... to be complete ...

CLAS12 Trigger System

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

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

Abstract

Article describes CLAS12 Trigger system. Simulation, hardware and software design as well as validation procedured are discussed.

1. Overview

CLAS12 Tigger System provides trigger signals for CLAS12 detector Data Aquisition System[1]. Originally it was designed to search for electrons using CLAS12 two electromagnetic calorimeters and high threshold cherenkov counter, with possibility to add drift chamber tracker. On later stages of development additional detectors were included into the trigger system making it flexible and poerful machine to select events for various experiments included into CLAS12 physics program.

2. Requirements

CLAS12 Trigger System requirements are defined by CLAS12 physics program and shown in table

3. Design

CLAS12 Trigger System is designed as 3-stage pipeline-style system with total latency up to 8us. Stage 1 receives information from various CLAS12 detecrors and performs data processing in according to the type of detector. Stage 2 performs timing and geometry coincidence between different subset of dectors in 6 groups in according to 6-sector CLAS12 detector structure, as well as coincidence with information from central detectors. Stage 3 forms final trigger decision.

3.1. Stage 1

The most complex processing is performed for electromagnetic calorimeters (cluster finding) and drift chamber tracker (segment and road finding). Stage

3.1.1. Electromagnetic calorimeters
3.1.2. Drift Chamber
3.1.3. High Threshold Cherenkov Counter
3.1.4. Forward Time-Of-Flight Counter
3.1.5. Central Time-Of-Flight Counter
3.1.6. Neutron Detector
3.1.7. Forward Calorimeter and Hodoscope

3.2. Stage 2

3.3. Stage 3

4. Hardware Implementation

The CLAS12 Level 1 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 three sections starting with the front end VXS crate Trigger Processor (VTP), a global Sub-System Processor (SSP) and a Trigger Supervisor that manages the timing, synchronization and front end event readout. Within a front end crate, trigger information is gathered from each 16 Channel, 12bit Flash ADC module at 4nS intervals via the VXS backplane, to a Trigger Processor (VTP). Each Trigger Processor is capable of handling these 500MBps VXS links from the 16 FADC modules, and then performs real time crate level trigger algorithms. The crate trigger processor transmits the Level 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 10Gbps. The front end crate trigger data is transmitted on the VXS backplane, and on the multi-fiber link using the Aurora protocol from Xilinx. The front end modules use Virtex-V devices with Gigabit Transceivers operating at 2.5Gbps. The VXS

Trigger Processor 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 250Mhz 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 ¡3uS, and the proposed 12GeV experiments at Jefferson Lab require up to 200KHz Level 1 trigger rate.

4.1. VTP board

4.2. SSP board

5. High Level Synthesis in CLAS12 Trigger System development

5.1. Our motivation to use HLS

HLS makes it easier to incorporate well-established data processing algorithms, typically written in C++ or other high-level language, into FPGA-based projects. It allows to involve programmers who developed algorithms for offline data processing but with limited FPGA programming experience. It also makes it possible to validate code inside offline processing framework.

5.2. Trigger components implemented with HLS

HLS was used to develop most of stage 1 components of CLAS12 trigger:

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

Time-of-Flight and Cherenkov counters implementation was rather trivial, it would typically takes less then 10-15

Calorimeter trigger implementations required much more effort because of its complex nature which requires significant FPGA resources. Details are further explained in the next sections using Electromagnetic Calorimeters as the example.

5.3. CLAS12 Electromagnetic Calorimeters

Among all trigger system elements, the most challenging for FPGA implementation was trigger component serving two CLAS12 electromagnetic calorimeters. Due to its structure, those calorimeters do not provide cluster coordinates or energies without significant event reconstruction. Both calorimeters consists of three sets of scintillation strip layers with PMT readout on one side of strip. To reconstruct clusters it is needed to find peaks in all three layers, find crosses of those peaks, apply attenuation length corrections to individual scintillation strips, correct peaks energies, correct clusters energies, and finally report clusters coordinates and energies to the following stage of the trigger system. Cluster reconstruction procedure was developed before as part of offline analysis and was well established, so we decided to use it as starting point for trigger component design.

Event with single cluster is shown in CLAS12 PCAL (preshower calorimeter); corrected energies are shown for individual strips

5.4. HLS Settings

Clock uncertainty is set as 30% of main clock, we found that it forces HLS to produce more realistic timing estimates. A single HLS project often cannot exceed several percent of FF and LUT budget, otherwise it may be a problem to meet timing requirement on VIVDO implementation step. Typical project from Calorimeter trigger is shown below.

5.5. VIVADO Settings

Common settings were used as shown on picture below. It usually takes 3+ hours to compile Calorimeter project on Dell R730 server under RHEL7. For some firmware versions, we were able to utilize 100% of LUTs and still meet timing – if clock domain was 125MHz or lower.

5.6. Firmware validation for HLS-based projects

The ability to validate firmware using 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 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.

5.7. Our conclusion about HLS Usage

CLAS12 Calorimeters and other detectors were successfully implemented into trigger system using HLS to produce core part of the firmware. This trigger was used in the first physics run and worked as expected. We were able to select events based on individual 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 Cherenkov or time-of-flight counters. For components utilizing significant portion of the FPGA, it will be great to improve HLS in following directions:

Support multi-clock domains Improve subroutine calls by allowing option to fully registered 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. It will allows to avoid splitting project manually and use HDL as top interface as we are currently forced to do.

6. Software

6.1. Development Software

6.2. Operational Software

Stage 1 and stage 3 of the CLAS12 Trigger System controlled through Arch Linux running in Xilinx arm ...

Stage 2 controlled by CentOS Linux running on Intel controller. As results all 3 stages designed to provide convenient access to ... using c/c++ programs.

6.3. Configuration Software

..config files..

..time coinsidence tool..

.. gain calibration and threshold settings ..

7. Simulation

Several simulation tools were used in CLAS12 Trigger System Project. For electromagnetic calorimeters components development input data were generated using GEANT.

8. Validation

CLAS12 Trigger System Validation was initially performed on data samples simulated by GEANT when system was in design stage. When system was installed cosmic data were used. Finally when beam operation started, validation was compelte and part of entire CLAS12 detector commissioning. Details discussed below.

8.1. Validation during design phase

Validation process consists of several methods and depends on the nature of validated trigger component. For stage1 components written on C++ for HLS/VIVADO implementation, GEANT-simulated data were processed directly by c++ code and compared with initial similation parameters. In addition, the same samples were processed by offline reconstruction software and results compared with the trigger output. That double check method practicaly guarantees bug-free implementation. It was no single case when c++ implementation passed validation on simulated data and failed on final validation stage. Most complicated stage1 components were validated using this method.

Some stage1, as well as all stage2 and stage3 components were implemented using VHDL. Fof those, VIVADO tools were used for validation diring design phase, in particular ...

9. Validation of the installed system using cosmic runs

10. Validation on beam during CLAS12 detector commissioning

.. random trigger runs ..

.. gain check ..

11. Validation for different experiments

After CLAS12 detector was commissioned and trigger system was validated, we still have to repeat validation process occasionally. It is needed because different experiments requesting configuration changes in trigger system, taking advantage of it's flexibility.

12. Performance and Reliability

As free-running pipeline-style system, CLAS12 Trigger System allows to run with event rate up to ... Actial trigger rate in CLAS12 depends on experiment and can be 20kHz and higher ..

... problems ...

13. Acknowledgements

We appreciate the support of administrative staff ...

14. Conclusion

15. References

[1] A. Einstein, Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies], Annalen der Physik 322 (10) (1905) 891–921. doi:http://dx.doi.org/10.1002/andp.19053221004.