The CLAS12 Trigger System

B. Raydo[a], S. Boyarinov[a], A. Celentano[b], C. Cuevas[a], R. De Vita[b], V. Kubarovsky[a], B. Moffit[a], R. Paremuzyan[a], C. Smith[a]

[a] *Thomas Jefferson National Accelerator Facility, Newport News, VA, USA*
[b] *INFN, Milan, Italy*

Abstract

The article describes the CLAS12 Trigger System. Simulation, hardware and software design, as well as all validation procedures are discussed. Firmware development tools used are discussed as well, including our experience with VIVADO High Level Synthesis.

1. Overview

The CLAS12 Trigger System provides trigger signals for the CLAS12 detector Data Acquisition System (DAQ, [1]). Originally it was designed to select physics events with scattered electrons detected in the CLAS12 Electromagnetic Calorimeter System (ECAL, [8]) and High Threshold Cherenkov Counter (HTCC, [7]), with the possibility to require a track in the CLAS12 Drift Chamber (DC, [9]). In later stages of development, signals from additional detectors were included into the Trigger System, making it flexible and efficient to select events for various experiments within the CLAS12 physics program.

2. Requirements

The CLAS12 detector was designed to study the interactions of electrons and photons with nucleons and nuclei at nominal luminosity of 1×10^{35} cm^{-2}s^{-1}. The CLAS12 trigger system has to provide trigger signals for these processes. Based on the simulation of the physics processes of interest, the required event rate was estimated to be up to 20 kHz. The trigger latency was required to be not less then 8 μs to provide sufficient time for trigger logic processing.

The following detectors were defined to be part of the trigger system:

- High Threshold Cherenkov Counter (HTCC, [7])
- Drift Chambers (DC, [9])
- Forward Time-of-Flight System (FTOF, [5])
- Electromagnetic Calorimeter (ECAL, [8]).
- Central Time-of-Flight (CTOF, [6])
- Central Neutron Detector (CND, [4])
- Forward Tagger (FT, [10])

Each of these detectors is required to provide information to the trigger system. To achieve that, the front-end electronics were required to be designed with built-in trigger components.

3. Design

The CLAS12 Trigger System was designed as a 3-stage pipeline-style system with total latency up to 8 *mu*s. Input information for the Trigger System comes from two sources: Flash Analog to Digital Converters (FADCs) used in the photomultiplier tube (PMT)-based detectors, and Drift Chamber Readout Boards (DCRBs) used in Drift Chambers. The FADCs and DCRBs work as the pre-trigger level, reporting information to the Trigger System in the appropriate form. Stage 1 receives information from the FADCs and DCRBs and performs data processing according to the type of detector. Stage 2 performs a timing and geometry coincidence between different subsets of detectors in six groups, corresponding to the six-sector CLAS12 detector structure, as well as requires coincidence with information from central detectors. Stage 3 forms the final trigger decision. The CLAS12 Trigger diagram is shown in Fig. 1.

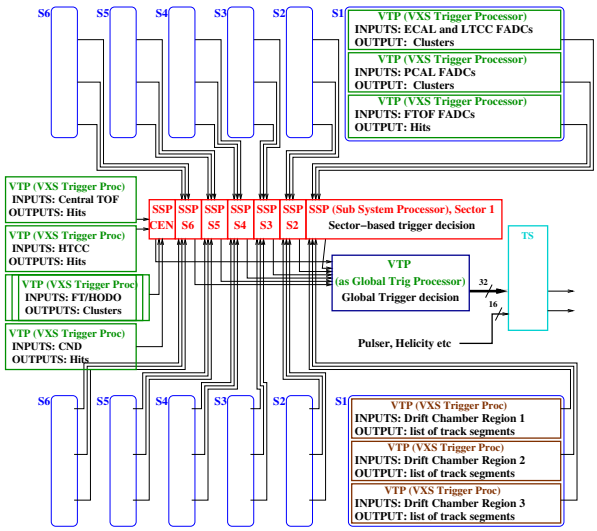


Figure 1: The CLAS12 Trigger System diagram.

3.1. FADCs as Pre-trigger

All PMT-based detectors in CLAS12 participating in the Trigger System use JLab VXS 250 MHz flash ADCs as the starting point of the trigger logic (FADC, [1]). Each channel of the FADC boards is pre-programmed with gain, pedestal, and amplitude threshold above pedestal. Every pulse above amplitude threshold is integrated and sent to the corresponding section of the Stage 1 trigger logic. The 16-channel FADC boards report 13-bit pulse integrals and 3-bit pulse time every 32 ns, which allows the following trigger logic to restore 4 ns pulse resolution while the double pulse resolution remains 32 ns. Based on the FADC reporting schedule, the following trigger logic stages can work on a 250 MHz clock, although in that case we found it problematic to meet the Field Programmable Gate Array (FPGA) timing. Because of that, our Stage 1 algorithms run on 125 MHz or slower clocks as described below. Tteh trigger information is provided to the following stages using VXS backplane serial lines.

3.2. DCRBs as Pre-trigger

The Drift Chamber-based trigger uses JLab 125 MHz discriminator/TDC boards (DCRB, [1]) to feed the Trigger System. These 96-channel units report hits above the pre-programmed thresholds every 16 ns. As for the FADC boards, the DCRBs are implemented in VXS format and provide trigger information using VXS backplane serial lines.

3.3. Stage 1 Trigger

The Stage 1 trigger uses specially designed VXS Trigger Processor boards (VTP, see Section 4.2). The VTP boards are installed in switch slots in every VXS crate participating in the Trigger System. The VTPs collect trigger data from the pre-trigger boards (FADCs and DCRBs) over VXS serial lines.

The most complex processing is performed for the electromagnetic calorimeters (cluster finding) and the Drift Chambers (segment and road finding). In the following sections we describe the design of the various trigger components.

3.3.1. Electromagnetic Calorimeters

The CLAS12 electromagnetic calorimeter (ECAL, [8]) includes two separate subsystems, the EC and PCAL. Each consists of multiple layers of scintillating strips and lead sheets with photomultiplier readout on one side of the scintillators (the PCAL is shown in Fig. 2, the EC is similar). The primary purpose of these detectors is electron identification by defining the energy and coordinate of their electromagnetic showers, referred to as clusters. The cluster finding algorithm was well established during off-line data processing development, and was adopted for the trigger implementation with some simplifications.

The algorithm first searches for one-dimensional clusters in each of the three calorimeter views (u,v,w), sorting them by energy and keeping only those above threshold, with a maximum number of four clusters in each view. Next the algorithm searches for two-dimensional clusters looking for overlap between the three views. For all two-dimension clusters found, it performs attenuation corrections based on pre-loaded tables of the attenuantion lengths of the scintillation strips using the distance from the cluster to the PMT, to deternime the correct cluster energy. Finally, the algorithm sorts the two-dimension clusters by energy and reports those above threshold, with a maximum number limited to four. For every cluster, the energy and coordinates are reported to the Stage 2 trigger every 8 ns. There is a persistency parameter that allows the same clusters to be reported for several consecutive 8 ns intervals to check for a timing coincidence with the other trigger components, as well as a timing delay parameter for the same purpose. One event with a single cluster is shown in the PCAL (preshower calorimeter) in Fig. 2. The corrected energies are shown for the individual strips.

It should be mentioned that such an algorithm is designed to find clusters with a maximum energy to tar-

get electron identification. For some CLAS12 experiments, it is necessary to identify minimum-ionizing particles (MIPs) using the same trigger component. For that purpose, clusters with energy below a certain defined threshold can be selected. Such a method works for events where the number of clusters does not exceed four, otherwise there is a risk of losing low-energy clusters corresponding to MIPs. Intensive trigger efficiency studies were conducted for such cases, and the MIP trigger efficiency was measured and found acceptable.

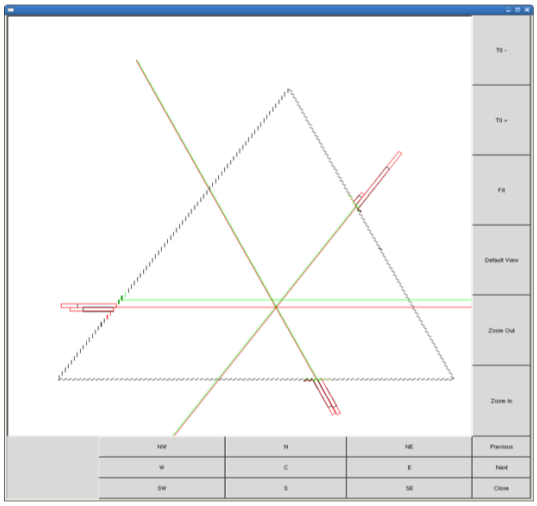


Figure 2: Trigger System representation of a cluster reconstruction using the three views of the PCAL in one sector of CLAS12.

3.3.2. *High Threshold Cherenkov Counter*

The CLAS12 High Threshold Cherenkov Counter (HTCC, [7]) serves as one of the primary components of the electron trigger logic. It was specially designed to discriminate electrons from other charged particles. The HTCC consists of 48 mirror sections readout by PMTs connected to FADCs (see Fig. 3). For trigger purposes, a 2x2 section sliding window is used to identify clusters. The cluster may include from one to four PMT signals collecting the Cherenkov light from the adjacent mirrors as shown in Fig. 3. The configuration parameters include the single channel energy threshold, cluster multiplicity threshold, and cluster energy threshold. The results are reported to the Stage 2 trigger as 48-bit masks every 4 ns. The FADC "gain" configuration parameter allows for PMT energy calibrations, making it possible to set energy thresholds in terms of the number of photoelectrons.

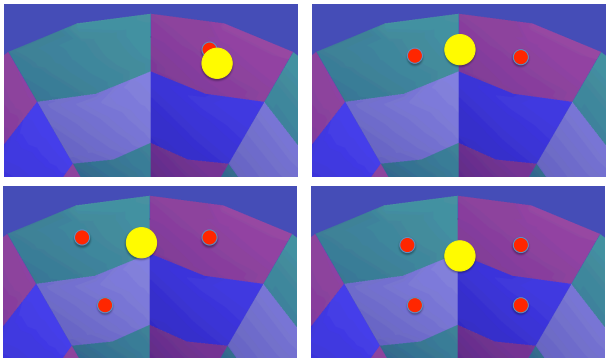


Figure 3: Hits registered by the HTCC (red circles) and the reconstructed cluster position (yellow). The hit position and cluster position coincide for one hit clusters (top left plot), which has the lowest position resolution.

3.3.3. *Drift Chamber*

The CLAS12 Drift Chambers (DC, [9]) contain six superlayers in each of the six CLAS12 sectors. Each superlayer contains six layers, and 112 wires in each layer. There is no signal amplitude information available, only hit information can be used in the trigger. The Trigger algorithm was designed as a two-step process.

In the first step it searches for segments in each of the six superlayers, reporting a 112-bit mask with the bits set for the segments found. The search for segments is conducted based on a pre-loaded segment dictionary, generated by the Drift Chamber simulation software based on the wire locations in the superlayers. If several segments are found in the same location, the one with the maximum number of hits is kept. In theory, the number of layers contributing to each segment must be equal to 6, and the number of hit wires in a segment can vary from 6 to 12 depending on the track position and angle. In practice, the number of layers and hits in each segment can be less because of Drift Chamber inefficiencies and hardware problems, so the threshold for the segment finder in the trigger logic was usually set to 4 and sometimes to 3 layers out of 6 to ensure an efficient trigger.

After the segment search is complete and the six 112-bit masks are ready, the second step is performed, in which a pre-loaded road dictionary (corresponding to all possible charged particle trajectories) is used to identify possible track candidates (so-called road finding). The road disctionaries were generated by the GEMC Monte Carlo simulation program (**??** or taken from the real

beam data (**??**. At least five out of six superlayers are required to satisfy the trigger condition. All found roads are reported to the Stage 2 trigger every 16 ns. The information reported in the form of 112-bit words is used for the geometry match in the Stage 2 trigger.

3.3.4. Forward Time-Of-Flight System

The CLAS12 Forward Time-Of-Flight System (FTOF, [5]) contains two layers of scintillating counters in each sector, but only one layer is used by the trigger logic. This layer contains 62 counters with PMT readout on both ends. When both PMTs report a signal above threshold, the trigger system considers it as a hit. A 62-bit hit mask is reported to the Stage 2 trigger every 4 ns. The trigger logic configuration includes a single channel energy threshold and a counter average energy threshold (geometry mean). The FTOF participates in non-electron triggers such as the muon trigger.

3.3.5. Central Time-Of-Flight System

The CLAS12 Central Time-Of-Flight System (CTOF, [6]) consists of 48 scintillation counters, surrounding the target as a barrel, with PMT readout from both ends. Its trigger logic is similar to that for FTOF, with a 48-bit mask reported to the Stage 2 trigger every 4 ns.

3.3.6. Central Neutron Detector

The CLAS12 Central Neutron Detector (CND, [4]) consists of three layers of scintillation counters, installed radially outward from CTOF, with 24 counters per layer and 72 counters total. Its trigger logic is similar to that for FTOF and CTOF, with a 24-bit mask reported to the Stage 2 trigger every 4 ns (usually the inner layer only).

3.3.7. Forward Tagger Calorimeter and Hodoscope

The CLAS12 Forward Tagger Calorimeter and Hodoscope (FT, [10]) trigger is designed to trigger on electrons at small forward angles (theta from 2deg to 5deg). The calorimeter is a stack of 332 lead tungstate crystals connected to avalanche photodiodes (APDs) that are readout by FADCs. The hodoscope consists of two scintillating fiber layers, each having 116 pixels (of two sizes) that matches the geometry of the calorimeter. The calorimeter trigger finds clusters by looking for a seed hit at each crystal location. If the deposited energy in a crystal is greater than the seed threshold and is a local maximum in space (using a 3x3 crystal view) and time, then it is considered a seed hit. For each seed hit, a cluster is formed by summing all of the energies centered on the seed hit in a 3x3 crystal view for all hit times coincident with the seed hit (up to ±16 ns). The seed hit time, which due to time walk effects is the earliest hit in the cluster, is used for the cluster time stamp, providing a 4 ns resolution. The geometrically matched hodoscope pixels for both layers are checked for time coincident hits with the calorimeter seed hit and the cluster is tagged as having none, layer 1, layer 2, or both layers of the hodoscope present. Found clusters are serialized and streamed to the Stage 2 trigger where several programmable trigger cuts can discriminate clusters based on energy, charge, and multiplicity.

Name	Specification
Latency (Stage 1+2)	5 μs
Jitter	4 ns
Stage 2 trigger bits	8
Deskew range	4 μs
Deskew step size	4 ns
Coincidence window range	2 μs
Deskew step size	4 ns

Table 1: Stage 2 Trigger Specifications

3.4. Stage 2 Trigger

The Stage 2 trigger collects data from Stage 1 using fiber optics. It is based on the number of SubSystem Processor boards (SSP, see Section 4.3) all installed in one VXS crate. After receiving the Stage 1 trigger streams, the SSPs form subsystem coincidences for the six identical sets of forward detectors (called sectors) and the central detectors (all separately). Each subsystem trigger stream goes through a programmable delay that provides 4 ns resolution when deskewing to optimize the time coincidence. Next follows a programmable coincidence window for each subsystem trigger stream, also with a 4 ns step resolution, to ensure that the different subdetector signals will remain stable long enough to form a time coincidence regardless of jitter due to particle time-of-flight, detector response, and trigger jitter. Stage 2 Trigger specifications are shown in Table 1.

The forward detectors in the trigger consist of FTOF, EC, PCAL, HTCC, and DC. A single SSP collects all forward detector trigger streams from a single sector of CLAS12. After the delay and concidence widths are applied to each input stream, the input streams are copied to 8 programmable sector trigger bits. Each sector trigger bit contains a variety of trigger primitives and customizeable thresolds/cuts that can be tailored for a particular trigger type. The sector trigger bits are computed

Primitive Name	Trigger Bit Parameters
PCU	Mask
FTOF	Mask
PCAL	Cluster Emin, Emax
ECAL	Cluster Emin, Emax
PCAL+ECAL	Cluster Emin
HTCC	Mask
Geometry Matched	
PCUxFTOF	Bar match tolerance
PCALxDC	Cluster Emin

Table 2: Forward Detector Trigger Primitives

Primitive Name	Trigger Bit Parameters
CND	Mask
CTOF	Mask
FT	Cluster Emin, Emax, Cluster Size, Hodoscope
Geometry Matched	
CNDxCTOF	Bar match tolerance

Table 3: Central Detector Trigger Primitives

and sent to the final Stage 3 trigger. Forward Detector Trigger primitives are shown in Table 2.

The central detectors participating in the trigger consist of CTOF, CND, and FT. A single SSP collects all central detector trigger streams. After the delay and concidence widths are applied to each input stream, the input streams are copied to 8 programmable central trigger bits. Each central trigger bit contains a variety of trigger primitives and customizable thresolds/cuts that can be tailored for a particular trigger type. The central trigger bits are computed and sent to the final Stage 3 trigger where all sector and central trigger bits arrive to compute the global trigger bits. Central Detector Trigger primitives are shown in Table 3.

3.5. Stage 3 Trigger

The Stage 3 trigger is the final stage and collects all sector and central trigger bit streams in a single module where they can be combined in a variety of ways to generate the global trigger bits used for reading out the Data Acquisition System (DAQ). It is implemented on a single VTP board installed in the switch slot on the same VXS crate where all Stage 2 trigger SSPs reside. There are 32 independent trigger bits that can form a trigger based on any combination of sector and/or central trigger bits. Each trigger bit contains two sector trigger bit conditions (required to both be true) and a signle central trigger bit condition. Additionally, each trigger bit contains a 16 bit prescaler, final pulse width, and scaler. Stage 3 Trigger specifications are shown in Table 4.

Name	Specification
Latency (Stage 1+2+3)	7 μs
Jitter	4 ns
Stage 3 trigger bits	32
Prescaler	0-65535
Trigger bit width	4 ns - 1 μs
Pulse rate	0.05 Hz - 125 MHz

Table 4: Stage 3 Trigger Specifications

3.6. Trigger Information in Data Stream

An important part of the Trigger System is the Event Builder, which allows the trigger components to participate in event-by-event readout the same way as is done for the DAQ components. All three stages of the Trigger System are equipped with Event Builders. Every time the CLAS12 DAQ is triggered, Stage 1 will build the data bank(s) with trigger decision details (such as the ECAL cluster coordinate/energy or DC segments/roads information), Stage 2 will build the data bank with sector-level and central detector coincidence results, and Stage 3 will build the data bank that contains the trigger bit decisions for all final 32 trigger bit decisions. Event Builders read information from the pipeline-style buffers for a given programmable window related to the readout trigger time. All trigger-related data banks are available in the data stream along with the DAQ data banks, providing detailed information about the trigger decision for every accepted event. In particular, this allows the Trigger System to be run in "tagging mode", which is a powerful way to test the trigger efficiency (using either a loose or a random trigger).

4. Hardware Implementation

The CLAS12 Trigger System is implemented using High Speed Serial (VXS) techniques for a complete fully pipelined multi-crate Trigger System that takes advantage of the elegant high-speed VXS serial extensions for VME. This trigger system includes a pre-trigger level and three stages, starting with the front-end VXS crate Trigger Processor (VTP), a sector-level SubSystem Processor (SSP), a global VTP processor (GTP),

and a Trigger Supervisor (TS) that manages the timing, synchronization, and front-end event readout.

Within a front-end crate, the trigger information is gathered from the pre-trigger boards, consisting of 16-channel, 12-bit FADC (**??**, and 96-channel DCRB (**??** modules via the VXS backplane, to a VXS Trigger Processor (VTP). Each VTP is capable of handling these 500 MBps VXS links from the 16 modules, and then performs real-time crate-level trigger algorithms. The VTP transmits the Stage 1 trigger information through multiple Gigabit transceivers that are combined into a fiber link. The VTP uses a multi-fiber link to increase the aggregate trigger data transfer rate to the global trigger to 10 Gbps.

The trigger data is transmitted on the VXS backplane, and on the multi-fiber link using the Aurora protocol from Xilinx. The front-end VXS modules use Virtex-V devices with Gigabit Transceivers operating at 2.5 Gbps. The VTP collects these serial streams with a Virtex7 device and works with a Zynq7 processor to manage the network interface and on-board Linux operating system.

The entire trigger system is synchronous and operates at 250 MHz with the Trigger Supervisor managing not only the front-end event readout, but also the distribution of the critical timing clocks, synchronization signals, and the global trigger signals to each front-end readout crate. These signals are distributed to the front-end crates on a separate fiber link, and each crate is synchronized using a unique encoding scheme to guarantee that each front-end crate is synchronous with a fixed latency, independent of the distance between each crate. The overall trigger signal latency is <8 μS, and the CLAS12 experiments require a trigger rate of up to 20 kHz, which can be easily handled since the hardware has an ability to operate with a trigger rate of up to 200 KHz.

The following sections describe the main Trigger System hardware components.

4.1. Pre-trigger Boards

Two type of boards are used at the pre-trigger level to supply information to the trigger system: FADC and DCRB. They are described in detail in the CLAS12 DAQ paper **??**

4.2. VTP Board

The VXS Trigger Processor (VTP), see Fig. 4) is a VXS switch card that is used to implement trigger logic on the front-end crates (Stage 1) and global trigger crate (Stage 3). There are 80 full-duplex serial links each capable of running at up to 8.5 Gbps that can be used for transporting the trigger data. The links are bonded in groups of 4 for a total of 20 channels, which include 16 VXS payload slot interfaces (copper) and 4 QSFP interfaces (optical).

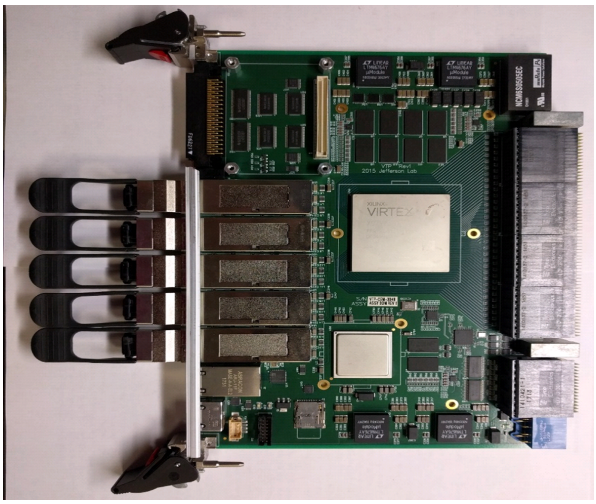


Figure 4: VXS Trigger Processor (VTP) Board.

Front-end (Stage 1) Crate Processing. The VTP in the front-end crate collects data from the VXS payload FADC and DCRB modules (and optionally from some of the QSFP links), where it aligns the data in time for all links, and presents it to the detector-specific trigger logic, which resides in a XC7V550T FPGA. The trigger logic processes the data and produces an output trigger stream that is sent to the Stage 2 trigger crate (and optionally to other Stage 1 VTP modules) using up to 4 QSFP optical links. The QSFP optical links allow the Stage 1 trigger logic to use information from multiple Stage 1 crates, which is required for some detectors that span multiple VXS crates (e.g. the DC and FT subsystems). The QSFP optical links also allow multiple links to go to Stage 2 when more bandwidth is needed (e.g. HTCC and CTOF).

Global Trigger (Stage 3) Crate Processing. In the global trigger crate the VTP collects data from the VXS payload SSP modules. The SSP modules supply a stream of trigger bits for each sector (HTCC, FTOF, ECAL, PCAL, and DC) and also a stream of trigger bits for the central detectors (CTOF, CND, and FT). These sector and central trigger bit streams have already performed timing, multiplicity, and geometry coincidences between the detectors within the sector or central detectors. The Stage 3 VTP allows the final ("global") trigger bits (up to 32) to be defined using different combinations for sectors, sector trigger bits, and central trigger

bits. The 32 global trigger bit decisions are evaluated at 250 MHz so that no additional jitter is introduced by this stage. These bits are sent to the TS using the high-density LVDS front-panel output using a twisted-pair ribbon cable.

Event Builder. A Zynq FPGA is used on the VTP to run the standard CLAS12 CODA readout controller (ROC) component, which allows the VTP to be configured and readout the same as other VME/Intel-based CODA components. Event data can be generated by the VTPs that contain the trigger decisions for both the Stage 1 and 3 components, which is used to understand the trigger efficiency. Additionally, there is a large buffer (4 GB with 200 Gbps bandwidth) and 40 Gbps Ethernet interface that is intended for future upgrades of the front-end crate readout system, which would use the VTP and 40 Gbps Ethernet for event readout rather than the VME interface. Fig. 5 shows the interfaces between the FPGAs, memory, network, VXS, and fiber modules.

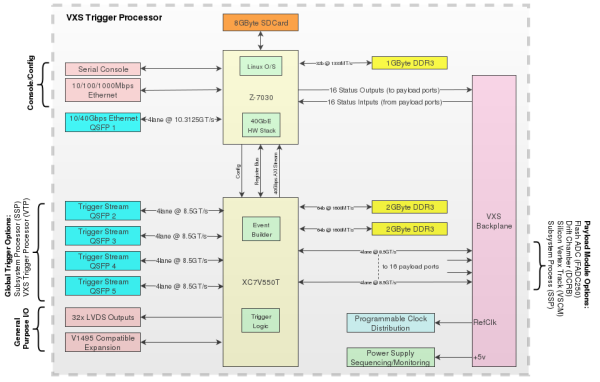


Figure 5: VTP Block Diagram.

4.3. SSP Board

The SubSystem Processor (SSP, see Fig. 6) is a VXS payload card used to collect data from multiple front-end (Stage 1) crates. The SSP performs the Stage 2 trigger processing by creating sector and central trigger bit decisions. Up to 16 SSP modules can be housed in a single VXS crate, but only 7 are currently needed: 6 for the sector-based detectors and 1 for the central detectors. There are 36 full-duplex serial links each capable of running at up to 6.5 Gbps that can be used for transporting the trigger data. The links are bonded in groups of 4 for a total of 8 channels: 1 VXS switch slot interface (copper) and 8 QSFP interfaces (optical). All VXS and QSFP lanes run at 5 Gbps (or 20 Gbps per channel).

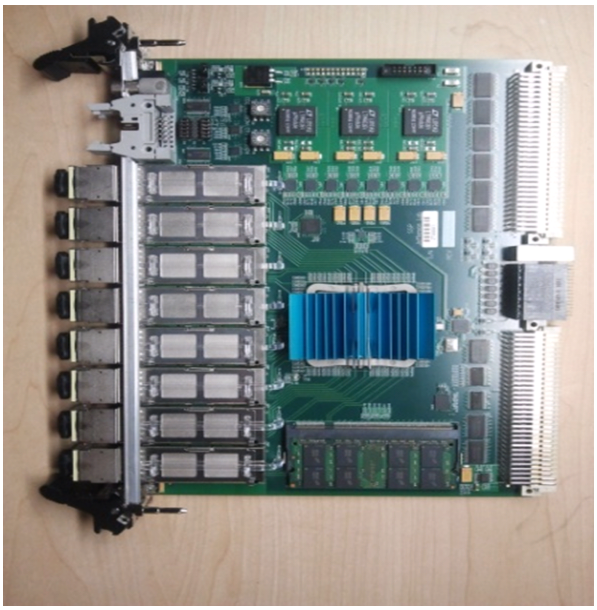


Figure 6: SubSystem Processor (SSP) Board.

Stage 2 Trigger Processing. The Stage 1 optical data arrives at the SSP where it is aligned and processed through various algorithms to make the sector and central trigger bit decisions. There are 8 sector and central trigger bits (expandable to 32) that evaluated at 250 MHz so that no jitter is introduced by this stage. These bits are sent to the stage 3 VTP using the VXS switch serial interface. As for the VTP, the SSP has an Event Builder that allows for readout of the trigger information and its insertion into the data stream.

5. High Level Synthesis in CLAS12 Trigger System Development

A significant portion of the Trigger System components were developed using Vivado High-Level Synthesis (HLS) from Xilinx ([11]. HLS was introduced to reduce the electronics knowledge required to design hardware. It also makes the hardware design flow easier when it comes to achieving a certain behavioral model required by the hardware without worrying too much about the electronics underneath.

5.1. Motivation to use HLS

HLS makes it easier to incorporate well-established data processing algorithms, typically written in C++ or other high-level languages, into FPGA-based projects. HLS allows for scientists to be involved in the CLAS12

Trigger System development who do not have an electronics engineering background. It allows for the involvement of programmers who developed the algorithms for offline data processing but who have limited or no FPGA programming experience. It also makes it possible to validate code with the offline processing framework.

5.2. Trigger Components Implemented with HLS

HLS was used to develop most of the Stage 1 components of the CLAS12 trigger. These include the following elements:

- High Threshold Cherenkov Counter (cluster energy reconstruction);
- Forward and Central Time-of-Flight Counters (clustering and timing correction);
- Electromagnetic and Preshower Calorimeters (cluster energy and position reconstruction).

The Time-of-Flight System and Cherenkov counter trigger implementation was rather straightforward. This typically takes less than 10-15% of the Virtex 7 chip and the timing requirements were easily met.

The calorimeter trigger implementation required much more effort because of its complex nature that requires significant FPGA resources. The details are further explained in the next section using the Electromagnetic Calorimeters as an example.

It should be mentioned that it took a significant amount of time to implement the desired ECAL algorithm, mostly because of the lack of experience in HLS usage. As soon as all important details of the HLS tool were understood, the development process converged, and the trigger components related to various other CLAS12 detectors were implemented in a prompt manner.

5.3. CLAS12 Electromagnetic Calorimeters (ECAL)

Among all of the trigger system elements, the most challenging for the FPGA implementation was the trigger component serving the two CLAS12 electromagnetic calorimeters. Due to their structure, these calorimeters do not provide cluster coordinates or energies without significant event reconstruction. The trigger implementation details are described in Section 3.3.1. Below we describe our experience with HLS using ECAL as an example.

5.4. C++ vs. HLS C++

The FPGA implementation of the ECAL trigger was done in a 125 MHz domain, a balance between speed and resource utilization. The Trigger System components, in general, require a fixed latency, which sets certain constraints on the design. The reconstruction algorithm borrowed from the offline analysis framework was adopted for VIVADO HLS by rewriting it to C++, using HLS streams, HLS pragmas, unrolling for-loops, pipelining, and making all other needed changes. The resulting implementation was tested on simulated data and showed correct results. After that we started to run it through VIVADO HLS and VIVADO tools to address various issues related to generating an FPGA image that met the timing requirements and fit within the resource allotment.

5.5. HLS and HDL

When HLS is used, compiling the design consists of the following main steps:

- VIVADO HLS - convert C++ to Hardware Description Language (HDL);
- VIVADO synthesis - HDL to FPGA primitives;
- VIVADO implementation - map FPGA primitives to chip and route connections.

For large designs, VIVADO HLS will very often report extremely optimistic results that suggest a viable solution, but during VIVADO implementation will fail to meet the timing requirements. To address this the failing paths must be traced back to the HLS component where it can be changed to try to improve the design. It often took many iterations to either find the workable HLS settings, code structure, or clock period adjustment.

5.6. HLS Clock Domain

For the different trigger components related to the different CLAS12 detectors, we use different clock domains between 250 MHz and 31.25 MHz. In the 250 MHz domain, the modules occupying more than 10% of a XC7V550 Xilinx FPGA failed to meet the timing requirements. In the 125 MHz and lower frequency domains, the FPGA utilization was close to 100%. For the ECAL project with a chip utilization of about 70%, the 125 MHz clock was used.

In general, a slower clock speed (31.25 MHz) was preferable for smaller projects where resources were plentiful. When using a slow clock, the HLS code was

able to be written as a single module and had no problem meeting the timing requirements during implementation.

Larger projects, such as for the ECAL, require more efficient use of the FPGA resources and have latency requirements that require a faster clock, but cannot be too fast such that the HLS modules cannot reliably meet the timing requirements. The 125 MHz clock was found to be the optimal middle ground for the "-1" speed grade Virtex7 used in the CLAS12 Trigger System.

5.7. HLS Project Size and Organization

The typical HLS project for the CLAS12 Trigger System contains only a few routines, and uses HLS streams in the function parameter list to communicate easily with the surrounding HDL. That scheme works well for small projects.

For the ECAL with some versions being close to 100% of FPGA utilization, the situation was quite different. The biggest problem we faced was the inability to meet the timing requirements during the implementation (even when HLS reports that the timing is good). HLS uses state machines to schedule the operations it synthesizes. For large HLS components, the generated state machines can have massive control signal fanouts. As the clock period shrinks, so must the maximum signal fanout for the general control signals for a design to reliably meet the timing requirements. For a clock period of 8 ns using a "-1" speed grade Virtex7, each HLS module was kept smaller than the 30K look-up tables (LUTs) (<10% of the LUT resources) to achieve a design that consistently meets the timing requirements.

The original ECAL project consisted of about 20 C++ procedures that occupied most of the FPGA resources - with HLS generating big fanouts on this scale, it was impossible to meet the 8 ns timing on the implementation stage. The workaround was to split the entire project into smaller procedures, glued together in HDL by using well defined, simple interfaces between the separate procedures. Still, some procedures were too big, especially for the sorting algorithms. We were able to split some procedures further until finally the entire project met the timing requirements and the resulting FPGA image was loaded into the hardware.

After every significant change, we re-tested the code on simulated data, making sure it still produced correct results. The chart in Fig. 7 shows how many HLS projects were created in the end.

Another reason for subdividing the project is the lack of multi-clock domain support. Since the Event Builder in the VTP board works on a 250 MHz domain and most projects use a slower clock, every project was subdivided and separate pieces communicated over the HDL-written interface. The necessity of subdividing HLS projects and of using HDL to assemble them together, is probably the most restricting feature in HLS usage. Much of this subdividing can be reduced by using the HLS DATAFLOW directives (which can isolate functions communicating through registered FIFO interfaces), but it requires further code restructuring to be compatible with this flow and does not support multiple clock domains.

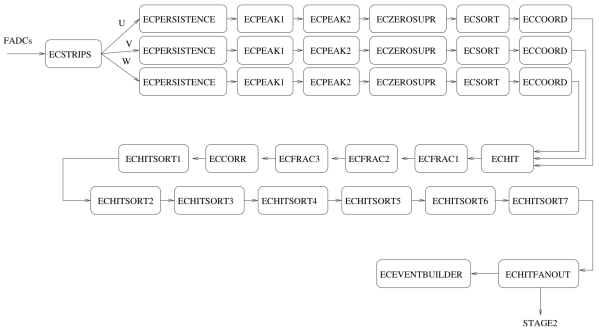


Figure 7: ECAL HLS project chart.

5.8. HLS Versions and Cross-project Dependences

As mentioned before, splitting the project into smaller pieces allowed us to meet the timing requirements. This worked, in particular, because we were able to eliminate combinatorial paths between the HLS projects connected by the streams. Such dependences can be clearly seen looking into the schematics for the failed timing chains, and were usually related to the large state machine control signals going between modules. Initially we used HLS version 2015, and despite all of our efforts, we could not eliminate these long combinatorial paths across modules. This was resolved after switching to HLS version 2017, where the streams could be fully registered (with pragma "axis register both port="). This meant that if the registered HLS streams were used between separate HLS projects, then the state machine paths were also registered between modules. With that, it was only a matter of splitting projects into smaller pieces to improve/meet the timing requirements.

5.9. HLS Settings

The clock uncertainty is set as 30% of the main clock, which we found forces HLS to produce more realistic timing estimates. A single HLS project often cannot exceed several percent of the flip-flops (FF) and LUT

budget, otherwise it may be a problem to meet the timing requirement on the VIVADO implementation step. A typical HLS project for one of the PCAL trigger elements is shown in Fig. 8.

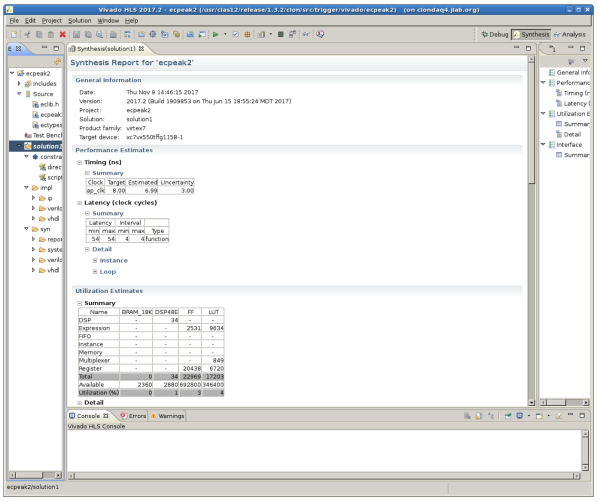


Figure 8: Typical HLS project for one of the PCAL trigger elements. 4% LUTs is close to the maximum possible to meet timing requirements on the following steps.

5.10. VIVADO Settings

Common settings for VIVADO were used as shown in Fig. 9. It usually takes 3+ hours to compile the PCAL project on a Dell R730 server under RHEL7. For some firmware versions, we were able to utilize 100% of the LUTs and still meet the timing requirements if the clock domain was 125 MHz or lower. The VIVADO project for the PCAL trigger is shown in Fig. 9.

5.11. Firmware Validation for HLS-based Projects

The ability to validate the firmware using a C++ implementation is the one of the biggest advantages of HLS. During the course of development and commissioning, we ran HLS C++ code on simulated and real data from the CLAS12 detectors, implementing required features and fixing bugs. During data taking we were able to find and fix observed problems or add new features in several hours, which was very important to save beam time.

5.12. Our Conclusion about HLS Usage

The CLAS12 ECAL and other detectors were successfully implemented into the Trigger System using HLS to produce the core part of the firmware. This trigger was used in the first physics run in 2018 and worked

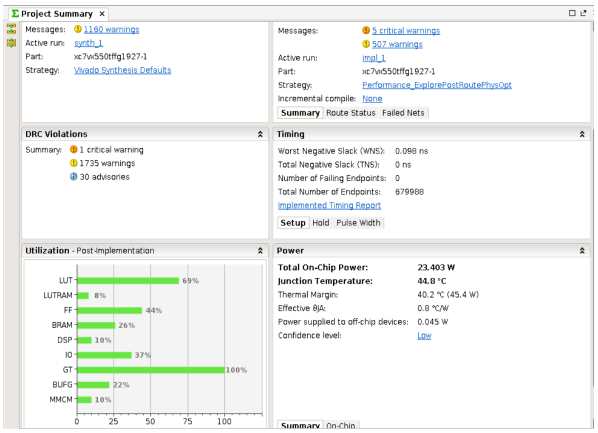


Figure 9: VIVADO project for PCAL trigger. Strategies used were "Vivado Synthesis Defaults" and "Performance_ExplorePostRoutePhysOpt". LUT utilization is about 2/3 of the total balance.

as expected. We were able to select events based on individual ECAL cluster energy, something which was possible before only during offline data processing.

HLS in general appears to be a useful tool, especially to implement smaller trigger components like the Cherenkov or Time-of-Flight counters. For components utilizing a significant portion of the FPGA, it will benefit development significantly to improve HLS in the following directions:

- Support multi-clock domains;
- Improve subroutine calls by allowing the option to fully register paths between modules;
- Improve state machine logic, for example support streams between routines inside the project and be able to generate separate state machines for separate routines. This will allow for the avoidance of splitting the project manually and using HDL as a top interface as we are currently forced to do.

6. Software

This section describes the various software tools used during the CLAS12 Trigger System development, validation, and operation.

6.1. Development Software

Several software packages were used to implement and test the trigger logic developed for CLAS12. These were the FPGA synthesis and implementation packages, and the FPGA high-level synthesizer.

For FPGA synthesis and implemenation, Xilinx ISE/Planahead and Vivado were used. Most of the front-end boards were developed years ago and use Virtex 5 and Spartan 6 FPGAs, which are not supported by Xilinx Vivado, so we relied on Xilinx ISE/Planahead for synthesis and implementation. Even though these tools are no longer updated by Xilinx, they have proved to be stable, reliable, and deliver consistent results. Newer designs use Xilinx 7-series parts (we used Artix, Kintex, and Virtex), so we used the Vivado tools, which had far better support than ISE/Planahead.

Vivado HLS was used to implement a variety of trigger algorithms, Event Builder logic, and general purpose logic. HLS components were able to be verified with C/C++ test benches on their own without anything more than GNU GCC and associated C/C++ header files from the HLS toolchain. This tool often allowed for faster FPGA implemenation, but many components still were implemented in HDL where resource and/or timing requirements become critical. Vivado HLS generated VHDL files from the C/C++ sources that were included into the FPGA compilation and full simulations. Occasionally simulation of the HDL generated files was required to debug C/C++ modules when simulation under GCC found no problem, but the HDL result did. Differences between the C/C++ simulation and the corresponding HDL output were primarily due to: C/C++ assumption of infinite length buffering while HDL buffers were finite, ambiguous C/C++ coding where GCC and HLS behaviors differed, and latency issues due to the C/C++ simulation having no concept of elapsed time. Our experience with HLS is described in detail in Section 5.

6.2. Operating Systems

Linux operating systems are used on all readout controllers in the CLAS12 DAQ. The VME crate controllers use Intel-based CPUs and run a standard Centos and Linux kernel distribution. The VTP modules use an ARMv7 CPU with custom hardware and run Arch Linux using a custom Linux kernel. Both CPUs boot from the network using a shared kernel and root filesystem images, which simplifies administration. Common DAQ software (readout, configuration, slow control, diagnostics, etc.) tools are used for both platforms and most tools available on standard desktop PCs are also available.

Details on the installation of Arch Linux with ARMv7 are as following:

- Linux kernel 4.4.0
  - Updates with specific support for Xilinx Zynq Processors
  - Available at https://github.com/Xilinx/linux-xlnx.git
  - Custom device-tree
  - Provides FPGA programming interface
  - Allocates physical memory for use with DMA and event buffers
  - Standard I^2c and SPI API
- Arch Linux compiled for ARMv7
  - Available at https://archlinuxarm.org/
  - Filesystem over NFS
  - diskless booting using tftp (UBoot)

6.3. Configuration Software

6.3.1. Configuration Files

The CLAS12 Trigger System has a large number of parameters controlling its logic. Those parameters are set by writing values to hardware registers, and are controlled by reading those registers back. The system uses ASCII configuration files, as shown for example in (Fig.10). Every line in these configuration files contains a key word and the number of corresponding parameter values. The directive “include” can be used to create hierarchical set of configuration files. Normally the main configuration file is selected during the run startup procedure, and the CLAS12 run control software resolves all “include” directives, resulting in the creation of one big configuration file. That file is used to program all trigger hardware registers, and its content is also written to the data stream for bookkeeping purposes. The register contents are read back and the results are recorded into the data stream as well, providing full control of the Trigger System settings. Normally the same configuration files contain the DAQ settings as well, making it a complete source for the entire DAQ/Trigger System settings.

6.3.2. Timing Settings

One of the important aspects in setting up the trigger is the measurement of the relative timing between the signals from the detector elements used to define a trigger. These are referred to as delay curves. For this purpose software procedures were developed. This includes special trigger configuration files and software tools to scan individual subsystem latencies, record sets of beam current normalized scalers, and produce the corresponding delay plots (see Fig. 11. The trigger time

```
VTP_CRATE adcecal1vtp # trigger stage 1
VTP_ECINNER_HIT_EMIN     100 # strip energy threshold for 1-dim peak search
VTP_ECINNER_HIT_DT       8 # +-8clk = +-32ns: strip coincidence window to form 1-dim peaks,
                           # and peak-coincidence window to form 2-dim clusters
VTP_ECINNER_HIT_DALITZ   568    584   # x8: 71<dalitz<73
.....


SSP_CRATE trig2 # trigger stage 2
SSP_SLOT all
SSP_GT_STRG_ECALIN_CLUSTER_EMIN_EN    0  1 # enable ECAL_INNER clusters in trigger
SSP_GT_STRG_ECALIN_CLUSTER_EMIN       0  2000 # ECAL_INNER cluster threshold
SSP_GT_STRG_ECALIN_CLUSTER_WIDTH      0  100 # coincidence window to coincide with PCAL etc
.....


VTP_CRATE trig2vtp  # trigger stage 3
#       slot:              10 13  09 14  08 15  07 16  06 17  05 18  04 19  03 20
#     payload:             01  02  03  04  05  06  07  08  09 10 11 12 13 14 15 16
VTP_PAYLOAD_EN  0   0   0   0   1   0   1   0   1   0   1   0   1   0   1   0
# 6780 corresponds to 7900 FADC latency
VTP_GT_LATENCY 6780
# sector bits: trig number, ssp trig mask, ssp sector mask, multiplicity,
# coincidence=number_of_extended_clock_cycles
VTP_GT_TRGBIT 8 1 1 1 1
.....
```

Figure 10: Configuration file example.

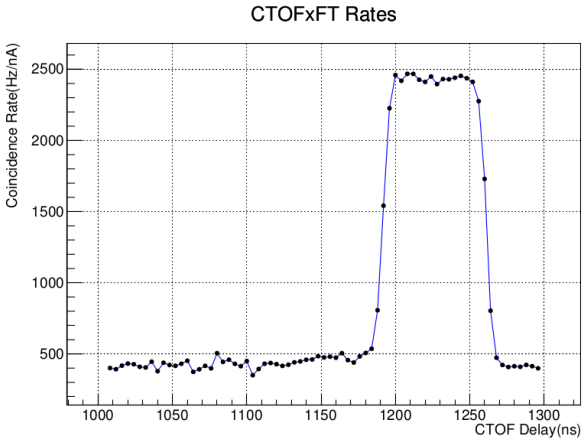


Figure 11: CTOFxFT Delay Scan.

setting for a specific detector element was kept at a constant value, which determined the DAQ readout time window width and offset, while the timing for the other subsystems were changed step by step to monitor the delay curve. Delays and coincidence widths were adjusted to account for known jitter sources to ensure no events were lost due to poor timing alignment. This procedure was repeated every time the trigger logic was changed.

6.3.3. Gain Calibration and Threshold Settings

One of the important settings in the Trigger System is for the FADCs. As stated above, the FADC boards serve as the pre-trigger for most of the Stage 1 trigger components (except the Drift Chambers), and correct pedestal and gain calibrations for these units are critical for correct Trigger System performance. Pedestal and gain measurements are conducted before run startup, and the values are loaded using configuration files. All of the thresholds in the configuration files are set using physical units such as MeV for the calorimeter energy and the number of photoelectrons for the Cherenkov counter.

6.4. Readout and Control Software

All trigger hardware modules were implemented in VXS format and installed into VXS crates along with the other DAQ electronics. All readout and control libraries were developed as part of the DAQ System software project as described in [1]. From the software point of view, the DAQ and Trigger Systems can be considered as one system equipped with standard software tools.

7. Simulation Tools

Several simulation tools were used in the development of the CLAS12 Trigger System. These tools were very useful during the development and implementation stages, and are still in use now for continued validation. The primary simulation tools for the CLAS12 Trigger System are detailed below.

7.1. GEMC/GEANT4 Simulation Tool

In the beginning of the CLAS12 Trigger System development, the hardware was not available, so input data were generated by the CLAS12 Geant4 Monte-Carlo package (GEMC, **??**. This package was used to produce data files with data banks in the same form as produced by the DAQ. The Trigger System software includes a playback package that is able to read GEMC-generated files and produce FADC and DCRB responces identical to the those from the hardware. All Stage 1 trigger components implemented with HLS were developed using simulated data, including for the most complex responces in the calorimeter and drift chambers. For VHDL-written components, simulated data were used as well along with other specialized tools described below.

For example, Fig. 12 and Fig. 13 show a comparison of the energy and coordinates of the EC clusters reconstructed by the offline analysis software and by the Trigger System. Different dividing methods were used in the Trigger System, including one based on digital signal processing (DSP) and another based on using lookup tables. The first method provides better results but requires more resources than the second method. Over the course of system development, many decisions were made based on similar comparisons.

Another example (see Fig. 14) shows the absolute EC cluster energy obtained by offline analysis and by the

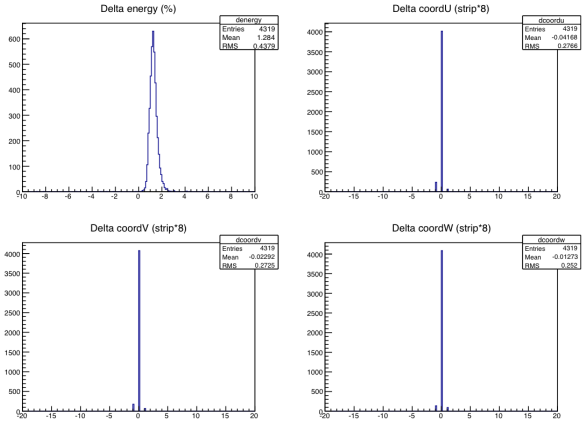


Figure 12: EC cluster finding: difference between results from offline reconstruction and trigger system (using dividing in coordinate calculation).

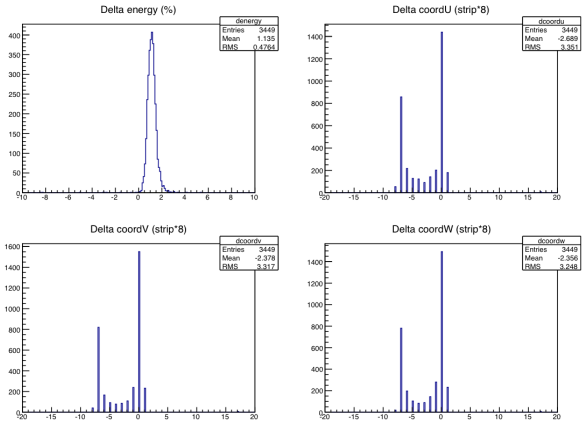


Figure 13: EC cluster finding: difference between results from offline reconstruction and trigger system (using lookup table in coordinate calculation).

Trigger System. As can be seen, the Trigger System provides the same result as predicted by simulation and obtained from offlne analysis.

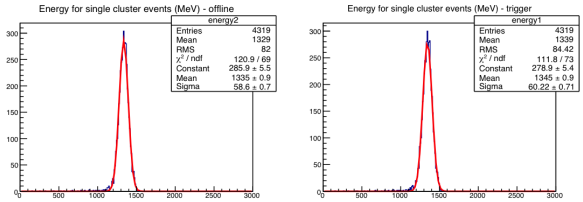


Figure 14: EC cluster finding: 5GeV electron, no PCAL, energy deposition in offline and trigger model.

7.2. FPGA Simulation Tools

A cycle accurate simulator was setup to model, test, and debug the full Trigger System using Aldec Riviera. This tool is able to perform simulations of the full Trigger System in a mixed language environment (VHDL, Verilog, C/C++) for all FPGA components used in CLAS12 (Xilinx series 5, 6,and 7 components). VHPI was used to interface external C/C++ programs from the DAQ to the VHDL test bench. Using VHPI, calls to the DAQ EVIO (the native DAQ data format) C/C++ libraries were possible from VHDL, making it possible to feed detector waveform data into the trigger simulation directly from Monte Carlo, data files taken with beam, or from user-created text files. Simulation intensive components, primarily gigabit SerDes components, were replaced with fast, simple models once verified to improve the simulation performance. The simulator is single-threaded and requires a license for each running instance, making it costly to parallelize. Even so, it was capable of processing 1 event every 30 s on a typical desktop PC simulating the CLAS12 forward and central Trigger System comprised of 24,192 drift chamber wires and about 3400 FADC channels.

The simulation was built from the actual HDL firmware source files compiled for the various modules used in the trigger system (DCRB, FADC, VTP, SSP). HDL wrappers were created to model the VXS crates which include: backplane, fiber interconnect, trigger

System	Crates	ModType	ModCnt	ChCnt
DC	18	DCRB	252	24192
ECAL	6	FADC	84	1296
PCAL	6	FADC	96	1152
FTOF	6	FADC	36	576
FTCAL	2	FADC	21	332
FTHODO	1	FADC	15	232
CND	1	FADC	9	144
CTOF	1	FADC	6	96
HTCC	1	FADC	3	48
GT	1	SSP	6	28 (Fiber)

Table 5: Trigger Simulation crates

distribution, clock distribution, configuration, and readout. The Table 5 summarizes the trigger simulation components:

Each of the front-end crates uses a VTP trigger module that runs the detector-specific trigger algorithm. The front-end VTP modules feed the trigger data to the global trigger (GT) crate Stage 2 (SSP). The SSP modules feed the trigger data into the final trigger Stage 3 (GTP). There are 10 different VTP firmware types to support the Stage 1 (front-end) and Stage 3 (GTP) algorithm. There are 2 different SSP firmware types that support the Stage 2 CLAS12 Forward and Central Detector trigger logic.

This full simulation is primarily run for two scenarios. The first is whenever a significant firmware change is made. In this case a small number of specially selected events (about 2k) can be fed through to tag the trigger decisions on each. The second is whenever the DAQ system records events where the trigger failed to properly tag them (typically during a random trigger run to assess the efficiency). For both cases the failed events can be loaded into the simulation and the failed decisions can be explored in detail to determine the cause. The first scenario takes one day or more depending on the number of events needed to check, while the second case can take minutes, since only failed events are presented to the simulation so problems can immediately begin to be examined.

8. Trigger System Firmware Development

After generic Trigger System design was complete and the hardware components entered their production stage, the firmware development began. Both HLS and VHDL tools were used. Work was performed using data samples generated by the CLAS12 GEMC/GEANT4 package and cosmic data after the hardware components were installed. This section describes our procedures. Additional development and validation with beam are described in Section 10.

8.1. Preparation of Simulated Data Sets

Simulated data sets for Trigger System development were prepared using GEMC (GEANT4-based CLAS12 simulation package, **??**. GEMC has a fully realistic CLAS12 geometry description and complete maps of the magnetic field, and produces digitized results suitable to be converted into the pre-trigger data format.

Various data sets were generated depending on what was needed for the development of particular trigger components. For example, fixed energy single electron sets were produced for the initial development of the EC and PCAL components. For these data sets, all detectors positioned upstream of the EC or PCAL were disabled to make sure the single electron directly hit the EC or PCAL. In this way the cluster finding algorithm could be developed and tested in ideal conditions. After that, realistic data samples were produced and the algorithm was tested again.

Another data set was used to create the road dictionary for the Drift Chamber-based trigger component. For this purpose, positive and negative tracks were generated uniformly in a selected momentum, θ and ϕ range and tracked through the CLAS12 detector to determine the list of DC wires involved by the particle trajectory.

8.2. Development Using Simulated Data

The trigger development process consisted of several methods and that depended on the nature of the trigger component. Most Stage 1 components were implemented using the HLS/VIVADO tool, where the firmware was written using an HLS C++ extension. In that case, it was possible to develop and validate the firmware as part of the offline reconstruction framework using a standard desktop computer. Usually the offline processing alghorithms were re-written using HLS/C++, with appropriate simplification and structural changes to make it suitable for the FPGA firmware. Simulated data were used as input, which were processed directly by the HLS/C++ code and compared with the initial simulation parameters. In addition, the same samples were processed by the offline reconstruction software and the results were compared with the trigger output. This double-check method practically guarantees bug-free implementation. There was no single case when the C++ implementation passed tests on

the simulated data and then failed during the final validation stage. The most complicated Stage 1 components were developed and tested using this method.

Several components of the trigger were written mostly in VHDL and initially no software existed for feeding GEMC data into the HDL simulations. This was the case for the Stage 1 FT and DC tracking trigger, as well as the Stage 2 and Stage 3 components of the trigger. These modules relied on standard VHDL test benches to feed/generate test vectors for evaluating the correctness of the design modules. For example, the FT test bench generated clusters at each position of the calorimeter and hodoscope to test the channel mapping and geometry matching. Additional specific test cases verified the FT trigger clustering time coincidence, cluster multiplicity, and latency to ensure it operated as expected. C/C++ modules were written that emulated the FT the DC tracking trigger so the algorithms could work in the same offline framework as described above for the other Stage 1 components.

8.3. Development and Validation Using Cosmic Data

When the hardware components for the CLAS12 detector were constructed and mostly installed, and first version of the firmware was ready for testing, all three Trigger System firmware stages were loaded and development continued for the entire Trigger System using cosmic data. At that point we started to perform Trigger System validation for some components, while development was continued for others, as described in the following sections.

8.3.1. Alternative 'Hit-Based' Trigger System

The CLAS12 detector inheritated some components from the original CLAS detector (see [3]), in particular its Trigger System. That system was fed by TDC/discriminator boards and was able to produce "hit-based" information only. We decided to keep it for reference purposes as an alternative to the new Trigger System. It was used during CLAS12 cosmic data detector calibrations and validation of the new Trigger System up to the point when new Trigger System was ready. It is still operational and can be used to double-check the main CLAS12 Trigger System if needed.

8.3.2. Development and Validation of EC/PCAL Special Purpose Trigger with Cosmic Data

The first detector calibrations employed cosmic data. Here we will describe, as an example, one of the procedures related to EC/PCAL calibration needed for correct Trigger System performance. Similar procedures were executed for all detectors participating in the Trigger System.

The efficiency and spatial uniformity of the cluster finding trigger in the EC/PCAL described in Section 3.3.1 requires already calibrated calorimeters with predetermined PMT gain and light attenuation constants loaded into the VTP/FPGA trigger firmware. Calibration runs using a special purpose MIP trigger were used to obtain these constants. For that purpose a so-called "pixel trigger" was developed and loaded into the Stage 1 firmware along with the main trigger, so it was possible to calibrate the system using a pixel trigger and then switch to the main one for data taking. This "pixel trigger" used a simple multiplicity condition on 1D cluster size for each U,V,W view to reject undesirable muon trajectories and select normally incident tracks. This reduced the trigger and data rate by 95% and ensured the same MIP energy was deposited for all possible triple intersections of single strips.

The pixel trigger pipeline executes these steps in parallel, with user configurable parameters in bold: 1) If FADC hit energy > **EMIN**, make a pulse **HITWIDTH***4 ns for that strip. 2) Look for coincidence of U,V,W pixel strip candidates from step 1. 3) Evaluate multiplicity **EVALDELAY***4 ns clock cycles after the leading edge of a candidate pixel from step 2. 4) Generate a pixel trigger if the multiplicity requirement is met and we still have a hit on U,V, and W.

Additional configurable trigger elements were introduced, including a total energy sum threshold **ESUM** and a lookup table for triplets of strips that satisfy the geometrical constraint $dU+dV+dW$ = **DALITZ**, where d is the normalized distance to the hit strip indicated by the arrows in Fig.15 and **DALITZ** = 2 for perfect pixels. The latter test was sometimes necessary to prevent noisy PMTs from saturating the multiplicity (N=3) trigger condition. Offline analysis showed that about 90% of pixel triggers satisfied the Dalitz test (Fig.16), while adjacent calorimeter elements that did not use the trigger had a much smaller pixel fraction. This suggests the pixel trigger helps to suppress events that undergo multiple scattering, which would trigger adjacent strips and violate the multiplicity requirement.

8.3.3. Development and Validation of Drift Chamber Component of the Trigger System with Cosmic Data

8.3.4. Development and Validation of Entire Trigger System with Cosmic Data

While the Stage 1 trigger components were validated separately from each other during the development stage, the Stage 2 and Stage 3 components requires

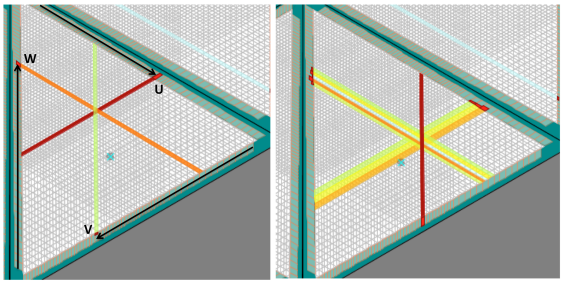


Figure 15: Examples of clusters from cosmic muon triggers. Desired trajectory (left) is normally incident on the face of PCAL and satisfies the single pixel multiplicity condition (N_u=N_v=N_w=1) in the FPGA pixel trigger. Event at right shows a more vertical trajectory rejected by this trigger.

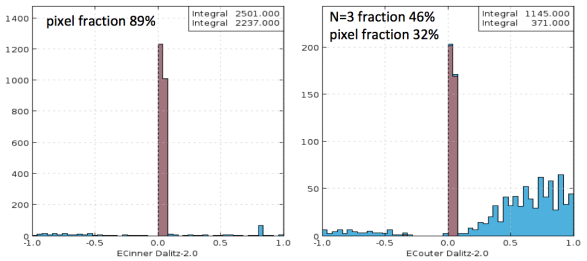


Figure 16: Offline analysis of events which satisfied the pixel trigger on ECINNER calorimeter. Left plot shows 89% of ECINNER triggers satisfied the pixel test $dU + dV + dW = 2$. Right plot shows only 14% of the ECINNER triggers found an ECOUTER event that satisfied both the $N = 3$ and pixel test.

the entire system to be assembled to perform validation. Initially those two stages were programmed with simplified alghorithms to test signal propagation and basic trigger system functionality, and only the timing coincidence between different detectors was implemented. Development of Stage 2 and Stage 3 continued during cosmic run operations and later with beam operations, adding a geometrical matches between different detectors and increasing the coincidence logic complexity.

8.3.5. *Trigger System Flexibility and "Permanent Development" Mode*

The initial plan was to develop the Trigger System firmware to satisfy all CLAS12 experiments for the entire duration of CLAS12 operation, meaning high (close to 100%) trigger efficiency and reasonable purity. As the power and flexibility of the trigger system was revealed to the community, additional requirements were requested to improve the system purity, and to include additional physics processes. As a result, the Stage 2 and Stage 3 components of the Trigger System were under constant development during the first year of CLAS12 operation, and the firmware was upgraded and the entire system was validated after every change. After a while, the Trigger System reached the point when relatively small improvements in the trigger purity could only be achieved with significant efforts. At that point the development was declared complete. The nature of the FPGA-based Trigger System allows almost endless improvements, but such "permanent development" mode is not practical.

9. Physics Triggers

9.1. *Electron Trigger*

The electron trigger is designed to select the inclusive electron scattering from the CLAS12 targets:

$$e(p, n, A) \rightarrow e'X. \quad (1)$$

The trigger selects events with at least one scattered electron detected by the forward detectors. The High Threshold Cherenkov Counter (HTCC), Preshower Calorimeter (PCAL), Electromagnetic Calorimeter (EC), and Drift Chambers (DC) participate in the generation of the trigger decision. Searching for the electrons is performed in all six CLAS12 sectors in parallel. The final electron trigger is a simple "OR" of the six sector trigger signals.

The HTCC discriminates electrons from other charged particles. This detector must be calibrated in terms of the number of photoelectrons before the start of any experiment. The HTCC trigger logic is searches for clusters and calculates the total number of photoelectrons detected by HTCC. The cluster may include up to four PMT signals that collect the Cherenkov light from the adjacent mirrors as described in Section 3.3.2. The minimum number of photoelectrons in the cluster is one of the main electron trigger parameters. Usually this parameter equals 1-2 photoelectrons depending on the experiment requirements.

The PCAL and EC calorimeters are designed to detect photons and electrons as described in Section 3.3.1. The high energy deposition in the calorimeters is a signature of electron detection, and is one of the electron trigger parameters. The PCAL and EC detectors must be calibrated before the start of any experiment in terms of energy deposition measured in MeV. The electron trigger uses cuts on the cluster energy in the PCAL (E_{PCAL}) and EC (E_{EC}) separately, and cuts on the total energy deposition in both detectors $E_{Total} = E_{PCAL} + E_{EC}$. These cuts depend on the beam energy and the experiment requirements, and usually lie in the region from 150-300 MeV (corresponding to minimum

electron energy from 600-1200 MeV) for the energy sum E_{Total}.

Geometrical matching between the HTCC signal and the position of the shower in the PCAL calorimeter helps to suppress random coincidences between two detectors. The trigger firmware uses the HTCC-PCAL lookup table to make a proper event selection.

The track reconstruction in the DC system at the trigger level is very useful for the further suppression of accidental background, as described in Section 3.3.3. The trigger decision requires at least 3 layers in every superlayer and at least 5 superlayers in every road, which is a standard setting for all triggers where the DC-based component is used. The geometrical matching between track candidates and hits in the PCAL and EC detectors is used to strengthen the trigger performance in terms of event purity.

The electron trigger configuration may be represented by the formula:

$$HTCC_i(N_{phe}>N_{min}^{HTCC})\times \\ [E_{PCAL}>E_{min}^{PCAL}) \times E_{Total}>E_{min}^{Total}) \times DC]_i \quad (2)$$

where index i is the CLAS12 sector number, and N_{phe} is the number of photoelectrons detected by the HTCC in a defined cluster. N_{min}^{HTCC}, E_{min}^{PCAL}, E_{min}^{Total} are trigger parameters, and DC means that a track was reconstructed by the DC-system. The space correlations between all detectors and coordinates of the track are implemented as well.

9.2. Photoproduction Trigger

The photoproduction trigger is designed to select events where a scattered electron is detected by the Forward Tagger in the polar angular range from 2° to 5°. Strictly speaking it is not a photoproduction process but electron scattering with low four-momentum transfer $Q^2 = 4E_{beam}E' \sin^2\theta/2$. The trigger logic continuously searches for the clusters in the FT calorimeter from electromagnetic shower, and calculates the shower energy and space coordinates. The cluster energy is the sum of all thecrystal energies within a 3x3 spatial array that meet the time matching constraints. Once the clustering algorithm has identified a cluster, the corresponding data is reported to the next trigger stage. This includes the timestamp, the energy, and the spatial coordinates (center of the seed crystal). The cluster energy is not corrected for shower leakage effects at this stage. Finally, the trigger processor makes the trigger decision by applying further cuts to the clusters. The trigger selection is based on lower and upper energy limits and the number of hits in the cluster. The trigger may also select events with a specified number of clusters detected by the calorimeter. The coincidence with the two-plane scintillating hodoscope located in front of the calorimeter serves to discriminate charged particles from high energy photons. This also provides for the possibility to select reactions with an electron and several photons in the final state, for example

$$ep \rightarrow e'\gamma\gamma X.$$

The trigger system may use the information from the CLAS12 Forward and Central Detectors to select events with several charged or neutral particles in coincidence with the electron in the FT calorimeter. The trigger detector composition depends on the reaction under study.

Charged particles in the Forward Detectors are selected by a coincidence between the FTOF, PCAL, and EC with tracks reconstructed by the DC system. The space correlation between all trigger detectors are required, including coordinates of tracks crossing the detector planes. Hit matching along the track is an important part of the background reduction at the trigger level. The cuts on the energy depositions in the trigger detectors are used to select charged and neutral particles.

The following trigger configuration

$$FT(E_{min}^{FT}<E<E_{max}^{FT})\times \\ [FTOF(E>E_{min}^{FTOF}) \times PCAL(E>E_{min}^{PCAL}) \times DC]_i$$

was used in the first CLAS12 experiments to select the reaction $ep \rightarrow e'h^{+/-}X$ with at least one electron and one charged particle in the final state. The index i denotes the CLAS12 sector number. Each detector has his own trigger energy cuts: E_{min}^{FT}, E_{max}^{FT}, E_{min}^{FTOF}, and E_{min}^{PCAL}. A space correlation matching requirement between the FTOF and PCAL elements was implemented. The trigger rate was too high for the available DAQ bandwidth, so this trigger was prescaled.

The selection of the events with at least two charged particles detected in the forward direction in different sectors was done by the trigger configuration

$$FT(E_{min}^{FT}<E<E_{max}^{FT})\times \\ [FTOF(E>E_{min}^{FTOF}) \times PCAL(E>E_{min}^{PCAL}) \times DC]_i\times \\ [FTOF(E>E_{min}^{FTOF}) \times PCAL(E>E_{min}^{PCAL}) \times DC]_j,$$

where i and j denote different CLAS12 sectors. This trigger selects events with an electron detected by the FT calorimeter and two charged particles in different CLAS12 sectors.

The central detectors, such as Central Time-of-Flight (CTOF) and Central Neutral Detector (CND), were used

for the selection of the events with at least one particle detected in the Central Detector. The following trigger configuration

$$FT(E^{FT}_{min}<E<E^{FT}_{max})\times$$
$$[FTOF(E>E^{FTOF}_{min})\times PCAL(E>E^{PCAL}_{min})\times DC]_i\times$$
$$CTOF(E>E^{CTOF}_{min})$$

was used for the selection of events with an electron in the FT, at least one charged paticle going in the forward direction, and at least one particle detected in the central detectors. In case the trigger rate is too high for the available DAQ bandwidth the CND detector could be added to the coincidence chain with the space correlation between the CTOF and CND counters as:

$$FT(E^{FT}_{min}<E<E^{FT}_{max})\times PCAL(E>E^{PCAL}_{min})\times DC]_i\times$$
$$CTOF(E>E^{CTOF}_{min})\times CND(E>E^{CND}_{min}).$$

As stated above, the minimum energy depositions in all detectors in the trigger are parameters that depend on the individual experiment requirements.

9.3. J/ψ Meson Trigger

A special trigger was designed to detect the quasi-photoproduction of J/ψ-mesons

$$ep \rightarrow e' J/\psi p'.$$

Two decay modes are useful for the selection of the J/ψ meson: $J/\psi \rightarrow e^+e^-$ and $J/\psi \rightarrow \mu^+\mu^-$. The conventional electron and photoproduction triggers select the J/ψ-meson in case of the decay to the electron-positron pair. However these trigger configurations do not work with muons in the final state. Therefore, another trigger was added to select one more decay mode for this experiment. The CLAS12 spectrometer has no dedicated muon system, but it turns out that the selection of particles with energy deposition in the PCAL-EC calorimeters close to the minimum-ionizing value is sufficient to suppress the background from pions when the invariant mass of the two particles (muons or pions) is near the J/ψ-mass. The muons from the J/ψ decay appear in opposite CLAS12 sectors, which allowed for the trigger configuration:

$$[FTOF(E>5)\times PCAL(15<E<60)\times EC(60<E<120)\times DC]_i\times$$
$$[FTOF(E>5)\times PCAL(15<E<60)\times EC(60<E<120)\times DC]_j$$

The energy units are in MeV. Note, that there is no requirement to search for the scattered electron at all. This gives an order of magnitude advantage in the virtual photon flux in comparison with the case when the electron is detected in the FT-calorimeter.

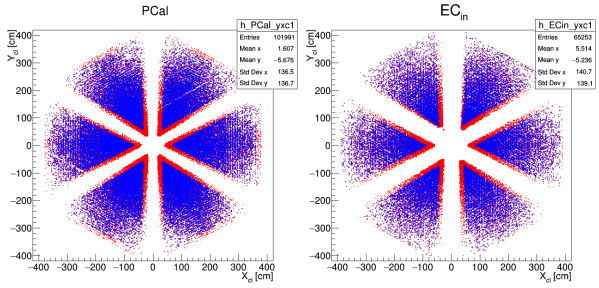
10. Trigger System Validation with Beam

When beam operations started, the Trigger System validation was completed as part of the entire CLAS12 detector commissioning. This section describes the trigger validation procedures.

10.1. "Random Trigger" Validation Procedure

The ultimate validation of the trigger is done using the so-called "Random Trigger" (RT) runs. RT runs are special runs where the event readout is initiated not by the trigger logic, but by an external random generator that can be tuned to the desired frequency. Most of the events in the RT runs do not contain any tracks, however, a small fraction of the events will have real particles that were reconstructed because particle accidentally fell in the readout window that was initiated by the random generator. In the event readout, in addition to various data banks from DAQ system, the trigger decisions are stored as well (see Section 3.6).

These accidental "good" events are used to check whether the desired trigger bit in the Stage 3 32-bit trigger mask was set by the trigger logic. In case it is not set, information from the Stage 1 and Stage 2 trigger is available to analyse the possible reasons for the inefficiency.



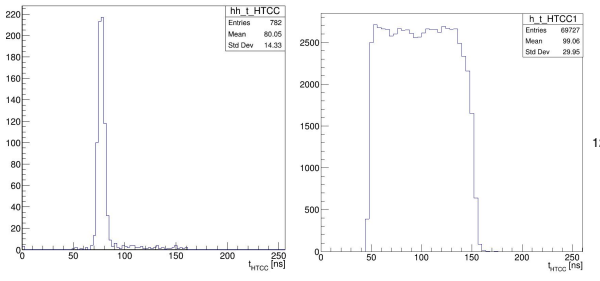
(a) (b)

Figure 17: Distribution of cluster coordinates of PCal (left) and EC_in (right). scatter plot in Red shows all events, while blue scatter plot show events where cluster is in the fiducial region of the calorimeter (about 15 cm away from the edges).

The technique of the trigger validation is as follows. The trigger logic is configured exactly as it will be set in an experiment, but it runs in "tagging mode", reporting trigger decisions into the data stream for every randomly generated event. After several hours of running we collect at least 100 million events.

After the data is processed and the events are reconstructed, we select subset of events with correct trigger

time. It is done using FADC spectra for detectors participating in the trigger logic. We need to select events with the FADC spectra similar to the one in the data obtained using regular trigger. Fig. 18 (left) shows typical FADC pulses for the regular (not random) trigger, with pulse width below 50 ns. Reconstructing and analyzing the data obtained using random trigger, we select events with signal in the middle of the FADC window to make sure we do not have boundary effects when the signal region is selected. Based on the typical pulse shape, we ignore areas below 50 ns and above 150 ns (see Fig. 18 (right)).



(a) (b)

Figure 18: HTCC FADC pulses: left - physics triggers, right - randon trigger

We typically find several thousands events that accidently fell into the correct trigger window. These events can be used to obtain the trigger efficiency and purity assuming that our off-line reconstruction software works correctly. It should be mentioned that correctly working of the off-line reconstruction is an important prerequisite for complete trigger validation.

10.2. Validation of the Electron Trigger

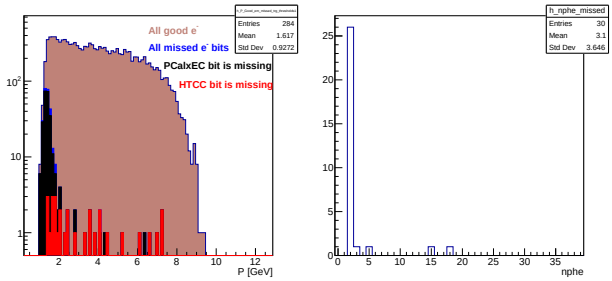
As a reminder the electron trigger logic uses responces from PCal, EC, HTCC and DC (see eq. 2), and as it was described in section 10.1, for trigger validation we have used a RT data. The 1st step in the validation of electron trigger is a selection of events with "clean electron". The CLAS12 offline reconstruction software assignes PID to each reconstruction particle (reference to offline recon.) (for electron PID=11) however in this studies we imposed additional cuts. In particular

- DC roads are optimized for tracks originating from the target, that is why in the offline analysis we have put a cut on the vertex "Z" coordinate to make sure the track originates from the target.

- Selected events where electron hits calorimeters (PCal and EC) in a fiducial region, to make sure the shower is fully reconstructed.

- Applied trigger condition cuts on offline cluster energies in the PCal and EC, also on number of photoelectrons in HTCC.

After applying above mentioned cuts, for each of these electrons, we have checked whether the electron trigger bit is set for the corresponding sector. At the end the trigger efficiency is defined as the number of "Bit Not Set" events over the number of all events with "clean" electron. Both RG-A and RG-B experiments required to have a good (close to 100%) trigger efficiency for electron above 2 GeV. Since both PCal and EC are sampling calorimeters, 2 GeV electrons will deposit only part (in average about 25% in our case) of the the total energy. Then because of shower and light fluctuations some 2 GeV electrons will have less than 25% energy reconstructed in the calorimeters. Based on this, in the trigger we required the energy to be more than 300 MeV, which guarantee that more than 99% of 2 GeV electrons will deposit energy above the threshold.



(a) (b)

Figure 19: (a) Momentum distribution of "Good electrons". The Brown distribution represents all "Good electrons", The blue histograms represents all events where the electron trigger bit was not set, the black histogram represent events, which doesn't have EC × PCal trigger bit, and the red one represent events that missed the electron trigger bit. (b) distribution of the number of photoelectrons for where the electron has more than 2 GeV energy and missed the HTCC trigger bit.

In Fig.19a shown momentum distributions of all "Good" electrons (Brown), electrons when electron trigger bit was not set by trigger (blue), when EC×PCal bit was not set (black), and red represents events when HTCC bit was not set. As one can see above 2 GeV most of events have only HTCC bit missing. On the Fig.19b shown distribution of the number of photoelectrons for events, which miss the HTCC trigger bit. As

one can see about 90% of these events are at at the threshold (reminder that in the trigger we used 2 photoelecton threshold), and since in the trigger the precision of gains and pedestals is smaller (probably Ben can comment what was the exact precision) than the one used in offline reconstruction, this, then this could create potentially such threshold related effects.

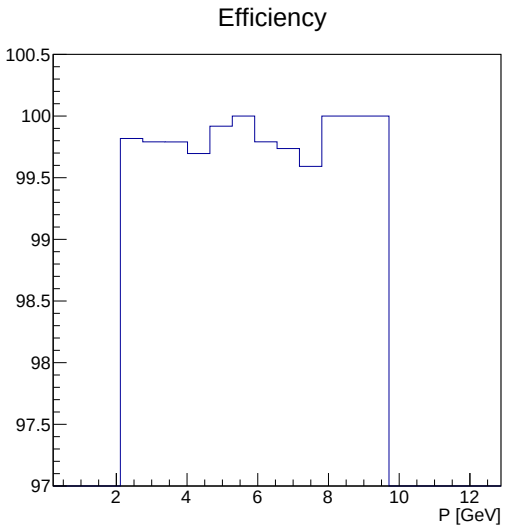


Figure 20: Electron trigger efficiency above 2 GeV

The final efficiency is shown in the Fig.20, where we can see above 2 GeV region, the trigger efficiency is above 95 %.

10.3. Validation of the Photoproduction Trigger

As described in Section 9.2, the CLAS12 photoproduction trigger requires a coincidence between one electron measured in the Forward Tagger detector, and two hadrons measured within the CLAS12 detector in either the forward or central part. The validation procedure aims to verify if, for a given event foreseeing one final-state electron in the FT acceptance and two or more hadrons scattered within the CLAS12 acceptance, the trigger system would recognize it properly, resulting in event readout. In order to validate the system with beam during commissioning, the following strategy was adopted. First, the e^- detection by the FT was validated using Random Trigger runs. After this, the detection of single hadrons in CLAS12 was studied in special runs, where the only trigger source was the FT. Finally, the coincidence between the two systems was assessed. Full details are described below.

10.3.1. Validation of e^- Detection in FT

A scattered electron in the Forward Tagger is identified as an electromagnetic shower in the Forward Tagger Calorimeter (FT-Cal) within a proper energy range, in time coincidence and geometrically matched to a hit in both layers of the Forward Tagger Hodoscope (FT-Hodo). The map providing the matching between the cluster seed position in the FT-Cal and the tile position in the FT-Hodo was first derived from the nominal detector geometry, and then confirmed by Monte Carlo simulations.

The identification of the scattered e^- in the FT was validated through a similar procedure as the one adopted for the CLAS12 electron trigger discussed in Section **??**, based on Random Trigger runs. The recorded events were processed through the standard CLAS12 reconstruction software and filtered, keeping only those with a reconstructed e^- in the FT system. Since event readout was triggered by a random pulser, events with the reconstructed e^- signal close to the margins of the readout window were also rejected. For these events, the electromagnetic clusters found by the reconstruction software ("offline" clusters) were compared to those reported by the trigger system and stored in the trigger data banks.

The efficiency of the FT-Cal clustering algorithm in the Trigger System was evaluated by comparing all "offline" clusters to those matched - in space and time - to the "online" clusters [1]. The efficiency was computed as:

$$\varepsilon = \frac{N_{trigger}}{N_{all}} , \qquad (3)$$

where N_{all} and $N_{trigger}$ are, respectively, the total number of "offline" clusters and the number of "offline" clusters matched to an "online" cluster. The result is shown in Fig. 21, reporting the FT trigger efficiency for electromagnetic clusters as a function of the corresponding corrected energy. The efficiency is higher than 97.5% over the full energy range of interest, and 99.5% in the energy range above 1 GeV. This small efficiency difference is mainly due to the fact that the clustering algorithm in the Trigger System works on a 3x3 matrix of crystals, whereas this limitation doesn't hold in the offline reconstruction. The efficiency of the FT-Cal /

[1] The energy of "offline" clusters is properly corrected to account for electromagnetic shower leakage from the back of the FT-Cal, while "online" clusters do not implement this. Therefore, for a given e^- in the FT-Cal, there is a systematic difference between the two energies. This effect is properly taken into account when setting the energy range for e^- detection in the Trigger System, and does not affect the corresponding trigger efficiency.

FT-Hodo matching algorithm was evaluated in a similar way, repeating the previous calculation but considering only electromagnetic clusters associated with one hit in each FT-Hodo layer. The result is reported in Fig. **??**.

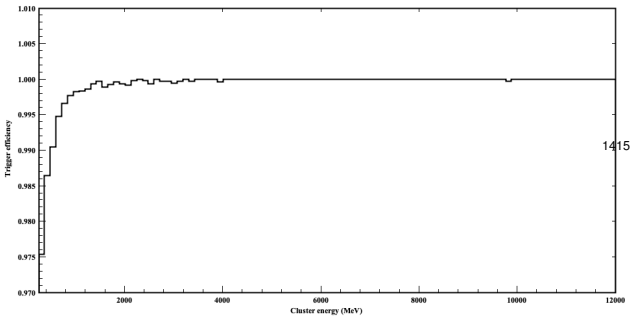


Figure 21: TODO: better figure, with errors (run 4288)

10.3.2. Validation of Charged Hadrons Detection in CLAS12-FD

The Trigger System recognizes a charged hadron in the CLAS12 Forward Detector as a hit in the Forward Time-Of-Flight System (**??**) (panel 1B) in time coincidence and geometrically matched to a hit in the U-strips of the Preshower Calorimeter **??** associated with a cluster with energy larger than a programmable threshold. The map providing the geometrical matching between the FTOF counter and the PCAL U-strip was first derived from the nominal detector geometry, and then confirmed by Monte Carlo simulations. To reduce the rate of random coincidences, the Trigger System also requires the presence of a segment in 5 out of 6 Drift Chamber superlayers in a given CLAS12 sector. The charged hadron identification algorithm was validated in special data-taking runs in which the Forward Tagger was the only enabled event readout source. In these runs, the Trigger System was configured to report in the output trigger bank the presence of a charged hadron in any CLAS12-FD sector, as defined in Section **??**.

The recorded events were processed through the standard reconstruction software and filtered, keeping only those with a well reconstructed charged track measured in the CLAS12-FD. The track was required to be within the nominal acceptance of the CLAS12 PCAL, and a momentum threshold of 300 MeV was applied. The Trigger System efficiency was evaluated by comparing all reconstructed tracks to the tracks recognized by the Trigger System. During commissioning, the efficiency was evaluated as a function of different observables, such as the energy deposited in the FTOF counters and in the PCAL, and the topology of the geometrical matching window. The trigger parameters were individually tuned to maximize the trigger efficiency. In the final configuration, an energy threshold of 2 MeV and 10 MeV for the FTOF counters and PCAL clusters, respectively, was selected. The result is reported in Fig. 22, showing the CLAS12-FD trigger efficiency for charged hadrons as a function of the track momentum. The efficiency is larger than 99% in the full momentum range, with the inefficiency dominated by threshold effects for the PCAL clusters.

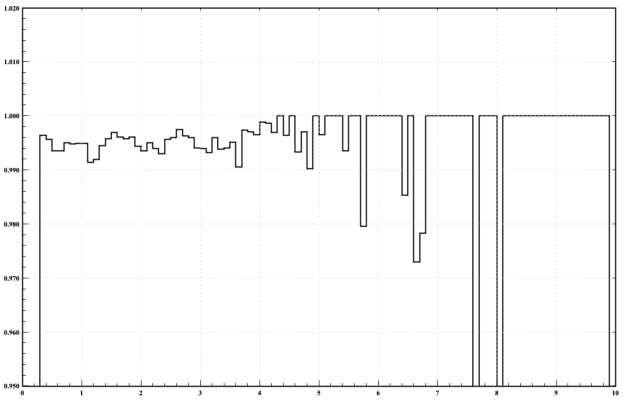


Figure 22: TODO: better figure, with errors - run 5049-5050 (run4)

10.4. Drift Chamber-Based Trigger Component and Data-Based Dictionary

The road dictionary for the Drift Chambers used within the Trigger System was initially generated using a fast Monte Carlo approach, where positive and negative particles in a selected momentum and angular range were randomly generated, tracked in the CLAS12 magnetic field using the CLAS "swimmer" developed for the offline reconstruction based on a 4th-order Runge-Kutta approach, and projected onto the DC wire planes to determine the hit position and therefore the DC wire IDs. This method has intrinsic limitation because of the approximation done in tracking the particle through the detector that does not include energy loss, multiple scattering or other effects due to the particle interaction with the detector material.

To overcome these limitations, roads were also generated from full GEANT4 simulations of the CLAS12 detector based on the GEMC package as described in Section 8.1. This provides an accurate description of the relevant materials particles travel through resulting in a more accurate road dictionary at the expenses of a significantly higher computing time to generated the same size dictionary.

The effectiveness of these two approaches was tested by using real tracks from beam data to evaluate the completeness of the dictionaries, i.e. the fraction of tracks for which a matching road is found. This study indicated that very large statistics is needed in the dictionary making to populate specific regions of the phase space.

As a third alternative approach, dictionaries were also produced from real tracks from beam data: in this case dictionaries with very large statistics can be produced in limited computing time with the advantage of the best accuracy in accounting both for particle interaction in matter and for the real detector geometry. These were the dictionaries that were used in the final trigger implementation.

10.5. Final Validation Before Experiment Start-up

After the CLAS12 detector was commissioned and the Trigger System components were validated, we still have to execute our validation processes for the entire system at the begining of every experiment. This is necessary because different experiments request configuration changes in the Trigger System, that take advantage of its flexibility. Also, we apply firmware changes occasionally to improve the Trigger System components based on our previous experience, and then changes have to be validated. The final Trigger System validation is performed by taking beam data with a random trigger (see Section 10.1).

The final trigger validation procedure was executed several times during the first year of CLAS12 experiments and was proven to be very useful: bugs in the trigger firmware were found and fixed, and the trigger configuration parameters were optimized. On one occasion a firmware bug was introduced into the PCAL Stage 1 trigger logic during the firmware update that was expected to be small and simple. The final validation procedure revealed an irregularity in the spacial distribution of clusters (see Fig. 23) (it also shows one CLAS12 sector is missing but this was another problem unrelated to the Trigger System). Since the PCAL Stage 1 trigger firmware is implemented in C++/HLS, the GEANT4 data sample was reprocessed through the C++ firmware implementation (see Fig. 24), and the problem was confirmed and subsequently found and fixed. The firmware was recompiled and reloaded, and the final trigger validation was repeated showing that the problem was fixed. It took only several hours between finding the problem and being ready to run again. Every experiment in CLAS12 begins with a complete Trigger System validation.

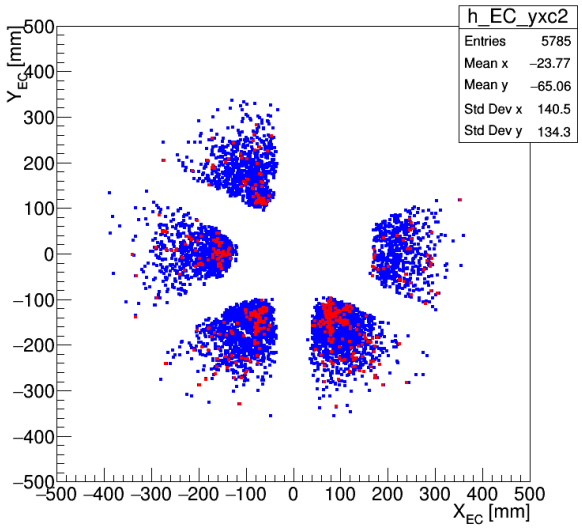


Figure 23: PCAL bug in beam data

11. Performance and Reliability

The CLAS12 Trigger System opperates as a free-running, pipeline-style system run from a global 250 MHz clock. It provides 32 global trigger decisions based on >10 different subdetectors (>28k channels), which allows for multiple experiments to operate simultaneously. The trigger efficiencies have in general been measured to be about 99%, indicating a reliable and efficient trigger implementation. The trigger purity has been measured to be about 55% for electrons (negatives inbending torus polarity configuration), taking advantage of energy-corrected clustering in the EC along with Drift Chamber track matching. It is possible to improve this purity by reducing the timing coincidence windows, jitter, and cell size of the Drift Chamber tracking dictionary. This can be checked by re-analyzing existing event data since it contains the raw waveform data. Not all physics triggers utilize the tracking trigger due to an incomplete road dictionary (e.g. neutral particle decays into charged particles with a detached vertex). Further work to expand the dictionary roads can be investigated as to further increase the selection purity.

12. Acknowledgements

We appreciate the support of administrative staff ...

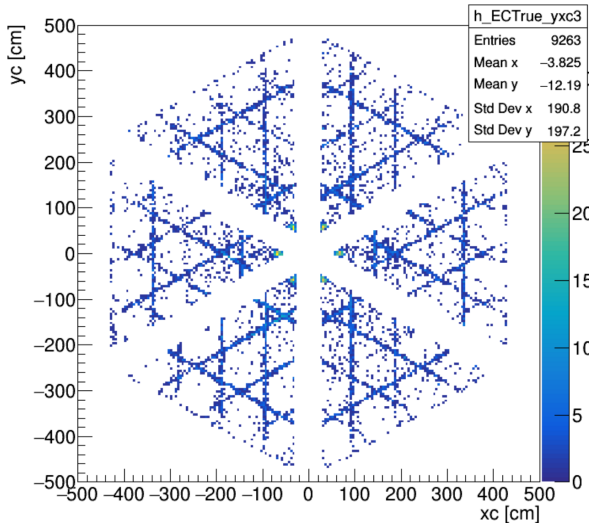


Figure 24: PCAL bug in GEANT4 simulation

13. Conclusion

14. References

References

[1] S.Boyarinov *et al.*, *"The CLAS12 Data Acquisition System"*, see this issue.

[2] V.D. Burkert *et al.*, *"The CLAS12 Spectrometer at Jefferson Laboratory"*, see this issue.

[3] B.A. Mecking *et al.*, Nucl. Inst. and Meth. **A503**, 513 (2003).

[4] J. Bettane *et al.*, *"The CLAS12 Central Neutron Detector"*, see this issue.

[5] D.S. Carman *et al.*, *"The Forward Time-of-Flight System for CLAS12"*, see this issue.

[6] D.S. Carman *et al.*, *"The Central Time-of-Flight System for CLAS12"*, see this issue.

[7] Y. Sharabian *et al.*, *"The CLAS12 High Threshold Cherenkov Counter"*, see this issue.

[8] G. Asryan *et al.*, *"The CLAS12 Forward Electromagnetic Calorimeter"*, see this issue.

[9] M. Mestayer *et al.*, *"The CLAS12 Drift Chamber System"*, see this issue.

[10] NNN *et al.*, *"The CLAS12 Forward Tagger System"*, see this issue.

[11] HLS *et al.*, *"VIVAO High Level Synthesis"*, yyy.