Trigger System for a Large Area RPC TOF-tracker

Filomena M. C. Clemêncio, Alberto Blanco, Nuno Carolino, Custódio F. M. Loureiro

***Abstract*–The TOF-tracker concept, the simultaneous measurement of accurate time and bi-dimensional space coordinates in a single gaseous detector, has been previously demonstrated. Recently, a larger area, 1550x1250 mm^2, RPC detector has been constructed. Signals are induced in metallic strips located in each side of the RPC active volume and coupled to both charge-sensitive and timing circuits, that can be used to generate a coincidence trigger.**

In this work, the architecture and performance of the trigger system of the detector is reported. The system is based on off-the-shelf inexpensive modules and in a custom program written in VHDL running in a Xilinx Spartan-6 FPGA. It is fast, fully parameterized and supports several trigger strategies, allowing at the same time the collection of several information referring to the operating condition of the detector, relevant for tests and detector maintenance.

I. Introduction

THE TOF-tracker concept, the simultaneous measurement of accurate time and bi-dimensional space coordinates in a single gaseous detector, has been previously demonstrated [1]. The detector yielded a time accuracy of 77 ps sigma along with a bi-dimensional position accuracy of 38 um sigma over a full active area of 60 x 60 mm^2.

Recently, a large area, 1550x1250 mm2, RPC detector has been constructed for the same purpose. Preliminary results suggest a time resolution of 150 ps sigma together with a 2D position resolution of 1.3 mm and 3.4 mm sigma over the entire area of the detector, without cuts, a fine alignment procedure or systematic error correction [2]. Signals are induced in metallic strips located in each side of the RPC active volume and coupled to both charge-sensitive and timing circuits. The outputs of the timing circuits are used to generate a trigger signal if certain conditions are met, which initiates the acquisition of data from the charge-sensitive channels and the recording of the time stamps of the events, acquired with an external TDC.

In this work, the architecture and performance of the trigger system of the detector is reported. The system is based on off-the-shelf available modules. A Raspberry Pi2 running Linux is used as a control and logging computer. The trigger algorithms are described in VHDL and implemented using a Xilinx XC6LX45-2 FPGA housed in module TE0600 from Trenz Electronic GmbH [3]. Communication between the FPGA and Raspberry Pi2 is made using the SPI protocol.

The Raspberry software was written in C++ using the GNU GCC compiler [4] and enables programming the different trigger modes, testing the communication with the underlying FPGA, and the periodic log of data**.**

Manuscript received May 29, 2016.

F. M. C. Clemêncio is with LIBPhys, Department of Physics, University of Coimbra, P-3004-516 Coimbra, Portugal, and Escola Superior de Tecnologia da Saúde do Porto - IPP, Rua Valente Perfeito, 322, 4400-330 Vila Nova de Gaia, Portugal (e-mail: fcc@estsp.ipp.pt).

A. Blanco and Nuno Carolino are with LIP, Laboratório de Instrumentação e Física Experimental de Partículas, 3004-516 Coimbra, Portugal (email: alberto@coimbra.lip.pt).

C. F. M. Loureiro is with LIBPhys, Department of Physics, University of Coimbra, P-3004-516 Coimbra, Portugal (e-mail: custodio@uc.pt).

II. Trigger System requirements

The trigger system was initially specified to handle 16 signals (time channels) arriving from the timing circuits. The FPGA decides if coincidence conditions are met according to the programmed parameters and generates the trigger signal to the data acquisition system. The width (duration in time) of the trigger signal is to be programmable by the user.

Several different trigger modes of operation were specified. The first one, nicknamed And_trigger, says that the trigger signal is to be generated if and only if there is temporal coincidence between a given set of time channels (the programmed And group). The second one was coined Or_trigger, and specifies that the trigger signal should be generated if there is an event in any of a given group of time channels (the programmed Or group). For generality a mix of the two trigger modes was also implemented, named And_or_trigger, meaning that the trigger signal is generated if either the And_trigger condition or the Or_trigger condition is met. It was also decided that any time channel can be selected to be part of the And group or Or group of signals.

The system should also be able to monitor constantly the output of the timing circuits, a valuable information of the status of the detector.

As a last requirement the system should run under the control of the Raspberry Pi2 that is playing the role of host and controller computer. So it was decided that communication with the FPGA would be established using the Raspberry SPI interface.

The system should also be able to periodically collect information about the operating conditions of the detector.

To keep development costs low and reduce the time spent developing the trigger system it was decided that a commercially available FPGA prototyping board should be used. From the several available the module TE0600 from Trenz Electronic GmbH was selected, due to high availability, expected end of life in 2027 [5] and low cost. The module contains a Xilinx XC6LX45-2FGG484I FPGA and all electronics needed for its operation in a small form factor, the size of a credit card.

978-1-5090-2014-0/16/$31.00 ©2016 IEEE

III. System Architecture

The trigger system is composed of mainly three parts: a front-end module, a SPI interface, and the trigger processing unit.

A. Front-end module

Signals arriving from the timing circuits are not compatible with the FPGA inputs, so there is the need of converting them to compatible digital levels.

A custom board was designed with this aim: it implements a fast comparator followed by a buffer gate in each channel. It compares each signal arriving from the timing circuits to a programmable threshold, and converts the output of the comparators to LVTTL levels to be sent to the FPGA. These are the signal that contains the information to be processed.

In a similar way the trigger created by the FPGA is converted to LVDS through a high-speed differential line driver.

B. SPI interface

A parameterized SPI slave interface was developed in VHDL for communication with the host Raspberry Pi2. Practice showed that, due to the harsh electrical conditions, communication synchronization could be lost, and so a reset signal was added to the SPI interface to allow automatic recovery if an error is detected.

C. Trigger processing unit

The trigger processing unit is based on a Xilinx Spartan-6 FPGA and code is written in VHDL. It processes continuously all events arriving from the front-end module and generates the corresponding trigger signal, according to the parameters programmed.

It is essentially composed of: a coincidence detection block, an internally generated synchronization trigger signal, and a stretcher, along with several counters.

The coincidence detection block receives information of the And and Or group of channels, and of the trigger mode of operation desired. Its output is an internal trigger signal, if coincidence conditions are met.

To this internal signal can be added a periodic signal for synchronization of external devices

The resultant trigger signal is then passed through the programmable stretcher and sent to the data acquisition system.

To this internal signal can be added a periodic signal for synchronization of external devices

The resultant trigger signal is then passed through the programmable stretcher and sent to the data acquisition system.

D. Monitoring processing unit

In order to gather valuable information of the status of the detector several 32-bit counters are connected in the input of each FPGA channel. These counters can be enabled under the control of the Raspberry Pi2 and information collected during a certain amount of time. Events on all time channels are counted. All internally generated trigger signals (before and after the stretcher) are also counted. This information allows the user to have a good grasp of the operating condition of the detector.

IV. Control Software

The control of the trigger system is made using three independent programs (ctsrun, ctschk, ctslog) run by the Raspberry Pi2.

An object-oriented approach was taken while designing the software. The need of using the SPI interface and of accessing the numerous FPGA ports was hidden inside the several classes developed. The approach favored incremental design and test and an easily understandable and modifiable top software module.

A clean separation was made between programming the trigger mode of operation (ctsrun), checking that the trigger system is programmed as expected (ctschk), and acquiring and logging data (ctslog).

The ctsrun program receives as parameters the number of the channels pertaining to the And group and to the Or group and the trigger mode of operation desired (And_trigger, Or_trigger, And_or_trigger) and programs the corresponding ports of the FPGA.

The ctschk program verifies that the FPGA was correctly programmed by reading its ports and comparing their values to the required ones.

The ctslog program receives as a parameter the time channels to be displayed and the data acquisition time to be used.

V. Experimental Results and Conclusion

The trigger system behaves as expected. The file created by the ctslog program contain pertinent information relating to the operating status of the detector, allowing to verify and optimize its operating characteristics. The trigger signal is generated with low latency and in accordance to the programmed mode of operation and with the correct time width.

References

[1] A. Blanco, P. Fonte, L. Lopes, P. Martins, J. Michel, M. Palka, M. Kajetanowicz, G. Korcyl, M. Traxler and R. Marques, "TOFtracker: gaseous detector with bidimensional tracking and time-of-flight capabilities", 2012 JINST 7 P11012.

[2] https://indico.ugent.be/event/0/session/13/contribution/57

[3] http://www.trenz-electronic.de/products/fpga-boards/trenz-electronic/te0600.html

[4] https://gcc.gnu.org/

[5] http://www.trenz-electronic.de/products/fpga-boards/trenz-electronic.html