whole calorimeter – so the event rate is low and purity very high.

FADC event examples (FADC channels with hits in event. A low threshold often causes channel to see multiple hits) from the testing in EEL with a 3 crate setup (high multiplicity, veto enabled, v1495 AND mode – column shifts may be seen at Y = multiple of 8 boundaries):



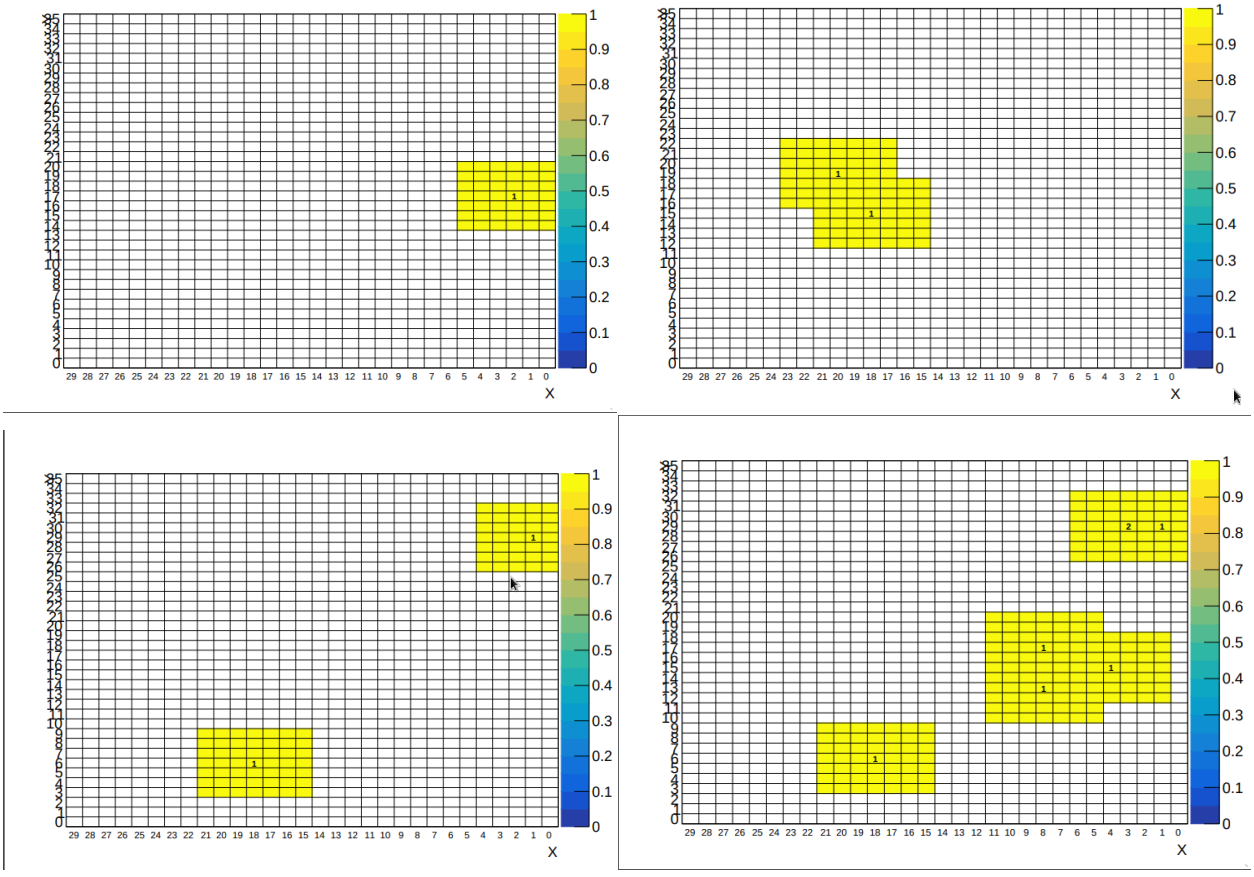
9.4.5 VLD trigger

The VLD trigger is used for LED trigger runs. The VLD modules have their trigger outputs all OR'd together in a daisy-chained connection with the last module being used as the trigger source for the VLD system. It connects to nps-vme3, FADC250 slot 20 (a spare FADC not used by NPS calorimeter), channel 15. Using the FADC250 as the trigger input allows the latency to closely match the NPS cluster trigger. This trigger is simple as it generates a trigger anytime a leading edge is seen on the FADC input. See section "VTP Trigger Bits & V1495 Trigger Bits" for trigger bit mapping details.

9.5 Sparisfication

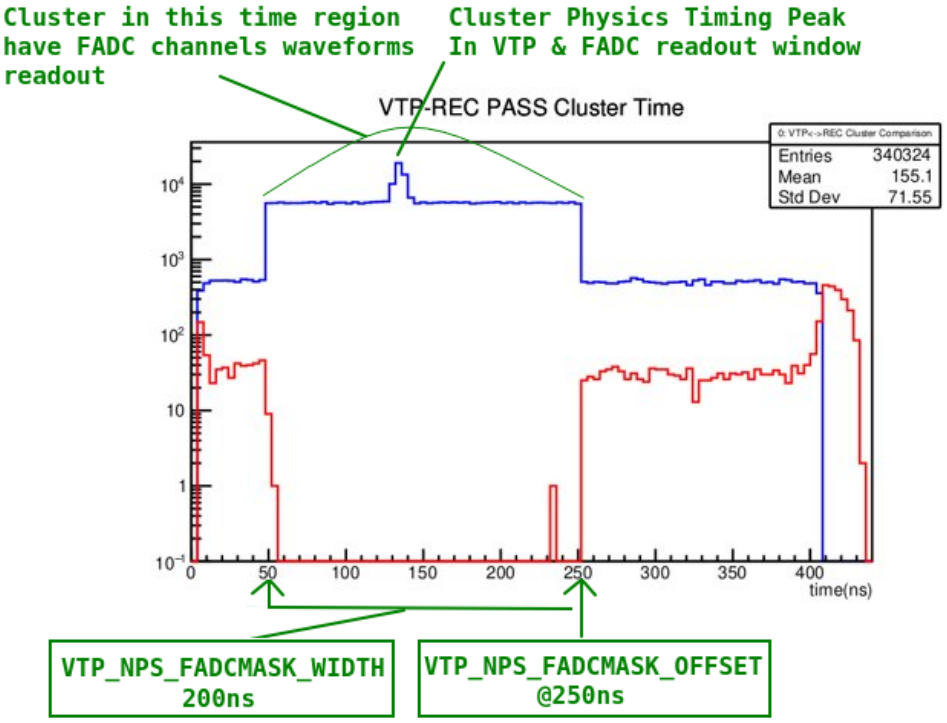
NPS calorimeter sparsification was done to significantly reduce the data rate while simultaneously supplying raw waveform readout for channels around the cluster (so no individual channel threshold is needed to readout the FADC channels of the cluster, potentially improving offline reconstruction resolution). Either a 5x5 or 7x7 channel pattern centered on the trigger cluster seed is used to determine which FADC channels to readout. The VTP uses another cluster energy threshold to determine what clusters will allow FADC channels to be readout: **VTP_NPS_ECALCLUSTER_CLUSTER_READOUT_THR**. When the DAQ issues a trigger, the VTP looks back in time to find clusters >= **VTP_NPS_ECALCLUSTER_CLUSTER_READOUT_THR** (a lookback time and window width is used so that trigger latency and jitter can be compensated for). Any clusters satisfying the readout cluster threshold will result in a 5x5 or 7x7 FADC channel pattern (selectable by software configuration) to be readout centered on the seed/central block of the clusters. The VTP sends the readout channels masks to all FADCs so they know which channels to keep and throw away data for on each event.

Some examples of this sparsified cluster readout can be shown by the following tests. The pictures show the calorimeter channel layout with yellow blocks indicating the FADC channels the readout sees raw waveforms for and the numbers show where and how many clusters the VTP found at that cluster center position. You can see a variety of examples showing 7x7 readout patterns for separate and overlapping cluster events:



A critically important detail is the timing diagram that illustrates the parameters to setup to ensure a trigger can properly be timed in to capture the VTP clusters to determine these channels patterns. The two main parameters to adjust this timing is the **VTP_NPS_FADCMASK_WIDTH** and **VTP_NPS_FADCMASK_OFFSET**. These

parameters decide which time range in the VTP readout window will be looked at to determine which clusters satisfy the readout cluster threshold. The “OFFSET” parameter determines at what point in time of the readout window the VTP will evaluate what clusters are evaluated – the example image below shows this offset at 250ns in the readout window (look at the blue curve, which is the FADC reconstructed cluster trigger times). Then the “WIDTH” parameter determines how much time before and up to the “OFFSET” the VTP will also look for clusters over the readout threshold – the example shows a width of 200ns. In this example you can see we consider clusters from 50ns to 250ns in the readout window, centered with plenty of margin around the physics peak at 150ns. When we readout the FADC channel, we readout the full waveform for the entire (which spans 400ns in the example) - this is why we also see a suppressed background outside the 50ns to 250ns range.



9.6 VTP Trigger bits & V1495 Trigger Bits

Each VTP outputs its trigger bits on the front panel ribbon cable (LVDS signals) and they are mapped as follows:

VTP Trigger Bit 0: Cluster Singles
VTP Trigger Bit 1: Cosmic Scintillator
VTP Trigger Bit 2: Cosmic Column
VTP Trigger Bit 3: Pair 1
VTP Trigger Bit 4: Pair 2
VTP Trigger Bit 5: VLD

9.6.1 Latency

The latency of these triggers (roughly the time from an input at the FADC250 to where the VTP produces an output on its trigger bit output) is defined by the programmable parameter:

VTP_NPS_TRIG_LATENCY

This latency must be at least as long as the total trigger path processing time to ensure deterministic behavior (as of the beginning of NPS runs, it has needed to be least 2.3μs). This parameter can be adjusted to bring it into coincidence with other detectors, but each trigger bit can be individually delayed with the following parameter: **VTP_NPS_TRIG_DELAY**.

9.6.2 Prescaling

Trigger bits can also be prescaled with the following parameter: **VTP_NPS_TRIG_PRESCALE**

9.6.3 Trigger width

A programmable trigger bit pulse width (common to all trigger bits) can also be set (**VTP_NPS_TRIG_WIDTH**). This width can be used to set the desired coincidence width. If the coincidence width is desired to be set outside the VTP, it is important to use a small width for the VTP if the outside logic is edge sensisitive: the VTP trigger bit widths work in "updating" mode – that is, they will extend the pulse width if another trigger happens while the trigger output is already active. An edge-sensitive logic outside the VTP can have efficiency losses due to pileup at the VTP trigger bit.

9.6.4 V1495

The V1495 is a programmable logic/FPGA unit from CAEN. It is used in NPS to consolidate the trigger bits from 5 VTPs to a few trigger cables that can run to the Hall C counting room. The V1495 accepts the VTP trigger bits over 5 different ribbon cables and it outputs the trigger bits to the Hall C counting house over coaxial cable using NIM drivers. The V1495 trigger bit outputs are as follows:

V1495.F1: T1 NPS calorimeter cluster trigger (logical OR of all VTP Trigger Bit 0)
V1495.F2: T2 NPS calorimeter scintillator coincidence (npsvtp3 only, VTP Trigger Bit 1)
V1495.F3: T3 NPS calorimeter cosmic column trigger OR (logic OR of all VTP Trigger bit 2)
V1495.F4: T4 NPS calorimeter cosmic column trigger AND (logic AND of all VTP Trigger bit 2)
V1495.F5: T5 VLD (npsvtp3 only, VTP Trigger Bit 5)
V1495.F6: T6 NPS calorimeter pair trigger (logical OR of all VTP Trigger Bit 4 or multiplicity >=2 of all VTP Trigger bit 3)

V1495 triggers T1 to T5 are simple asynchronous boolean functions – no pulse width or edge detection done, so the output widths are defined by the VTP trigger pulse widths. T6 is different because the V1495 needs to perform a multiplicity trigger and reform the trigger pulse width to ensure a minimum trigger pulse width is made so that it will arrive to the counting house over the long cable run. T6 is an OR of all "Pair 2" trigger bits – these are triggers from each VTP where 2 or more clusters satisfying the pair trigger are within that crate – so the VTP will definie the V1495 pulse width for these. "Pair 1" requires a multiplicity >=2 condition (2 or more VTPs have to assert this at the same time so we trigger from 2 or more clusters in different crates). The VTP trigger pulse width defines the coincidence width used by the pair 1 multiplicity logic. When the V1495 finds 2 or more Pair 1 trigger bits asserted at the same time, it will create a 100ns pulse width on the T6 output (it works in updating mode as well, so new pairs triggers will extend this pulse width if already active).

9.7 Event builder data types

Data will be reported from each nps-vtp (1-5). Each VTP processes different regions of the NPS calorimeter, so the data from all VTP should be combined to see the full event information. This data will be wrapped in an EVIO formatted data bank which is not described in this document. Refer to the CODA configuration for ROC ID assignments and the readout list for the EVIO bank types/tags used to encapsulate the following described data.

9.7.1 Data Word Categories

Data words from the module are divided into two categories: Data Type Defining (bit 31 = 1) and Data Type Continuation (bit 31 = 0). Data Type Defining words contain a 4-bit data type tag (bits 30 - 27) along with a type dependent data payload (bits 26 - 0). Data Type Continuation words provide additional data payload (bits 30 – 0) for the *last defined data type*. Continuation words permit data payloads to span multiple words and allow for efficient packing of various data types spanning multiple data words. Any number of Data Type Continuation words may follow a Data Type Defining word.

9.7.2 Data Type List

0	Block Header
1	Block Trailer
2	Event Header
3	Trigger Time
12	Expanded (Data SubType)
12.11	NPS Cluster
13	Trigger Decision
14	Data Not Valid (empty module)

15 Filler Word (non-data)

Data Type: Block Header

Type: 0
Size: 1 word
Description: Indicates the beginning of a block of events. (High-speed readout of a board or a set of boards is done in blocks of events)

31	30	29	28	27	26	25	24
1	0	0	0	0	SLOTID		
23	22	21	20	19	18	17	16
SLOTID		UNDEFINED				EVENT_PER_BLOCK	
15	14	13	12	11	10	9	8
EVENT_PER_BLOCK							
7	6	5	4	3	2	1	0
BLOCK_CNT							

BLOCK_CNT:
Event block number (used to align blocks when building events)
EVENT_PER_BLOCK:
Number of events in block
SLOTID:
Slot ID (set by VME64x backplane)

Data Type: Block Trailer

Type: 1
Size: 1 word
Description: Indicates the end of a block of events. The data words in a block are bracketed by the block header and trailer.

31	30	29	28	27	26	25	24
1	0	0	0	1	SLOTID		
23	22	21	20	19	18	17	16
SLOTID		NUM_WORDS					
15	14	13	12	11	10	9	8
NUM_WORDS							
7	6	5	4	3	2	1	0
NUM_WORDS							

NUM_WORDS:
Total number of words in block of events
SLOTID:
Slot ID (set by VME64x backplane)

Data Type: Event Header

Type: 2

Size: 1 word

Description: Indicates the start of an event. The included trigger number is useful to ensure proper alignment of event fragments when building events.

31	30	29	28	27	26	25	24
1	0	0	1	0	UNDEFINED		
23	22	21	20	19	18	17	16
UNDEFINED		TRIGGER_NUMBER					
15	14	13	12	11	10	9	8
TRIGGER_NUMBER							
7	6	5	4	3	2	1	0
TRIGGER_NUMBER							

TRIGGER_NUMBER:

Accepted event/trigger number

Data Type: Trigger Time

Type: 3

Size: 2 words

Description: Time of trigger occurrence relative to the most recent global reset. The time is measured by a 48bit counter that is clocked from the 250MHz system clock. The assertion of the global reset clears the counter. The de-assertion of global reset enables counter and thus sets t=0 for the module. The trigger time is necessary to ensure system synchronization and is useful in aligning event fragments when building events.

Word 1:

31	30	29	28	27	26	25	24
1	0	0	1	1	UNDEFINED		
23	22	21	20	19	18	17	16
TRIGGER_TIME_L							
15	14	13	12	11	10	9	8
TRIGGER_TIME_ L							
7	6	5	4	3	2	1	0
TRIGGER_TIME_ L							

TRIGGER_TIME_ L**:**

This is the lower 24bits of the trigger time

Word 2:

31	30	29	28	27	26	25	24
0	UNDEFINED						
23	22	21	20	19	18	17	16
TRIGGER_TIME_H							
15	14	13	12	11	10	9	8
TRIGGER_TIME_ H							
7	6	5	4	3	2	1	0
TRIGGER_TIME_ H							

TRIGGER_TIME_ H:

This is the upper 24bits of the trigger time

Data Type: NPS Cluster

Type: 12.11
Size: 2 words
Description: This data type identifies a cluster

Word 1:

31	30	29	28	27	26	25	24
1	1	1	0	0	1	0	1
23	22	21	20	19	18	17	16
1	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	E					
7	6	5	4	3	2	1	0
E							

E: 14bit unsigned cluster energy

Word 2:

31	30	29	28	27	26	25	24
0	-	-	-	-	-	Y	
23	22	21	20	19	18	17	16
Y				X			
15	14	13	12	11	10	9	8
X	N				T		
7	6	5	4	3	2	1	0
T							

X: 5bit unsigned cluster X coordinate
Y: 6bit unsigned cluster Y coordinate
T: 11bit cluster time in 4ns units referenced from the beginning of the readout window
N: 4bit number of hits in the cluster

Data Type: Trigger Decision

Type: 13
Size: 2 words
Description: This data type reports trigger decision made. A 32bit trigger bit pattern is reported with 4ns timestamp relative to the readout window indicates where the VTP found a valid trigger. If multiple triggers happen at the same time then multiple bits will be set in the 32bit trigger bit pattern word. A trigger decision pattern will be recorded for each unique time in the VTP readout window.

Word 1:

31	30	29	28	27	26	25	24
1	1	1	0	1	T		
23	22	21	20	19	18	17	16
T							
15	14	13	12	11	10	9	8
TRIGBITS_L							
7	6	5	4	3	2	1	0
TRIGBITS _L							

TRIGBITS_L: Trigger bits 15:0
T: 11bit trigger bit pattern time in 4ns units
(referenced from the beginning of the readout window)

Word 2:

31	30	29	28	27	26	25	24
0	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
TRIGBITS_H							
7	6	5	4	3	2	1	0
TRIGBITS_H							

TRIGBITS_H: Trigger bits 31:16

Note: For NPS, the following trigger bits have been defined:
TriggerBit0: NPS Cluster Trigger, cluster >= threshold
TriggerBit1: Cosmic scintillator trigger, or(ScintTop) and or(ScintBot)
TriggerBit2: Cosmic calorimeter column trigger, or(mult($Column_n$) > mult_min)
These trigger bits from each nps-vtp will be processed in an additional stage using a CAEN V1495 FPGA module so the mapping of these bits into the Trigger Supervisor will be different from above (this should be documented in the NPS DAQ trigger setup when implemented).

Data Type: Data Not Valid

Type: 14
Size: 1 word

Description: Module has no data available for readout. This can if the module is being read out too quickly after receiving (event building is in process and no data words have been put into the buffer yet) a trigger or if the module doesn't have any events to report.

31	30	29	28	27	26	25	24
1	1	1	1	0	UNDEFINED		
23	22	21	20	19	18	17	16
UNDEFINED							
15	14	13	12	11	10	9	8
UNDEFINED							
7	6	5	4	3	2	1	0
UNDEFINED							

Data Type: Filler Word

Type: 15

Size: 1 word

Description: Non-data word appended to the block of events. This is used to force the total number of 32-bit words read out of a module to be a multiple of 2 or 4 when

31	30	29	28	27	26	25	24
1	1	1	1	1	UNDEFINED		
23	22	21	20	19	18	17	16
UNDEFINED							
15	14	13	12	11	10	9	8
UNDEFINED							
7	6	5	4	3	2	1	0
UNDEFINED							

9.8 Configuration Parameters

VTP_W_WIDTH <width>

Sets the VTP readout window width in units of nanosecond ranging from 0 to 8188. VTP triggers and clusters in the readout window will be recorded for each VTP event.

VTP_W_OFFSET <offset>

Sets the VTP readout window offset (or lookback time) in units of nanosecond ranging from 0 to 8191. VTP triggers and clusters in the readout window will be recorded for each VTP event. Note this window parameter is also used by the sparisfication cluster detection logic (the sparsification offset is relative to this window offset).

VTP_PAYLOAD_EN <en1> <en2> <en3> <en4> <en5> <en6> <en7> <en8>
<en9> <en10> <en11> <en12> <en13> <en14> <en15><en16>

<en1> to <en16> can be 0 or 1. 0 indicates the VXS payload card is disabled, 1 indicates the VXS payload card is enabled. These payloads are the FADC payload trigger enable/disable flags and must be setup according to the used FADC slots used in the trigger. The following table shows the map of the payload number to VME slot:

VXS Payload	VME Slot
1	10
2	13
3	9
4	14
5	8
6	15
7	7
8	16
9	6
10	17
11	5
12	18
13	4
14	19
15	4
16	20

VTP_FIBER_EN <en0> <en1> <en2> <en3>

<en0> to <en3> can be 0 or 1. 0 indicates the fiber is disabled, 1 indicates the fiber is enabled. These parameters are set to ensure fibers between the nps-vtp crates are enabled to share border FADC hits so clustering at crate perimeters works and also to share found cluster information so that sparsification channel patterns work. The following table illustrates the used fiber ports:

VTP Optical Connection Table

From:		To:	
nps-vtp#	T#	nps-vtp#	T#
5	4	unused	unused
5	3	4	4
5	2	unused	unused
5	1	4	2
4	4	5	3
4	3	3	4
4	2	5	1
4	1	3	2
3	4	4	3
3	3	2	4
3	2	4	1
3	1	2	2
2	4	3	3
2	3	1	4
2	2	3	1
2	1	1	2
1	4	2	3
1	3	unused	unused
1	2	2	1
1	1	unused	unused

VTP_NPS_COSMIC_SCINT_DT <dt>

<dt> is the scintillator hit coincidence time. Valid values for <dt> range from 0 to 7 and units are in 4ns.

VTP_NPS_COSMIC_COLUMN_MULTMIN <mult>

<mult> is the minimum multiplicity of blocks in a single crate and calorimeter column to have a hit to create a cosmic column trigger for that crate. Valid values for <mult> range from 1 to 8.

VTP_NPS_COSMIC_COLUMN_DT <dt>

<dt> is the column hit coincidence time. Valid values for <dt> range from 0 to 7 and units are in 4ns.

VTP_NPS_COSMIC_COLUMN_VETO_EN <en>

<en> disables (when 0) the cosmic column veto logic, and enables it when 1. Enabling this veto will reject cosmic column triggers when there are hits in more than 1 column within a single crate.

VTP_NPS_ECALCLUSTER_CRATE_ID <id>

<id> must be set to a value of 1 to 5, matching the number in the VTP host name (the # in: nps-vtp#). This will ensure clusters are recorded with the correct Y offset in the event data.

VTP_NPS_ECALCLUSTER_HIT_DT <dt>

<dt> is the cluster hit coincidence time with respect to seed hit. Valid values for <dt> range from 0 to 7 and units are in +/-4ns (e.g. dt=0 means 0ns coincidence, dt=1 means +/-4ns coincidence, etc...)

VTP_NPS_ECALCLUSTER_SEED_THR <thr>

<thr> is the clustering minimum seed threshold – a channel must have a hit with this energy or greater to initiate finding a cluster. Valid values for <thr> range from 0 to 16383. The units are arbitrary and depend on the FADC250 channel gain conversions, which nominally target 1MeV units for this threshold.

VTP_NPS_ECALCLUSTER_NHIT_MIN <min>

<min> is the minimum number of hits a cluster must have in order to be accepted (for cluster singles, pair, and readout). Valid values for <min> range from 1 to 9.

VTP_NPS_ECALCLUSTER_CLUSTER_TRIGGER_THR <thr>

<thr> is the clustering minimum cluster energy threshold to create a singles cluster trigger. Valid values for <thr> range from 0 to 16383. The units are arbitrary and depend on the FADC250 channel gain conversions, which nominally target 1MeV units for this threshold.

VTP_NPS_ECALCLUSTER_CLUSTER_READOUT_THR <thr>
<thr> is the clustering minimum cluster energy threshold for sparsified readout to readout FADC channels around clusters. Valid values for <thr> range from 0 to 16383. The units are arbitrary and depend on the FADC250 channel gain conversions, which nominally target 1MeV units for this threshold.

VTP_NPS_ECALCLUSTER_CLUSTER_PAIR_TRIGGER_THR <thr>
<thr> is the clustering minimum cluster energy threshold to create a pair1 or pair2 trigger (this threshold applies to individual clusters in the pair). Valid values for <thr> range from 0 to 16383. The units are arbitrary and depend on the FADC250 channel gain conversions, which nominally target 1MeV units for this threshold.

VTP_NPS_ECALCLUSTER_CLUSTER_PAIR_TRIGGER_WIDTH <width>
<width> is the cluster pair time coincidence window that is used for the cluster pair trigger for cluster found in the same crate. Valid values range from 0 to 124 and units are in 1ns.

VTP_NPS_FADCMASK_MODE <mode>
When <mode>=0 the sparsification logic reads out 5x5 FADC channel patterns. When <mode>=1 the sparisifcation logic reads out 7x7 FADC channel patterns.

VTP_NPS_FADCMASK_PRESCALE <prescale>
This prescale value can be set to occasionally readout all channels when sparisification mode is enabled. It provides a way to have a small fraction of events having all waveforms for potentially useful/special analysis that sparisfication mode may complicate. When <prescale> is 0, sparsification is disabled and all events have all FADC channels readout. When <prescale> is between 1 and 65534, every <prescale>+1 event will have all FADC channels readout. When <prescale> = 65535 then sparsification is enabled for all events.

VTP_NPS_FADCMASK_WIDTH <width>
Sparsification cluster hit <width> that defines how long clusters persist in sparisification detection logic once found. Valid range is 0 to 8188 and units are in nanoseconds. This width is the time the sparsification logic looks before and up to the the VTP_NPS_FADCMASK_OFFSET point in the readout window.

VTP_NPS_FADCMASK_OFFSET <offset>
Sparsification cluster hit <offset> that defines the time the sparisification detection logic once looks in the VTP readout window for clusters to determine FADC readout masks. Valid range is 0 to 8188 and units are in nanoseconds. The parameter is relative to the VTP_W_OFFSET parameter: this is intentional to simplify the timing - looking at the VTP cluster event data the time of cluster in the window

VTP_NPS_TRIG_DELAY <trgbit> <delay>
<trgbit> selects the trigger bit number, 0 to 31 that the <delay> will apply to. <delay> is the amount of delay to add to the trigger bit in nanoseconds. Valid values are 0 to 1020.

VTP_NPS_TRIG_PRESCALE <trgbit> <prescale>
<trgbit> selects the trigger bit number, 0 to 31 that the <prescale> will apply to. <prescale> is the amount to prescale the trigger bit outputs. 0 will disable the trigger bit output. 1 will enable the trigger bit output with no prescale. Values 2 to 65535 will prescale the output by that amount (e.g. <prescale>=2 will pulse the trigger output every other trigger...<prescale>=3 will pulse the trigger output every two triggers, etc.)

VTP_NPS_TRIG_LATENCY <latency>
<latency> is the VTP trigger latency in nanosecond units (this is roughly the amount of time from when a signal goes into the FADC to when the VTP will generate a trigger puse for cosmic, clusters, VLD, etc). Valid range is 0 to 8188, but some settings dictate the minimum acceptable value: for NPS it is around 2300ns minimum required for this parameter).

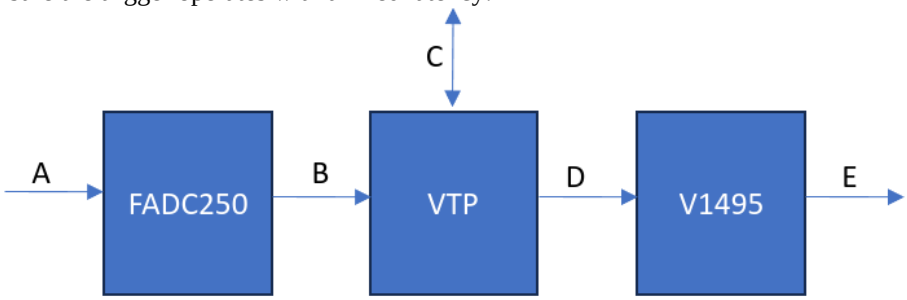
VTP_NPS_TRIG_WIDTH <width>

<width> is the VTP trigger output pulse width generated each time a trigger condition is satisfied. Units are in nanosecond. Valid range is 0 to 1020, actual pulse width will be this value +4ns.

9.9 Latencies

Some notes on general latencies expected through the trigger system:
A->B: ~400ns (FADC pulse to VTP Serdes RX output)
A->B->C: ~1700ns (FADC pulse to VTP from adjacent crate, to VTP over fiber Serdes RX)
A->B->C->D: ~2300ns (FADC pulse to VTP trigger output)
A->B->C->D: ~2330ns (FADC pulse to V1495 trigger output)

Most delays in the trigger processing is moving data around over serial links (VXS backplanes and fibers). A small amount is needed by the FADC to perform the "NSA" section of the pulse integration (time is roughly the amount of time defined by the NSA integration parameter). THe VTP clustering takes no more than a few hundred nanoseconds. The resulting latency ~2300ns requires the parameter **VTP_NPS_TRIG_LATENCY** be at least a little larger to ensure the trigger operates with a fixed latency.



9.10 Verification

The NPS trigger was verified through few ways, mainly: (1) FPGA logic simulator and (2) C/C++ logic emulator comparing VTP event builder data to reconstructed FADC data.

(1) FPGA logic simulation

This simulation compiles and simulates the FPGA code for the TI, FADC250, VTP, and V1495 modules that are all used in the real setup. The simulation runs the full system (5 crates, 1080 FADC channels) and consists of walking FADC hits across all possible cluster positions to ensure 3x3 clustering tests the FADC to calorimeter map and cluster across crate perimeters work. The simulation reads out the FADC event data and also the VTP event data to cross checking between them can be done. This includes testing of the sparisifcation logic mode to ensure FADC channels that don't paricipate in a cluster are not read while FADC channels that are near clusters over cluster readout thresholds are readout.

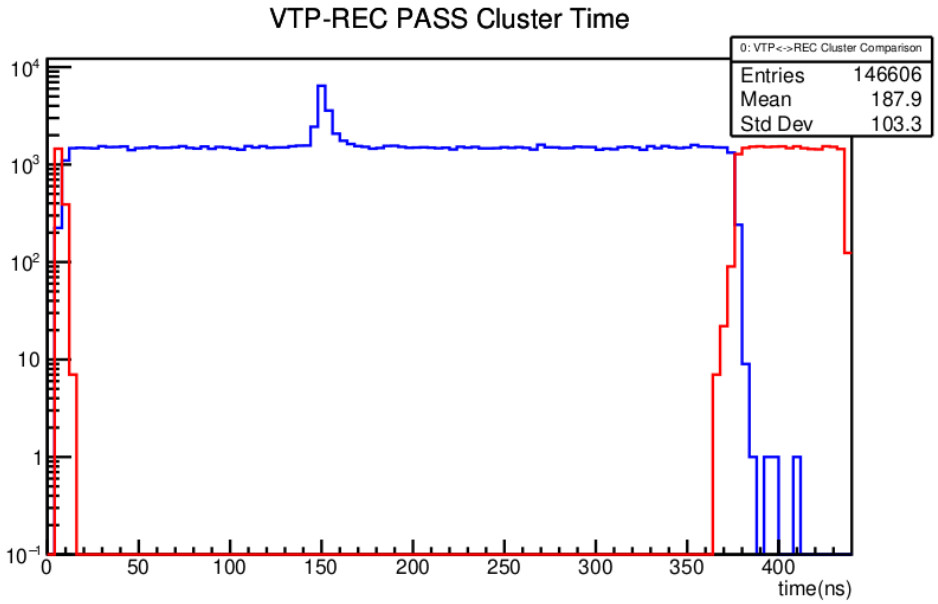
This simulator is extremely slow, taking about 1 hour of runtime to configure (a lot of VME emulation accesses to setup the FADCs and VTPs) and start to issue triggers. Once configuration is complete triggers and event readout can run at around 1 readout event every few minutes. Usually smaller scales of the setup were used to debug in much shorter time frames, then overnight simulation where run to check the full setup. This simulator is very accurate and very often reproduces problems found in real setups making it extremely valuable. It can also read EVIO files from CODA to process events to check the trigger logic on them.

The simulation software is Aldec Riviera and simulates using VHDL, Verilog, C, and C++ source codes.

(2) C/C++ Logic Emulator

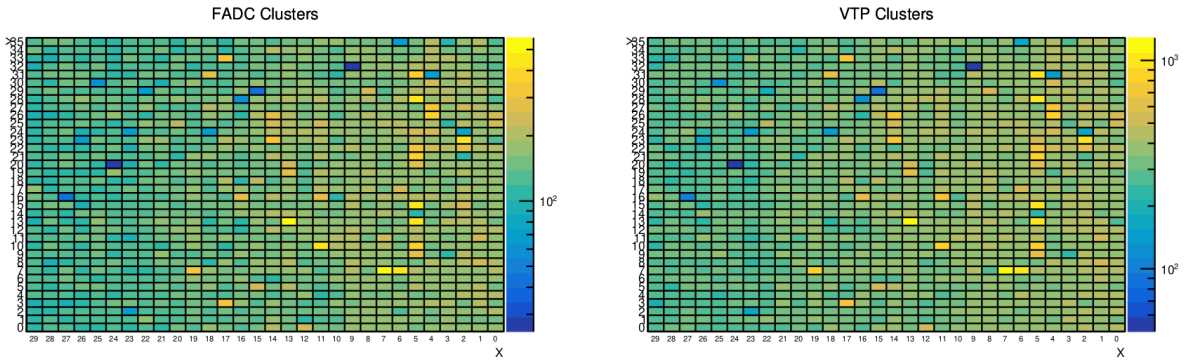
This is simple C/C++ application that emulates the trigger logic. It reads EVIO files and processes the FADC data to reconstruct the trigger logic. It also reads the VTP event data which has the trigger logic responses (trigger bits, times, and clusters). FADC reconstructed plots and be compared against the VTP plots directly as well as the cluster and trigger bits for all individual events. The accuracy is extremely good

as shown here where all FADC reconstructed trigger clusters are compared to the VTP reported clusters as a function of time in the readout window:

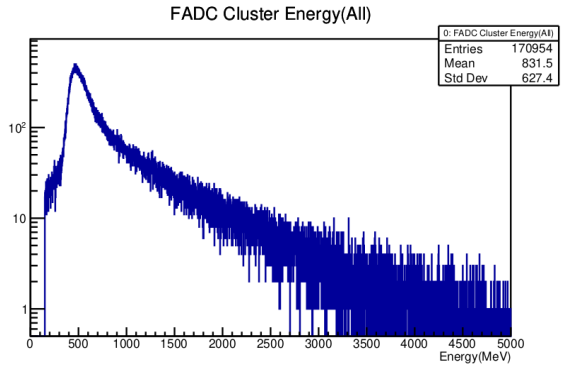


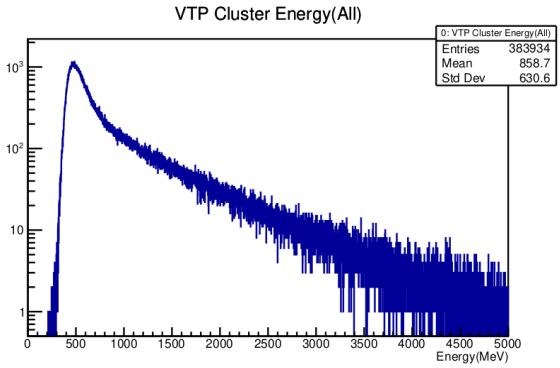
The blue line indicates where the is a match between the FADC reconstructed cluster and the VTP event data (X,Y, energy, time and number of hits in cluster). The timing peak around 150ns is the physics timed peak. The rest is background clusters. The red line indicates disagreements at the point in time between the FADC and VTP (i.e. a match is not found due to bad X, Y, time, energy, or number of hits). The reason for the disagreement at the window edges is because the FADC data has raw pulses that must be integrated to determine their energy, but the windows cause the integration to be clipped so some energy or hits are lost. The VTP doesn't suffer these windowing effects because it has no windowing effects when it is building clusters. The discrepancy at the window edges is expected and the agreement in center of the window is excellent demonstrating the C/C++ emulator is very accurate to the FPGA simulation. This means the emulator can be used to process data much faster (typically 1kHz or faster of event rate processing) to check for problems and debug a variety of issues (looking problems other than firmware issues that can cause physics trigger efficiency issues).

Many other plots are available as well. For example the number of clusters found in FADC data vs VTP data shows similar results, but the VTP has a larger readout window (and doesn't suffer window clipping) so it reports more clusters:

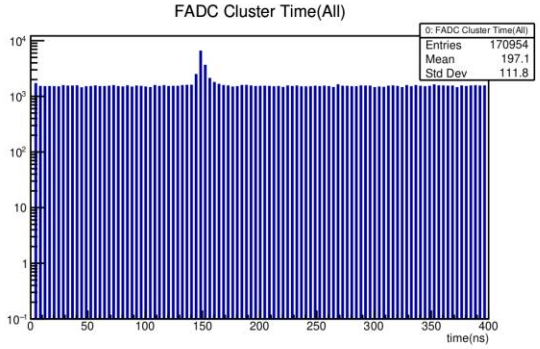


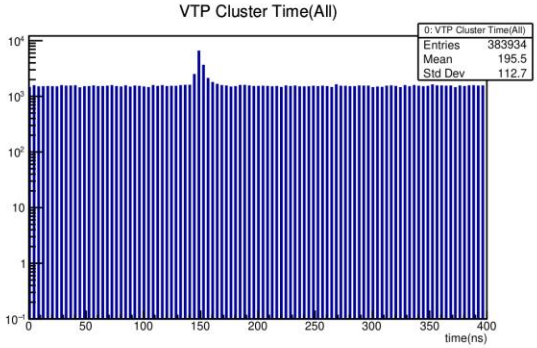
Cluster energy plots comparing FADC to VTP (again, VTP has no window clipping issues, but FADC does – so the FADC has an shoulder to the left that the VTP doesn't - otherwise, very similar):

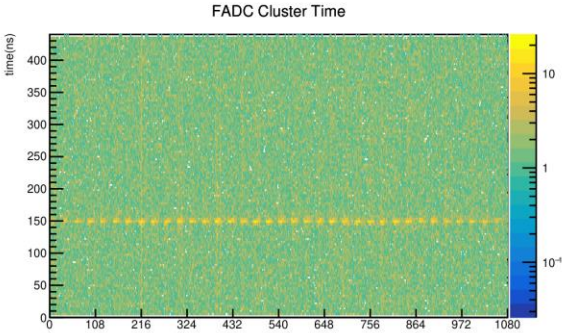


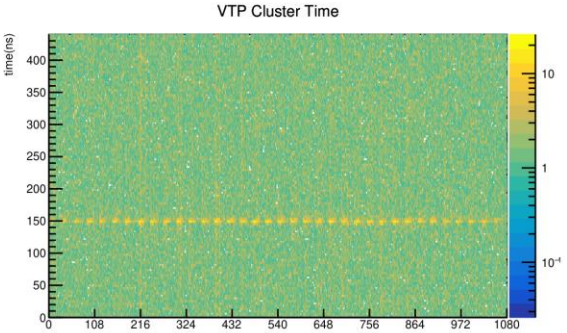


Cluster time comparision:

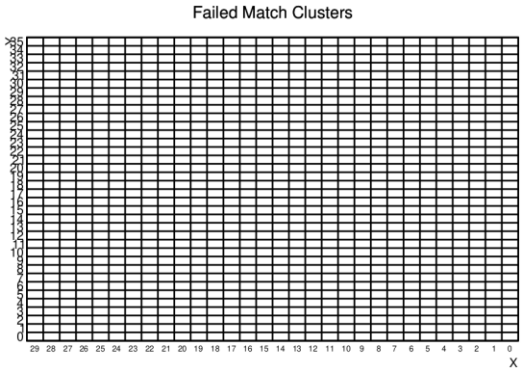


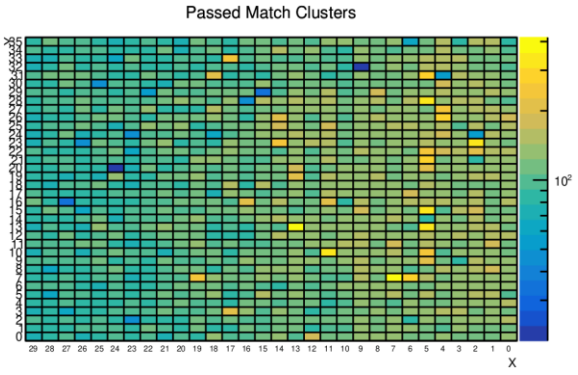






When adding a time-cut on the to not compare FADC vs VTP near the readout window, the following result is found for matching clusters across the different calorimeter positions:





10. Z7 FPGA Peripheral Register Descriptions

10.1 PER_ID_TI

Register: LINK_RESET

Address Offset: 0x0000
Size: 32bits
Reset State: 0x0000000F

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
-	-	-	-	FIFORST	RXFIFORST	-	RXRESET

FIFORST (RW):

'1' - TI & V7 event builder buffers reset
'0' - TI & V7 event builder buffers active

RXFIFORST (RW):

'1' - TI link receiving command FIFO reset
'0' - TI link receiving command FIFO active

RXRESET (RW):

'1' - TI link RX reset
'0' - TI link RX active

Notes:

For initial board configuration or when input clock changes:

1. Assert all reset bits
2. Wait until GCLK reference has been selected and stable (i.e. setup Si5341)
3. Release RX_RESET and confirm TI link is working (by checking bit RX_READY in LinkStatus)
4. Release RXFIFORST

Before triggers are allowed to be released:

1. Assert FIFORST to clear event builder buffers, then de-assert
2. Enable triggers

Register: TI_CTRL_REG

Address Offset: 0x0004
Size: 32bits
Reset State: 0x00000000

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
-	-	-	MODE	-	-	SEND_ACK	SEND_BL_REQ

MODE (RW):

'1' - FPGA ROC TI event readout mode
'0' - ARM CPU ROC (DMA/Single cycle) TI event readout mode

SEND_ACK (WO):

'1' - Send block acknowledgement to TI
'0' - does nothing

SEND_BL_REQ (WO):

'1' - Send block level request to TI
'0' - does nothing

Register: LINK_STATUS_REG

Address Offset: 0x0008
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	RX_LOCKED	RX_READY
15	14	13	12	11	10	9	8
RX_ERROR_CNT							
7	6	5	4	3	2	1	0
RX_ERROR_CNT							

RX_LOCKED (RW):

'1' - TI link is locked to bit stream
'0' - TI link is not locked to bit stream

RX_READY (RO):

'1' - TI link is up
'0' - TI link is down

RX_ERROR_CNT (RO):

Number of bit errors (8b10b invalid codes or disparity errors) seen on TI link. A non-zero value indicates a problem.

Register: TI_STATUS_REG

Address Offset: 0x000C
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
NEXT_BL_DATA							
7	6	5	4	3	2	1	0

CUR_BL_DATA

NEXT_BL_DATA (RO):
TI broadcasted next block level

CUR_BL_DATA (RO):
TI broadcasted current block level

Register: EB_STATUS_REG

Address Offset: 0x0014
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
EB_FIFO_SYNC	EB_FIFO_CNT_DATA						
23	22	21	20	19	18	17	16
EB_FIFO_CNT_DATA						-	EB_FIFO_END
15	14	13	12	11	10	9	8
EB_FIFO_CNT_LEN							
7	6	5	4	3	2	1	0
EB_FIFO_CNT_LEN		-	-	EB_FIFO_EMPTY_LEN	-	-	EB_FIFO_EMPTY

EB_FIFO_SYNC (RO):

'1' - Current TI block is a sync-event
'0' - Current TI block is not a sync-event

EB_FIFO_CNT_DATA (RO):

Number of words in block data FIFO

EB_FIFO_END (RO):

‘1’ - dummy word, indicating the end of a data block

EB_FIFO_CNT_LEN (RO):

Number of words in block length FIFO (this also represent number of blocks in FIFO)

EB_FIFO_EMPTY_LEN (RO):

'1’ - TI block length FIFO empty
'0’ - TI block length FIFO not empty

EB_FIFO_EMPTY (RO):

'1’ - TI block data FIFO empty
'0’ - TI block data FIFO not empty

Register: EB_TI_FIFO_REG

Address Offset: 0x0018
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
EB_FIFO_DOUT							
23	22	21	20	19	18	17	16
EB_FIFO_DOUT							
15	14	13	12	11	10	9	8
EB_FIFO_DOUT							
7	6	5	4	3	2	1	0
EB_FIFO_DOUT							

EB_FIFO_DOUT (RO):

Read this register for single cycle TI block readout. Status of FIFO should be known before reading this register to ensure data is present and ready for read.

Register: EB_TI_FIFO_REG

Address Offset: 0x0024
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
EB_FIFO_DOUT_LEN							
23	22	21	20	19	18	17	16
EB_FIFO_DOUT _LEN							
15	14	13	12	11	10	9	8
EB_FIFO_DOUT _LEN							
7	6	5	4	3	2	1	0
EB_FIFO_DOUT _LEN							

EB_FIFO_DOUT_LEN (RO):

Read this register for single cycle TI block readout to determine the block length for the next block of data (alternatively the data FIFO may be read until the status indicates the end of block). Status of FIFO should be known before reading this register to ensure data is present and ready for read.

9.2 PER_ID_CLK

Register: CTRL_REG

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000007

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
-	-	-	-	-	IDELAYCTRL_RESET	CLK_40GBE_RESET	GCLK_RESET

GCLK_RESET (RW):

'1' - Resets global reference clocks used to generate TI link references
'0' - Releases reset

GCLK_40GBE_RESET (RW):

'1' - Resets reference clock used by 40/10Gbps Ethernet MAC/Phy
'0' - Releases reset

IDELAYCTRL_RESET (RW):

'1' - Resets programmable delay reference module used by TI link and ebiorx buses
'0' - Releases reset

Register: STATUS_REG

Address Offset: 0x0004
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	IDELAYCTRL_RDY	PLL_LOCKED	GCLK_LOCKED

GCLK_LOCKED (RO):

'1' - Global reference clocks PLL is locked
'0' - Global reference clocks PLL is not locked

PLL_LOCKED (RO):

'1' - 40/10Gbps Ethernet MAC/Phy PLL locked
'0' - 40/10Gbps Ethernet MAC/Phy PLL not locked

IDELAYCTRL_RDY (RO):

'1' - Programmable delay reference module ready
'0' - Programmable delay reference module not ready

Register: VERSION_REG

Address Offset: 0x0008
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
VERSION_MAJOR							
23	22	21	20	19	18	17	16
VERSION_MAJOR							
15	14	13	12	11	10	9	8

VERSION_MINOR							
7	6	5	4	3	2	1	0
VERSION_MINOR							

VERSION_MAJOR (RO):

Firmware major version number

VERSION_MINOR (RO):

Firmware minor version number

Register: FIRMWARE_TYPE_REG

Address Offset: 0x0010
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
FIRMWARE_TYPE							
23	22	21	20	19	18	17	16
FIRMWARE_TYPE							
15	14	13	12	11	10	9	8
FIRMWARE_TYPE							
7	6	5	4	3	2	1	0
FIRMWARE_TYPE							

FIRMWARE_TYPE (RO):

1 = Zynq Streaming Firmware Type
2 = Zynq HW CODA ROC Firmware Type

Register: TIMESTAMP_REG

Address Offset: 0x0014
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
TIMESTAMP							
23	22	21	20	19	18	17	16
TIMESTAMP							
15	14	13	12	11	10	9	8
TIMESTAMP							
7	6	5	4	3	2	1	0
TIMESTAMP							

TIMESTAMP (RO):

Zynq FPGA bitstream compile timestamp

Bits 5:0 = seconds
Bits 11:6 = minute
Bits 16:12 = hour (24hr)
Bits 22:17 = year-2000
Bits 26:23 = month (Jan=0, Feb=1, …)
Bits 31:27 = day of month

9.3 PER_ID_CODA_ROC

Register: CTRL_REG

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	CPU_EVT_ASYNC_EN	CPU_EVT_SYNC_EN	V7_EVT_EN
7	6	5	4	3	2	1	0
-	-	-	-	DDR_DMA_RST_N	USE_DDR_BUFFERING	PAUSE_EVENTS	RESET

RESET (RW):

'1' - Resets HW CODA ROC
'0' - Releases reset

PAUSE_EVENTS (RW):

'1' - Will temporarily halt sending blocks of events to event builder (will always pause on block boundaries)
'0' - Allows event block transfers to event builder

USE_DDR_BUFFER (RW):

'1' - Enables additional Zynq DDR memory to buffer multiple/many EVIO blocks
'0' - No DDR buffering used, relying on smaller buffers and TCP buffers alone

DDR_DMA_RST_N (RW):

'1' - Release reset of DDR DMA module
'0' - Reset DDR DMA module (if USE_DDR_BUFFER=0, then this DDR_DMA_RST_N should remain 0)

V7_EVT_EN (RW):

'1' - Enables/requires V7 FPGA EVIO banks to be present & built
'0' - Disables V7 FPGA EVIO banks to event builder

CPU_EVT_SYNC_EN (RW):

'1' - Enables/requires ARM CPU “synchronous” banks to be present & built (each event)
'0' - Disables ARM CPU "synchronous” banks

CPU_EVT_ASYNC_EN (RW):

'1' - Enables ARM CPU “asynchronous” banks to be built (if async event is available the HW CODA ROC will send it at the next available event block boundary).
'0' - Disables ARM CPU "asynchronous” banks

Register: ROC_REG

Address Offset: 0x0004
Size: 32bits
Reset State: 0x00000000

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
ROCID							

ROCID (RW):

VTP HW CODA ROC ID

Register: ROC_STATUS

Address Offset: 0x0008
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	TCP_FULL
15	14	13	12	11	10	9	8
CODA_ROC_STATE				ROC_HEADER_STATE			
7	6	5	4	3	2	1	0
EVIO_HEADER_STATE				TI_DATA_STATE			

TCP_FULL (RO):

'1' - TCP engine input buffer is full (or TCP link not connected)
'0' - TCP engine can accept data from CODA ROC

CODA_ROC_STATE (RO):

0 – IDLE: waiting for TI event or ASYNC event
1 – GET_TI_EVT_LEN: read TI event length
2 – GET_V7_EVT_LEN: read V7 event length
3 – GET_CPU_EVT_LEN: read CPU synchronous event length
4 – WRITE_EVIO_HEADER: writing network transfer EVIO header
5 – WRITE_ROC_HEADER: writing ROC evio header
6 – WRITE_V7_EVT: writing V7 event data
7 – WRITE_CPU_EVT: writing CPU synchronous event data
8 – WRITE_CPU_ASYNC_EVT: writing CPU asynchronous event data
9 – WRITE_TI_ACK: acknowledges TI (block ACK)

ROC_HEADRER_STATE (RO):

0 – RECORD_LENGTH: writing ROC record length
1 – STATUS_ROCID: writing status/ROCID
2 – TRIGGER_BANK_LENGTH: writing trigger bank length
3 – TI_DATA: writing TI data

EVIO_HEADER_STATE (RO):

0 – CMSG_HEADER: writing CMSG type/length header
1 – BLOCK_LENGTH: writing block length & number
2– HEADER_LENGTH: writing header & length
3– RESERVED1: writing reserved1
4– RESERVED2: writing reserved2

TI_DATA_STATE (RO):

0 – BLOCK_LENGTH: writing block length & number
1 – HEADER_LENGTH: writing header & length
2 – RESERVED1: writing reserved1
3 – RESERVED2: writing reserved2

Register: CPU_EVT_SYNC_DIN

Address Offset: 0x0010
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
CPU_EVT_SYNC_DIN							
23	22	21	20	19	18	17	16
CPU_EVT_SYNC_DIN							
15	14	13	12	11	10	9	8
CPU_EVT_SYNC_DIN							
7	6	5	4	3	2	1	0
CPU_EVT_SYNC_DIN							

CPU_EVT_SYNC_DIN (WO):

Write this register to fill the CPU synchronous event data. The CPU_EVT_LEN_SYNC_DIN register must also be written to signal that HW CODA ROC that data is ready. This FIFO can only hold 512 words so care must be taken not to overflow the buffer. If acceptable to write a larger event size than this buffer – this can be done by writing the full event length to the CPU_EVT_LEN_SYNC_DIN register and then continue checking the FIFO status, CPU_EVT_SYNC_STATUS_REG, and when not full writing a single 32bit word.

Register: CPU_EVT_SYNC_STATUS_REG

Address Offset: 0x0014
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
CPU_EVT_SYNCFULL	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
-	-	-	-	-	-	CPU_EVT_SYNC_WRLEN	
7	6	5	4	3	2	1	0
CPU_EVT_SYNC_WRLEN							

CPU_EVT_SYNCFULL (RO):

'1' - CPU synchronous event FIFO is full
'0' - CPU synchronous event FIFO is not full

CPU_EVT_SYNC_WRLEN (RO):

Number of words in FIFO. Can be used to check how much free space is available in buffer if full size is known (512 words)

Register: CPU_EVT_LEN_SYNC_DIN

Address Offset: 0x0018
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
CPU_EVT_LEN_SYNC_DIN							
23	22	21	20	19	18	17	16
CPU_EVT_LEN_SYNC_DIN							
15	14	13	12	11	10	9	8
CPU_EVT_LEN_SYNC_DIN							
7	6	5	4	3	2	1	0
CPU_EVT_LEN_SYNC_DIN							

CPU_EVT_LEN_SYNC_DIN (WO):

Write this register to fill the CPU synchronous event data length (0 is an acceptable length). A single write per block of data is required when CPU synchronous event is enabled. There must be a matching number of written words to the CPU_EVT_SYNC_DIN register – writing the data or length first doesn't matter.

Register: CPU_EVT_LEN_SYNC_STATUS_REG

Address Offset: 0x001C
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
CPU_EVT_LEN_SYNCFULL	-	-	-	-	-	CPU_EVT_ OUTSTANDING	
23	22	21	20	19	18	17	16
CPU_EVT_OUTSTANDING							
15	14	13	12	11	10	9	8
-	-	-	-	-	-	CPU_EVT_LEN_SYNC_WRLEN	
7	6	5	4	3	2	1	0
CPU_EVT_LEN_SYNC_WRLEN							

CPU_EVT_LEN_SYNCFULL (RO):

'1' - CPU synchronous event length FIFO is full
'0' - CPU synchronous event length FIFO is not full

CPU_EVT_ LEN_ SYNC_WRLEN (RO):

Number of words in FIFO. Can be used to check how much free space is available in buffer if full size is known (512 words)

CPU_EVT_OUTSTANDING (RO):

Number of outstanding unwritten CPU SYNCHRONOUS events that must be written to allow hardware CODA ROC to fully build event blocks. This counter is incremented for each block of events received from the TI, and decremented when the CPU_EVT_LEN_SYNC FIFO is written.

Register: CPU_EVT_ASYNC_DIN

Address Offset: 0x0020
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
CPU_EVT_ASYNC_DIN							
23	22	21	20	19	18	17	16
CPU_EVT_ASYNC_DIN							
15	14	13	12	11	10	9	8
CPU_EVT_ASYNC_DIN							
7	6	5	4	3	2	1	0
CPU_EVT_ASYNC_DIN							

CPU_EVT_ASYNC_DIN (WO):

Write this register to fill the CPU asynchronous event data. The CPU_EVT_LEN_ASYNC_DIN register must also be written to signal that HW CODA ROC that data is ready. This FIFO can only hold 512 words so care must be taken not to overflow the buffer. If acceptable to write a larger event size than this buffer – this can be done by writing the full event length to the CPU_EVT_LEN_ASYNC_DIN register and then continue checking the FIFO status, CPU_EVT_ASYNC_STATUS_REG, and when not full writing a single 32bit word.

Register: CPU_EVT_ASYNC_STATUS_REG

Address Offset: 0x0024
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
CPU_EVT_ASYNCFULL	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
-	-	-	-	-	-	CPU_EVT_ASYNC_WRLEN	
7	6	5	4	3	2	1	0
CPU_EVT_ASYNC_WRLEN							

CPU_EVT_ASYNCFULL (RO):

'1' - CPU asynchronous event FIFO is full
'0' - CPU asynchronous event FIFO is not full

CPU_EVT_ASYNC_WRLEN (RO):

Number of words in FIFO. Can be used to check how much free space is available in buffer if full size is known (512 words)

Register: CPU_EVT_LEN_ASYNC_DIN

Address Offset: 0x0028
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
CPU_EVT_LEN_ASYNC_DIN							
23	22	21	20	19	18	17	16
CPU_EVT_LEN_ASYNC_DIN							
15	14	13	12	11	10	9	8
CPU_EVT_LEN_ASYNC_DIN							
7	6	5	4	3	2	1	0
CPU_EVT_LEN_ASYNC_DIN							

CPU_EVT_LEN_ASYNC_DIN (WO):

Write this register to fill the CPU asynchronous event data length (>0). There must be a matching number of written words to the CPU_EVT_ASYNC_DIN register. Writing the length register will signal the HW CODA ROC that an asynchronous event is to be presented at the next opportunity.

Register: CPU_EVT_LEN_ASYNC_STATUS_REG

Address Offset: 0x002C
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
CPU_EVT_LEN_ASYNCFULL	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
-	-	-	-	-	-	CPU_EVT_LEN_ASYNC_WRLEN	
7	6	5	4	3	2	1	0
CPU_EVT_LEN_ASYNC_WRLEN							

CPU_EVT_LEN_ASYNCFULL (RO):

'1' - CPU asynchronous event length FIFO is full
'0' - CPU asynchronous event length FIFO is not full

CPU_EVT_ LEN_ ASYNC_WRLEN (RO):

Number of words in FIFO. Can be used to check how much free space is available in buffer if full size is known (512 words)

Register: TI_STATUS_TRIG_NUMBER

Address Offset: 0x0040

Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
TRIG_NUMBER							
23	22	21	20	19	18	17	16
TRIG_NUMBER							
15	14	13	12	11	10	9	8
TRIG_NUMBER							
7	6	5	4	3	2	1	0
TRIG_NUMBER							

TRIG_NUMBER (RO):

32bit trigger number from the last processed TI event. This value is initialized to 0 when CTRL_REG->RESET is asserted.

Register: TI_ACK_COUNT

Address Offset: 0x004C
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
ACK_COUNT							
23	22	21	20	19	18	17	16
ACK_COUNT							
15	14	13	12	11	10	9	8
ACK_COUNT							
7	6	5	4	3	2	1	0
ACK_COUNT							

ACK_COUNT (RO):

32bit trigger number of all VTP->TI readout ACK sent. This value is initialized to 0 when CTRL_REG->RESET is asserted.

Register: BYTES_SENT0

Address Offset: 0x0044
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
BYTES_SENT(31:24)							
23	22	21	20	19	18	17	16
BYTES_SENT(23:16)							
15	14	13	12	11	10	9	8
BYTES_SENT(15:8)							
7	6	5	4	3	2	1	0
BYTES_SENT(7:0)							

BYTES_SENT (RO):

Number of bytes transmitted to CODA EB through the ethernet port.
This value is reset by the CODA_ROC_PER CTRL_REG->RESET
Reading this value latches the upper 32bit value in BYTE_SENT1 so that when BYTES_SENT0, then BYTES_SENT1 are read sequentially a coherent 64bit read is done.

Register: BYTES_SENT1

Address Offset: 0x0048
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
BYTES_SENT(63:56)							
23	22	21	20	19	18	17	16
BYTES_SENT(55:48)							
15	14	13	12	11	10	9	8
BYTES_SENT(47:40)							
7	6	5	4	3	2	1	0
BYTES_SENT(39:32)							

BYTES_SENT (RO):

Number of bytes transmitted to CODA EB through the ethernet port.
This value is reset by the CODA_ROC_PER CTRL_REG->RESET

Register: ROC_MAX_BLOCK_SIZE

Address Offset: 0x0070
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
ROC_MAX_BLOCK_SIZE							
23	22	21	20	19	18	17	16
ROC_MAX_BLOCK_SIZE							
15	14	13	12	11	10	9	8
ROC_MAX_BLOCK_SIZE							
7	6	5	4	3	2	1	0
ROC_MAX_BLOCK_SIZE							

ROC_MAX_BLOCK_SIZE (RW):

Maximum EVIO block size in 32bit units (typically 1M 32bit words ~4MBytes) used to predict/calculate when the DDR buffer is full and when to wrap around address pointers. This must be set before the DDR DMA component reset is released (bit 3 of CTRL_REG). This is also used as the EVIO network block size threshold/limit, which forces the data block to be built/sent before exceeding this threshold.

Register: ROC_MAX_BLOCKS

Address Offset: 0x0074
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	ROC_MAX_BLOCKS			
7	6	5	4	3	2	1	0
ROC_MAX_BLOCKS							

ROC_MAX_BLOCKS (RW):

This is used a the EVIO network block threshold. Once this number of event blocks in reached in the current EVIO network block being built it will force it to be sent.

Register: ROC_BLOCKTIMEOUT

Address Offset: 0x0078
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
ROC_BLOCKTIMEOUT							
23	22	21	20	19	18	17	16
ROC_BLOCKTIMEOUT							
15	14	13	12	11	10	9	8
ROC_BLOCKTIMEOUT							
7	6	5	4	3	2	1	0
ROC_BLOCKTIMEOUT							

ROC_BLOCKTIMEOUT (RW):

This is used a the EVIO network block timeout threshold. If no event blocks have been added to the active EVIO network block after the specified timeout it will force the block to be sent. The units of this timer are 12.8ns, providing a maximum timeout of ~1 minute.

9.4 PER_ID_10GBE_TCPIP_CLIENT0

Register: CTRL_REG

Address Offset: 0x0000
Size: 32bits
Reset State: 0x0000C025

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	FIBER_CTRL_LINKSTATUS	DYNAMIC_IPv4
15	14	13	12	11	10	9	8
RESET_STACK	RESET_MAC	FIBER_CTRL_LPMODE	FIBER_CTRL_RESETL	FIBER_CTRL_MODSELL	TX_FAULT	SIGNAL_DETECT	PMA_PMD_TYPE
7	6	5	4	3	2	1	0
PMA_PMD_TYPE		PHY_POWER_DOWN	PHY_TEST_MODE		PHY_RESET	PHY_CONFIG_CHANGE	-

PHY_CONFIG_CHANGE (RW):

Optional: pulse 1 to activate settings on PHY_TEST_MODE

PHY_RESET (RW):

'1' - Reset ethernet phy layer
'0' - Release reset

PHY_TEST_MODE (RW):

0 – normal mode
1 – loopback mode
2 – remote loopback mode

PHY_POWER_DOWN (RW):

'1' - Power=off ethernet phy
'0' - Power=on ethernet phy

PMA_PMD_TYPE (RW):

Specify the type of SFP+ transceiver installed:
7 – 10GBASE-SR
6 – 10GBASE-LR
5 – 10GBASE-ER

SIGNAL_DETECT (RW):

'1' - SFP+ signal detected (controlled by software – later to be moved to firmware)
'0' - SFP+ signal not detected

TX_FAULT (RW):

'1' - SFP+ tx fault detected (controlled by software – later to be moved to firmware)
'0' - SFP+ tx fault not detected

FIBER_CTRL_MODSEL (RW):

'1' - SFP+ I2C bus not selected
'0' - SFP+ I2C bus selected

FIBER_CTRL_RESETL (RW):

'1' - SFP+ not reset
'0' - SFP+ reset

FIBER_CTRL_LPMODE (RW):

'1' - SFP+ low power mode enabled
'0' - SFP+ low power mode disabled

RESET_MAC (RW):

'1' - Reset ethernet MAC layer

'0' - Release reset

RESET_STACK (RW):

'1' - Reset TCP/IP stack

'0' - Release reset

DYNAMIC_IPv4 (RW):

'1' - use DHCP

'0' - fixed IP

FIBER_CTRL_LINKSTATUS (RW):

'1' - SFP+ link light enabled

'0' - SFP+ link light disabled

Register: STATUS_REG

Address Offset: 0x0004

Size: 32bits

Reset State: 0x00000000

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
				TCP_TX_CTS0	FIBER_CTRL_MOD_PRSL	-	FIBER_CTRL_INTL

FIBER_CTRL_INTL (RO):

'0' - SFP+ interrupt active

'1' - SFP+ interrupt not active

FIBER_CTRL_MOD_PRSL (RO):

'0' - SFP+ module is present

'1' - SFP+ module is not present

TCP_TX_CTS0 (RO):

'1' - TCP socket 0 clear to send (debugging signal)

'0' - TCP socket 0 not clear to send (debugging signal)

Register: IP4_ADDR_REG

Address Offset: 0x0010

Size: 32bits

Reset State: 0x00000000

31	30	29	28	27	26	25	24
IP4_ADDR							
23	22	21	20	19	18	17	16
IP4_ADDR							
15	14	13	12	11	10	9	8
IP4_ADDR							
7	6	5	4	3	2	1	0
IP4_ADDR							

IP4_ADDR (RW):

IPv4 address for static IP mode

Register: IP4_MCASTADDR_REG

Address Offset: 0x0014

Size: 32bits

Reset State: 0x00000000

31	30	29	28	27	26	25	24
IP4_MCASTADDR							
23	22	21	20	19	18	17	16
IP4_MCASTADDR							
15	14	13	12	11	10	9	8
IP4_MCASTADDR							
7	6	5	4	3	2	1	0
IP4_MCASTADDR							

IP4_MCASTADDR (RW):

IPv4 multicast address

Register: IP4_SUBNETMASK_REG

Address Offset: 0x0018
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
IP4_SUBNETMASK							
23	22	21	20	19	18	17	16
IP4_SUBNETMASK							
15	14	13	12	11	10	9	8
IP4_SUBNETMASK							
7	6	5	4	3	2	1	0
IP4_SUBNETMASK							

IP4_SUBNETMASK (RW):

IPv4 subnet mask

Register: IP4_GATEWAYADDR_REG

Address Offset: 0x001C
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
IP4_GATEWAYADDR							
23	22	21	20	19	18	17	16
IP4_GATEWAYADDR							
15	14	13	12	11	10	9	8
IP4_GATEWAYADDR							
7	6	5	4	3	2	1	0
IP4_GATEWAYADDR							

IP4_GATEWAYADDR (RW):

IPv4 gateway address

Register: TCP_STATE_REQUESTED_REG

Address Offset: 0x0020
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	-	TCP_STATE_REQUESTED

TCP_STATE_REQUESTED (RW):

'0' - go back to idle (terminate connection is currently connected or connecting)
'1' - initiate connection

Register: TCP_KEEPALIVE_REG

Address Offset: 0x0028
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	-	TCP_KEEPALIVE_EN

TCP_KEEPALIVE_EN (RW):

'0' - disable TCP keep alive

'1' - enable TCP keep alive

Register: TCP_STATE_STATUS_REG

Address Offset: 0x002C
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	TCP_STATE_STATUS			

TCP_STATE_STATUS (RW):

0 – connection closed

1 – connecting

2 – connected

3 – unreachable IP

4 – destination port busy

Register: TCP_STATUS_REG

Address Offset: 0x0030
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	-	TCP_CONNECTED_FLAG

TCP_CONNECTED_FLAG (RW):

'0' – not connected

'1' – connected

Register: MAC_ADDR0_REG

Address Offset: 0x0034
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
MAC_ADDR[31..24]							
23	22	21	20	19	18	17	16
MAC_ADDR[23..16]							
15	14	13	12	11	10	9	8

MAC_ADDR[15..8]							
7	6	5	4	3	2	1	0
MAC_ADDR[7..0]							

MAC_ADDR (RW):

Lower 32bits of 10Gbps MAC address

Register: MAC_ADDR1_REG

Address Offset: 0x0038
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
MAC_ADDR[47..40]							
7	6	5	4	3	2	1	0
MAC_ADDR[39..32]							

MAC_ADDR (RW):

Upper 16bits of 10Gbps MAC address

Register: MAC_STATUS0_REG

Address Offset: 0x003C
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
MAC_N_RX_FRAMES							
23	22	21	20	19	18	17	16
MAC_N_RX_FRAMES							
15	14	13	12	11	10	9	8
MAC_N_TX_FRAMES							
7	6	5	4	3	2	1	0
MAC_N_TX_FRAMES							

MAC_N_TX_FRAMES (RO):

Number of transmitted MAC frames

MAC_N_RX_FRAMES (RO):

Number of received MAC frames

Register: MAC_STATUS1_REG

Address Offset: 0x0040
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
MAC_N_RX_FRAMES_TOO_SHORT							
23	22	21	20	19	18	17	16
MAC_N_RX_FRAMES_TOO_SHORT							
15	14	13	12	11	10	9	8
MAC_N_RX_BAD_CRCS							
7	6	5	4	3	2	1	0
MAC_N_RX_BAD_CRCS							

MAC_N_RX_BAD_CRCS (RO):

Number of received MAC frames with invalid CRC

MAC_N_RX_FRAMES_TOO_SHORT (RO):

Number of received MAC frames under valid length

Register: MAC_STATUS2_REG

Address Offset: 0x0044

Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
MAC_N_RX_FRAMES_WRONG_ADDR							
23	22	21	20	19	18	17	16
MAC_N_RX_FRAMES_WRONG_ADDR							
15	14	13	12	11	10	9	8
MAC_N_RX_TOO_LONG							
7	6	5	4	3	2	1	0
MAC_N_RX_TOO_LONG							

MAC_N_RX_FRAMES_WRONG_ADDR (RO):
Number of received MAC frames with MAC address not matching local host

MAC_N_RX_FRAMES_TOO_LONG (RO):
Number of received MAC frames over valid length

Register: MAC_STATUS3_REG
Address Offset: 0x0048
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
MAC_RX_IPG							
23	22	21	20	19	18	17	16
MAC_RX_IPG							
15	14	13	12	11	10	9	8
MAC_N_RX_LENGTH_ERRORS							
7	6	5	4	3	2	1	0
MAC_N_RX_LENGTH_ERRORS							

MAC_N_RX_LENGTH_ERRORS (RO):
Number of received MAC frames with length field != to received frame length

MAC_RX_IPG (RO):
Measured interpacket gap (in bytes)

Register: PCS_STATUS_REG
Address Offset: 0x0050
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	TX_DISABLE
7	6	5	4	3	2	1	0
PCS_CORE_STATUS							

PCS_CORE_STATUS (RO):
Bit0 – PCS block is locked
Bit7:1 - reserved

TX_DISABLE (RO):
Tx disable request from PCS core (move to firmware later)

Register: PHY_STATUS_REG
Address Offset: 0x0060

Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
PHY_STATUS2							
23	22	21	20	19	18	17	16
PHY_STATUS							
15	14	13	12	11	10	9	8
PHY_ID							
7	6	5	4	3	2	1	0
PHY_ID							

PHY_ID (RO):
Phy chip ID

PHY_STATUS (RO):
XAUI status (not used)

PHY_STATUS2 (RO):
SFP+ status from phy

Register: UDP_DEST_IP_ADDR_REG
Address Offset: 0x0064
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
UDP_DEST_IP_ADDR[31:24]							
23	22	21	20	19	18	17	16
UDP_DEST_IP_ADDR[23:16]							
15	14	13	12	11	10	9	8
UDP_DEST_IP_ADDR[15:8]							
7	6	5	4	3	2	1	0
UDP_DEST_IP_ADDR[7:0]							

UDP_DEST_IP_ADDR (RW):
Destination/server IP address for UDP connection

Register: UDP_PORT_REG
Address Offset: 0x0064
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
UDP_LOCAL_PORT[15..8]							
23	22	21	20	19	18	17	16
UDP _LOCAL_PORT[7..0]							
15	14	13	12	11	10	9	8
UDP _DEST_PORT[15..8]							
7	6	5	4	3	2	1	0
UDP _DEST_PORT[7..0]							

UDP_DEST_PORT (RW):
UDP destination port (on remote/target server)

UDP_LOCAL_PORT (RW):
UDP local port to use

Register: TCP_DEST_IP_ADDR_REG
Address Offset: 0x0080
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
TCP_DEST_IP_ADDR[31:24]							

23	22	21	20	19	18	17	16
TCP_DEST_IP_ADDR[23:16]							
15	14	13	12	11	10	9	8
TCP_DEST_IP_ADDR[15:8]							
7	6	5	4	3	2	1	0
TCP_DEST_IP_ADDR[7:0]							

TCP_DEST_IP_ADDR (RW):

Destination/server IP address for TCP connection

Register: TCP_PORT_REG

Address Offset: 0x00A0
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
TCP_LOCAL_PORT[15..8]							
23	22	21	20	19	18	17	16
TCP_LOCAL_PORT[7..0]							
15	14	13	12	11	10	9	8
TCP_DEST_PORT[15..8]							
7	6	5	4	3	2	1	0
TCP_DEST_PORT[7..0]							

TCP_DEST_PORT (RW):

TCP destination port (on remote/target server)

TCP_LOCAL_PORT (RW):

TCP local port to use

9.5 PER_ID_EBIORX0

Register: CTRL_REG

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000000

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
FIFO_RST	RX_DATA_ENABLE	ENABLE_MONITOR	ENABLE_PHASE_DETECTOR	DCD_CORRECT	BITSLIP	SYNC_ERROR_RESET	RESET

RESET (RW):

'1' - reset receiver/state machine
'0' - run receiver/state machine

SYNC_ERROR_RESET (RW):

Reset sync error counter (used in link alignment/verification)

BITSLIP (WO):

Write '1' to pulse/bitstip the deserializers (used in link alignment/verification)

DCD_CORRECT (RW):

Should be left '0'

ENABLE_PHASE_DETECTOR (RW):

Should be left '1'

RX_DATA_ENABLE (RW):

'0' - ignores received data words
'1' - accepted received data words

FIFO_RST (RW):

'0' - RX FIFO reset released
'1' - RX FIFO reset asserted

Register: STATUS_REG

Address Offset: 0x0004
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
SYNC_ERROR_CNT							
23	22	21	20	19	18	17	16
SYNC_ERROR_CNT							
15	14	13	12	11	10	9	8
-	-	-	BIT_TIME_VALUE				
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	RX_LOCKED

RX_LOCKED (RO):

'1' - receiver locked on incoming datasteam
'0' - receiver not locked on incoming datasteam

BIT_TIME_VALUE (RO):

Calculated bit time

SYNC_ERROR_CNT (RO):

Number of received errors

10. V7 FPGA Peripheral Register Descriptions

10.1 PER_ID_MPDFIBER0..39

Register: GTX_CTRL_REG

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000007

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
-	-	-	-	-	POWERDOWN	RESET	GT_RESET

GT_RESET (RW):

'1' - Assert gigabit link hardware reset
'0' - Deassert gigabit link hardware reset

RESET (RW):

'1' - Assert serial link protocol state machines
'0' - Deassert serial link protocol state machines

POWERDOWN (RW):

'1' - Power down the gigabit hardware
'0' - Power the gigabit hardware

Register: GTX_STATUS_REG

Address Offset: 0x0010
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
FIBER_ERR_CNT							
7	6	5	4	3	2	1	0
-	-	-	-	-	FIBER_FRAME_ERR	FIBER_HARD_ERR	FIBER_CHANNEL_UP

FIBER_CHANNEL_UP (RO):

'1' - Fiber/MPD link is up
'0' - Fiber/MPD link is down

FIBER_HARD_ERR (RO):

'1' - Fiber/MPD link has an error requiring the link to be reset
'0' - Normal operation

FIBER_FRAME_ERR (RO):

'1' - Fiber link protocol framing error occurred
'0' - Normal operation

FIBER_ERR_CNT (RO):

Number of soft/disparity errors on link since last reset

Register: EB_CTRL_REG

Address Offset: 0x0020
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	ENABLE_CM	BUILD_DEBUG_HEADERS	BUILD_ALL_SAMPLES	EVT_ENABLE

EVT_ENABLE (RW):

'1' - MPD participates is enabled in the event building
'0' - MPD is ignored by event builder

BUILD_ALL_SAMPLES (RW):

'1' - Disables zero suppression so all APV samples are readout
'0' - Zero suppression is enabled and APV samples may be rejected

BUILD_DEBUG_HEADERS (RW):

'1' - Additional readout data format types are enabled, useful for debugging
'0' - Debug headers are not readout

ENABLE_CM (RW):

'1’ - Enables the common-mode suppression logic
'0’ - Disables the common-mode suppression logic

Register: MAX_RX_LEN_REG

Address Offset: 0x0028
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	-	BUSY

BUSY (RO):

'1' - Input buffer for this MPD is full, pausing the event builder
'0' - Normal operation

Register: APV_OFFSET_REG

Address Offset: 0x0034
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
WR	-	-	-	-	ADDR		
23	22	21	20	19	18	17	16
ADDR							
15	14	13	12	11	10	9	8
-	-	-	OFFSET				
7	6	5	4	3	2	1	0
OFFSET							

OFFSET (RW):

13bit signed offset value

ADDR (RW):

11bit address. [6:0] = APV strip. [10:7] = APV ID

APV ID = 15 (which is not supported by MPD) is used to store the common-mode min and max thresholds. In this case:
APV strip = 0: APV ID = 0 common-mode min
APV strip = 1: APV ID = 0 common-mode max
APV strip = 2: APV ID = 1 common-mode min
APV strip = 3: APV ID = 1 common-mode max
...
APV strip = 28: APV ID = 14 common-mode min
APV strip = 29: APV ID = 14 common-mode max

WR (WO):

'1' - write OFFSET into ADDR (this bit is automatically cleared, so another write can be issued immediately)
'0' - does nothing

Register: APV_THR_REG

Address Offset: 0x0038
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
WR	-	-	-	-	ADDR		
23	22	21	20	19	18	17	16
ADDR							
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	THR
7	6	5	4	3	2	1	0

THR

THR(RW):

9bit unsigned threshold

ADDR (RW):

11bit address. [6:0] = APV strip. [10:7] = APV ID

WR (WO):

'1' - write THR into ADDR (this bit is automatically cleared, so another write can be issued immediately)

'0' - does nothing

10.2 PER_ID_MPDREGS

Register: DATA_REG

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

DATA (RW):

Writing this register will perform a write to the MPD fiber and address as defined by ADDR_REG

Reading this register will perform a read to the MPD fiber and address as defined by ADDR_REG

Register: ADDR_REG

Address Offset: 0x003C
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	MPD_SEL				
23	22	21	20	19	18	17	16
-	-	-	-	-	ADDR		
15	14	13	12	11	10	9	8
ADDR							
7	6	5	4	3	2	1	0
ADDR							

MPD_SEL (RW):

0 to 31 selects the active fiber to use for the register access

ADDR (RW):

19bit byte address to access on the selected MPD

********** BELOW: OLD CONTENTS TO IGNORE **********

11. Readout Data Format

The word length for the readout data is 32bits. The event length is variable and depends on several factors (detector occupancy, headers, trailers, filler words).

Data Word Categories

Data words from the module are divided into two categories: Data Type Defining (bit 31 = 1) and Data Type Continuation (bit 31 = 0). Data Type Defining words contain a 4-bit data type tag (bits 30 - 27) along with a type dependent data payload (bits 26 - 0). Data Type Continuation words provide additional data payload (bits 30 – 0) for the *last defined data type*. Continuation words permit data payloads to span multiple words and allow for efficient packing of various data types spanning multiple data words. Any number of Data Type Continuation words may follow a Data Type Defining word.

Data Type List

0 Block Header
1 Block Trailer
2 Event Header
3 Trigger Time
4 Reserved
5 Reserved
6 Trigger
7 Reserved
8 Reserved
9 Reserved
10 Reserved
11 Reserved
12 Reserved
13 Reserved
14 Data Not Valid (empty module)
15 Filler Word (non-data)

Data Type: Block Header

Type: 0x0
Size: 1 word
Description: Indicates the beginning of a block of events. (High-speed readout of a board or a set of boards is done in blocks of events)

31	30	29	28	27	26	25	24
1	0	0	0	0	-	-	-
23	22	21	20	19	18	17	16
-	-	NUM_EVENTS					
15	14	13	12	11	10	9	8
NUM_EVENTS					BLOCK_NUMBER		
7	6	5	4	3	2	1	0
BLOCK_NUMBER							

BLOCK_NUMBER:
Event block number (used to align blocks when building events)

NUM_EVENTS:
Number of events in block

Data Type: Block Trailer

Type: 0x1
Size: 1 word
Description: Indicates the end of a block of events. The data words in a block are bracketed by the block header and trailer.

31	30	29	28	27	26	25	24
1	0	0	0	1	-	-	-
23	22	21	20	19	18	17	16
-	-	NUM_WORDS					
15	14	13	12	11	10	9	8
NUM_WORDS							
7	6	5	4	3	2	1	0
NUM_WORDS							

NUM_WORDS:
Total number of words in block of events

Data Type: Event Header

Type: 0x2
Size: 1 word
Description: Indicates the start of an event. The included trigger number is useful to ensure proper alignment of event fragments when building events. The 27bit trigger number (134M count) is not a limitation, as it will be used to distinguish events within event blocks, or among events that are concurrently being built or transported.

31	30	29	28	27	26	25	24
1	0	0	1	0	TRIGGER_NUMBER		
23	22	21	20	19	18	17	16
TRIGGER_NUMBER							
15	14	13	12	11	10	9	8
TRIGGER_NUMBER							
7	6	5	4	3	2	1	0
TRIGGER_NUMBER							

TRIGGER_NUMBER:
Accepted event/trigger number

Data Type: Trigger Time

Type: 0x3
Size: 2 words
Description: Time of trigger occurrence relative to the most recent global reset. The time is measured by a 48bit counter that is clocked from the 125MHz system clock. The assertion of the global reset clears the counter. The de-assertion of global reset enables counter and thus sets t=0 for the module. The trigger time is necessary to ensure system synchronization and is useful in aligning event fragments when building events.

Word 1:

31	30	29	28	27	26	25	24
1	0	0	1	1	0	0	0
23	22	21	20	19	18	17	16
TRIGGER_TIME_H							
15	14	13	12	11	10	9	8
TRIGGER_TIME_H							
7	6	5	4	3	2	1	0
TRIGGER_TIME_H							

TRIGGER_TIME_H:

This is the upper 24bits of the trigger time

Word 2:

31	30	29	28	27	26	25	24
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
TRIGGER_TIME_L							
15	14	13	12	11	10	9	8
TRIGGER_TIME_ L							
7	6	5	4	3	2	1	0
TRIGGER_TIME_ L							

TRIGGER_TIME_L:

This is the lower 24bits of the trigger time

Data Type: Data Not Valid

Type: 0x14
Size: 1 word
Description: Module has no data available for readout. This can if the module is being read out too quickly after receiving (event building is in process and no data words have been put into the buffer yet) a trigger or if the module doesn't have any events to report.

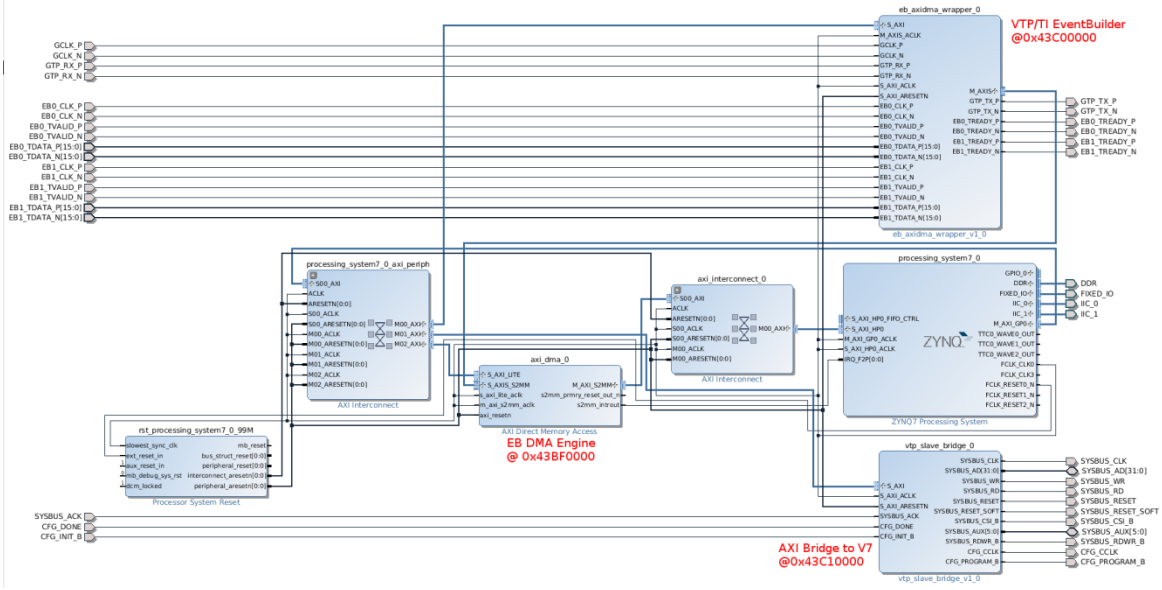
31	30	29	28	27	26	25	24
1	1	1	1	0	UNDEFINED		
23	22	21	20	19	18	17	16
UNDEFINED							
15	14	13	12	11	10	9	8
UNDEFINED							
7	6	5	4	3	2	1	0
UNDEFINED							

Data Type: Filler Word

Type: 0x15
Size: 1 word
Description: Non-data word appended to the block of events. This is used to force the total number of 32-bit words read out of a module to be a multiple of 2 or 4 when

31	30	29	28	27	26	25	24
1	1	1	1	1	UNDEFINED		
23	22	21	20	19	18	17	16
UNDEFINED							
15	14	13	12	11	10	9	8
UNDEFINED							
7	6	5	4	3	2	1	0
UNDEFINED							

Additionally, the Zynq7 contains an FPGA which currently has a few FPGA based peripherals. The Zynq7 processor and FPGA system is shown in the following diagram. The Zynq7 connects to the FPGA based peripherals (axi_dma0, eb_axidma_wrapper_0, vtp_slave_bridge_0) using the Master AXI bus interface (this is a 32bit data bus used for accessing control/status registers in both Zynq and V7 FPGAs). The "eb_axidma_wrapper_0" peripheral buffers and combines event block streams from the V7 and TI where the "axi_dma_0" peripheral takes the stream and can DMA it to the Zynq7 processor over the S_AXI_HP0 bus (which connects to the DDR3 memory).



FPGA Peripherals:

Register Name	Description	Address Offset
Event Builder peripheral	**VTP/TI Event Builder**	**0x43C00000**
LinkCtrl	TI<->VTP Serial Link Control	0x0000
TiCtrl	TI Control	0x0004
LinkStatus	TI<-> VTP Serial Link Status	0x0008
TiStatus	TI Status	0x000C
EbCtrl	Event Builder Control	0x0010

EB AXI DMA peripheral		**0x43BF0000**

V7 Bridge & FPGA Config		**0x43C10000**
Bridge space	Memory maps a ~64kByte window of V7 into Zynq7	0x0000
V7Status	V7 FPGA Configuration Status	0xFFF4
V7Ctrl	V7 FPGA Control	0xFFF8
V7Cfg	V7 FPGA Configuration	0xFFFC

V7 Clk peripheral		**0x43C10100**
Ctrl	Clock control	0x0000
Status	Clock status	0x0004

Sd peripheral (offset 0x0200)		**0x43C10200**
Ctrl	Sd Control	0x0000
Status	Sd Status	0x0004
ScalerLatch	Scaler latch control	0x0008
FPAOSel	Front Panel LVDS[31:0] Source Selection Control	0x0010
FPBOSel	Front Panel Mezzanine Source Selection Control	0x0014
BusySel	Busy Source Selection Control	0x0018
Trig1Sel	Trig1 Source Selection Control	0x001C
SyncSel	Sync Source Selection Control	0x0020
FPAOVal	Front Panel LVDS[31:0] Output Value	0x0040

FPBOVal	Front Panel Mezzanine Output Value	0x0044
FPBIStatus	Front Panel Mezzanine Input Status	0x0060
Trig1Status	Trig1 Status	0x0064
Trig2Status	Trig2 Status	0x0068
SyncStatus	Sync Status	0x006C

FadcDecoder peripheral		**0x43C10300**
Ctrl	Fadc Enable Control	0x0000
Latency[0-15]	Fadc Latency Status	0x0020 … 0x005C

VXS Serdes peripheral		**0x43C11000 … 0x43C11F00**
Ctrl	GTH Control	0x0000
Status	GTH Statusl	0x0004
DrlCtrl	Drp Control	0x0008
DrpStatus	Drp Status	0x000C

QSFP Serdes Peripheral		**0x43C12000 … 0x43C12300**
Ctrl	GTH Control	0x0000
Status	GTH Statusl	0x0004
DrlCtrl	Drp Control	0x0008
DrpStatus	Drp Status	0x000C

ECTrigger Peripheral		**0x43C14100 … 0x43C14200**
Ctrl	EC Trigger Control	0x0000

Trigger Ouptut Peripheral		**0x43C15000**
Latency	Trigger Latency control	0x0000
Width	Trigger Width Control	0x0004
BusyScaler		0x0010

V7 Event Builder Peripheral		**0x43C15100**
BlockSize	Event Builder Block Size	0x0000
TriggerFifoBusyThreshold	Queued Trigger Busy Threshold	0x0004
Lookback	Readout Window Lookback	0x0008
WindowWidth	Readout Window Width	0x000C

6.1 Event Builder Peripheral (Base Address 0x43C00000)

6.2 AXI DMA Peripheral (Base Address 0x43BF0000)

Refer to AXI DMA v7.1 manual (pg021_axi_dma.pdf) for details on this peripheral. It is a IP block from Vivado and is responsible for performing DMA operations to efficiently move data from Event Builder buffers into the Zynq7 processor memory.

For Linux driver/device-tree setup refer here:
http://www.wiki.xilinx.com/DMA+Drivers+-+Soft+IPs#AXI%20DMA--Device%20Tree%20Node

6.3 V7 Bridge & FPGA Config (Base Address 0x43C10000)

This peripheral bridges the V7 register space into the Zynq7 processor memory space. It uses a custom bus/protocol and is also used as the configuration interface for the V7 FPGA image. The V7 FPGA is setup as a 16bit SelectMap slave device for configuration. The last three 32bit words in the 64kB peripheral space are used for V7 FPGA configuration control and status registers, while the remaining lower space in the 64kB peripheral space maps registers in the V7 FPGA. This allows the Zynq7 processor to read/write registers using normal memory reads/writes.

Register: V7Status

Address Offset: 0xFFF4
Size: 32bits
Reset State: 0xXXXXXXXX

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
-	-	-	-	-	-	INIT_B	DONE

INIT_B (RO):

Reads the V7 INIT_B status
'0' - indicates V7 FPGA configuration is in the reset state

DONE (RO):

Reads the V7 DONE status
'1' - indicates the V7 FPGA configuration is valid and running
'0' - indicates the V7 FPGA configuration is invalid or incomplete

Register: V7Ctrl

Address Offset: 0xFFF8
Size: 32bits
Reset State: 0x00000000

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
-	-	-	PROGRAM_B	RDWR_B	CSI_B	RESET_SOFT	RESET

PROGRAM_B (WO):

'0' - FPGA configuration reset
'1' - FPGA configuration can commence

RDWR_B (WO):

'1' - indicates the V7 FPGA configuration bus will perform a read on next V7Cfg access
'0' - indicates the V7 FPGA configuration bus will perform a write on next V7Cfg access

CSI_B (WO):

'1' - deselects V7 FPGA configuration
'0' - selects V7 FPGA configuration

RESET_SOFT (WO):

'1' - asserts soft reset signal to Z7 and V7 peripherals
'0' - deasserts soft reset signal to Z7 and V7 peripherals

RESET (WO):

'1' - asserts hard reset signal to Z7 and V7 peripherals
'0' - deasserts hard reset signal to Z7 and V7 peripherals

Register: V7Cfg

Address Offset: 0xFFFC
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
CFG_DATA							
7	6	5	4	3	2	1	0
CFG_DATA							

CFG_DATA (RW):

Data to read/write to the V7 configuration interface

6.4 V7 Clk (Base Address 0x43C10100)

Register: Ctrl

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000001

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
-	-	-	-	-	-	-	GCLK_RESET

GCLK_RESET (RW):

'1' - GCLK PLL reset
'0' - GCLK PLL run

Register: Status

Address Offset: 0x0004
Size: 32bits
Reset State: 0xXXXXXXXX

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
-	-	-	-	-	-	-	GCLK_LOCKED

GCLK_LOCKED (RO):

'1' - GCLK PLL locked
'0' - GCLK PLL not locked

6.5 Sd (Base Address 0x43C10200)

Register: Ctrl

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	FPB_OEN	FPB_SEL

FPB_OEN (RW):

Front panel mezzanine expansion board Output Enable (Inverted) control. Not all mezzanine board types can disable their output. Refer to mezzanine board document on how this signal will control that board type (A395D: 0=enable, 1=disable).

FPB_SEL (RW):

Front panel mezzanine expansion board signal format selection control. Not all mezzanine board types can disable their output. Refer to mezzanine board document on how this signal will control that board type (A395D: 0=NIM, 1=TTL).

Register: Status

Address Offset: 0x0004
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	FPB_ID		

FPB_ID (RO):

Front panel mezzanine expansion board identifier:

0: A395A (32 x IN LVDS/ECL)
1: A395B (32 x OUT LVDS)
2: A395C (32 x OUT ECL)
3: A395D (8 x IN/OUT NIM/TTL)

Register: ScalerLatch

Address Offset: 0x0008
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	-	LATCH

LATCH (RW):

Distributed to board scalers & histograms for global control capture of counters. Two types of counters (buffered & unbuffered) use this signal differently. For buffered counters, the counter is copied to the readout register on the 0>1 transition of LATCH allowing dead-timeless operation. For unbuffered counters, the counts is halt while LATCH is 1, this allows all counters to stop for readout, then when LATCH returns to 0 the counters can continue to count.

Register: FPAOSel

Address Offset: 0x0010
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
SEL							
23	22	21	20	19	18	17	16
SEL							
15	14	13	12	11	10	9	8
SEL							
7	6	5	4	3	2	1	0
SEL							

SEL (RW):

Each bit in SEL corresponds to the same bit on the front panel LVDS output bus.
'1' - bit uses user programmable logic to drive LVDS bit
'0' - bit uses FPAOVal register bit to drive LVDS bit

Register: FPBOSel

Address Offset: 0x0014
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
SEL							
23	22	21	20	19	18	17	16
SEL							
15	14	13	12	11	10	9	8
SEL							
7	6	5	4	3	2	1	0
SEL							

SEL (RW):

Each bit in SEL corresponds to the same bit on the front panel mezzanine output bus.
'1' - bit uses user programmable logic to drive mezzanine output bit
'0' - bit uses FPBOVal register bit to drive mezzanine output bit

Register: BusySel

Address Offset: 0x0018
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	SEL	

SEL (RW): determines what is output on VXS BUSY signal to TI
- 0 - '0'
- 1 - '1'
- 2 - VTP Event Builder BUSY logic
- 3 - reserved

Register: Trig1Sel

Address Offset: 0x001C
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	SEL	

SEL (RW): determines what signal is routed to VTP internal Trig1
- 0 - '0'
- 1 - '1'
- 2 - VXS Trig1
- 3 - reserved

Register: SyncSel

Address Offset: 0x0020
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	SEL	

SEL (RW): determines what signal is routed to VTP internal Sync
- 0 - '0'
- 1 - '1'
- 2 - VXS Sync
- 3 - reserved

Register: FPAOVal

Address Offset: 0x0040

Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

VAL (RW):

Each bit in VAL corresponds to the same bit on the front panel LVDS output bus when the corresponding bit in FPAOSel is set to drive a value from this register.

Register: FPBOVal

Address Offset: 0x0044
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

VAL (RW):

Each bit in VAL corresponds to the same bit on the front panel mezzanine output bus when the corresponding bit in FPBOSel is set to drive a value from this register.

Register: FPBIStatus

Address Offset: 0x0060
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

VAL (RO):

Each bit in VAL corresponds to the bit value read on the front panel mezzanine input bus.

Register: Trig1Status

Address Offset: 0x0064
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	-	STATUS

STATUS (RO):

Value read on VXS Trig1 line

Register: Trig2Status

Address Offset: 0x0068
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	-	STATUS

STATUS (RO):

Value read on VXS Trig2 line

Register: SyncStatus

Address Offset: 0x006C
Size: 32bits
Reset State: 0x00000000

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
-	-	-	-	-	-	-	STATUS

STATUS (RO):

Value read on VXS Sync line