www.czasopisma.pan.pl PAN www.journals.pan.pl

jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2021, VOL. 67, NO. 1, PP. 103–108
Manuscript received December 29, 2020; revised January, 2021. DOI: 10.24425/ijet.2021.135950

Zero-Suppression Trigger Mode for GEM Detector Measurement System

Piotr Kolasinski, Krzysztof Pozniak, Andrzej Wojenski, Pawel Linczuk, Rafal Krawczyk, Michal Gaska, Wojciech Zabolotny, Grzegorz Kasprowicz, Maryna Chernyshova and Tomasz Czarski

***Abstract*—A novel approach to a trigger mode in the Gas Electron Multiplier (GEM) detector readout system is presented. The system is already installed at WEST tokamak. The article briefly describes the architecture of the GEM detector and the measurement system. Currently the system can work in two trigger modes: Global Trigger and Local Trigger. All trigger processing blocks are parts of the Charge Signal Sequencer module which is responsible for transferring data to the PC. Therefore, the article presents structure of the Sequencer with details about basic blocks, theirs functionality and output data configuration. The Sequencer with the trigger algorithms is implemented in an FPGA chip from Xilinx. Global Trigger, which is a default mode for the system, is not efficient and has limitations due to storing much data without any information. Local trigger which is under tests, removes data redundancy and is constructed to send only valid data, but the rest of the software, especially on the PC side, is still under development. Therefore authors propose the trigger mode which combines functionality of two existing modes. The proposed trigger, called Zero Suppression Trigger, is compatible with the existing interfaces of the PC software, but is also capable to verify and filter incoming signals and transfer only recognized events. The results of the implementation and simulation are presented.**

***Keywords*—FPGA, GEM, trigger, sequencer, DAQ, Xilinx**

I. Introduction

THIS paper refers to the currently being developed measurement system for plasma diagnostics of ITER-oriented tokamaks (WEST). In such tokamaks, measuring soft X-ray (SXR) radiation of magnetic fusion plasmas is a standard way of collecting valuable information on particle transport and MHD (MagnetoHydroDynamics) phenomena for plasma optimization. The radiation emissivity can be up to 20 keV [1], [2]. Observed photon fluxes in the soft X-ray energy region usually reach values from 10^7/s·cm^2 to 10^9/s·cm^2. For such intensive photon streams, the designed measurement system, beside requiring adequate and specific detectors characteristics, needs also radiation-tolerant, high speed readout electronic and data acquisition system [3].

As a sensor unit, the Gas Electron Multiplier technology is used [4]. GEM detector in the developed system is built from: window, 3 GEM foils for proper electrons' multiplication and readout board. The design of the readout board is modular and universal, which means that both 1D or 2D boards can be installed. They vary only by the data processing algorithms, implemented in the FPGA chips and PC. Thanks to mentioned versatility of construction measuring energy and position of photons absorbed in the detector conversion layer at the same time is possible. Working with 2D readout boards requires handling larger number of readout channels with very short analogue pulses containing charge information. In another words, comparing to the 1D readout board, in 2D board much more data has to be acquired and processed.

Data processing in the system is done in two main hardware blocks: in FPGA and PC units. One of the requirements for the system is possibility of continuous work in a real-time mode. Therefore all parts of the system are connected by high speed interfaces and are required to process data as fast as possible. The FPGA chips have many advantages in such multi-channel systems. They can handle many independent data channels and process all of them simultaneously. Parallel multi-channel processing can save space and decrease the dimensions of the measurement system. The next advantage of the FPGA is possibility to design hardware with known low latency which plays a significant role in the fully real-time systems. On the other hand, FPGAs, though are great tools for multi-channel and big data systems, have also some limitations, like finite availability of resources for implementation more advanced algorithms. Also development process of hardware for the FPGA in hardware languages like VHDL or Verilog is more time-consuming than in other high level software languages.

During development first generation of the system dedicated for the JET experiment, due to limited bandwidth, authors placed more complex processing algorithms in the FPGAs [5]. This approach caused that the system could deliver to scientists only already processed data. Working on the raw measurement data was impossible, thereby the number of algorithms which could be used in the experiment with data was significantly limited. The GEM system designed for the WEST tokamak was required to work in the real-time mode and had to deliver

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. This scientific work was partly supported by the Polish Ministry of Science and Higher Education within the framework of the scientific financial resources in the year 2020 allocated for the realization of the international co-financed project No 5125/H2020-Euratom/2020/2.

P. Kolasinski, K. Pozniak, A. Wojenski, P. Linczuk, R. Krawczyk, M. Gaska, W. Zabolotny and G. Kasprowicz are with Institute of Electronic Systems, Faculty of Electronics and Information Technology, University of Technology, Warsaw, Poland (e-mail: p.kolasinski@elka.pw.edu.pl).

R. Krawczyk is with CERN, Geneva,Switzerland.

M. Chernyshova and T. Czarski are with Institute of Plasma Physics and Laser Microfusion, Warsaw, Poland.

raw data. Therefore in the detector developed for WEST, classified as a second generation detector, the authors, based on the JET experience, decided to move all complex algorithms from the FPGA to the real-time PC software and focused on the communication interfaces between various parts of the system to increase their bandwidths. In FPGA are implemented only basic pre-processing algorithms like trigger system or event identification. The raw data is sent to the PC software by high speed interfaces. That new approach allows engineers to work on raw data in high level scientific environments like MatLab.

Because for the WEST experiment, the high bandwidth at every stage and low latency is crucial for the system, authors in the article propose a new concept of the trigger algorithm for transferring data between FPGA and PC. The interfaces in the FPGA were redesigned and optimised. Additionally new concept is still compatible with already existing readout software on the PC side. Reworking of the transferring algorithm between FPGA and PC, allows to process much more information in the same time [6].

II. System Hardware Overview

The readout system is fully modular [7]–[9]. The GEM detector is connected to the Analog Front-End boards (AFE). Each board handles processing and transmission of 16 analog input channels. All boards contain transconductance amplifiers and generators of calibration pulses for verification and test purposes. Because of the environment in which the readout system works, all parts used in the AFE hardware are radiation-tolerant. ADC sampling rate is 80 MHz at effective number of 10 bits (ENOB). The signals from all channels of the AFE boards are transferred to the next stage of the system located in the rack. There are placed Analog-to-Digital Converter boards (ADBs) connected to the backplane FPGA boards with Xilinx Artix7 FPGA chip placed on it. Every FPGA board supports up to 64 input channels (four sets of AFE and ADB boards). In the FPGA data is pre-processed, synchronized and sorted by time and position. The backplane boards are connected through the PCI-E switch to the embedded PC. Onboard DDR3 memory is used for buffering purposes. Data after arriving to the PC can be processed by complex algorithms and verified. The architecture of the system is depicted in Fig. 1.

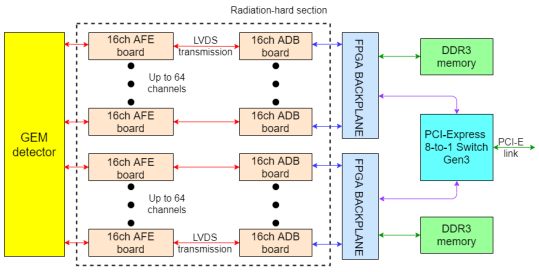

Fig. 1. Hardware System Architecture (based on [10])

III. Charge Signal Sequencer

Due to requirement for a real-time processing, received data by the PC has to be ordered and prepared for the next calculations. Therefore the Charge Signal Sequencer [10] implemented in an FPGA chip, responsible for sorting and delivering data to the PC, is one of the main modules in the system. This is the last stage of the processing data in the FPGA chip. It has big impact on the latency of the system and data consistency. The basic architecture of the Charge Signal Sequencer is depicted in Fig. 2.

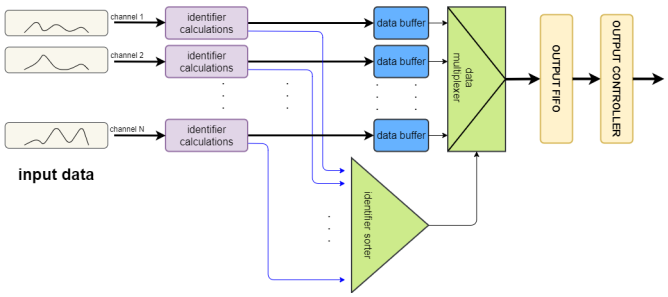

Fig. 2. Basic dataflow structure of the sequencer algorithm (based on [10]).

The Sequencer handles the data from all channels simultaneously, sorts it in chronological, unequivocal order and sends to the output buffer [11], [12]. To start processing the signals which create event, sequencer waits for the external trigger which comes with the data to every channel independently. How the incoming triggers will be processed it depends on the current trigger mode configuration.

Every event is wrapped into a frame, which is built from four fields:

- status bits for storing useful information
- channel number
- time of the sample
- raw data

First three fields create identifier of the event and frame.

Raw data, to save resources, is stored in the buffers independent for every channel. In the same time, identifiers of every frame, go to the sorter, where are sorted by time and, in case of equal time, by position.

Identifier Sorter (Fig. 3) is built from advanced comparator modules called Basic Sorting Block (BSB). It can be seen that Identifier Sorter has a binary tree structure, where every next BSB block is connected with twos from the previous stage. The following steps describe the BSB algorithm flow:

1) check if the output register in the BSB stores the valid data
2) if the output register is full, the BSB does not execute any more steps in the ongoing cycle. If the output register is empty, the controller checks if any data is available at the inputs ports.
3) if there is no data available at inputs, the BSB does not execute any more steps in the ongoing cycle. In other way, the controller checks if the data is available only on one port or both

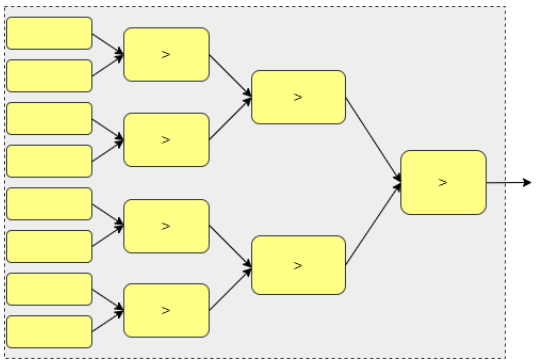

Fig. 3. Identifier Sorter (based on [13]).

4) if the data comes only at one port, then that data is taken over. If on both input ports data is valid, controller process data with a smaller timestamp
5) if both data have the same timestamp, priority has data from lower channel number

At the output of the last BSB block in the sorter, identifiers are sorted by time and by position. Based on the output from the sorter, the controller drives the multiplexer, which is responsible for transferring data from the correct channel FIFO, then concatenate it with the identifier and save it in the output FIFO memory. More about sorter concept can be found in [14].

Output buffer is 256 bits width, so data from the sequencer has to be adjusted to that width. Therefore full frame for the one event contains 768 bits (Fig. 4) and is built from the following positions:

- header: 32 bits
- status bits: 24 bits
- timestamp: 64 bits
- position: 8 bits (on a single board up to 64 channels, but extended to 8, just for adjusting purposes)
- data: 640 bit (40 samples and every is 16-bits length)

Header informs PC about first bits of the frame and simplifies the algorithms on PC side.

Because the frame at this stage contains 768 bits, to send whole frame to the output buffer, controller needs three cycles.

IV. Trigger Modes

In the system are currently implemented two trigger modes, which are placed at the input to the sequencer. The Global Trigger Mode and the Local Trigger Mode. The trigger is set during configuration procedure and can be changed after every measurement process. The Global Trigger mode is compatible with existing interfaces and software in the PC.

Fig. 4. Output frame structure.

The Local Trigger Mode is better optimised, by it is not yet compatible with existing PC software. It means that its operational possibilities are currently limited.

A. Global Trigger Mode

Basic concept of the Global Trigger mode is presented in Fig. 5.

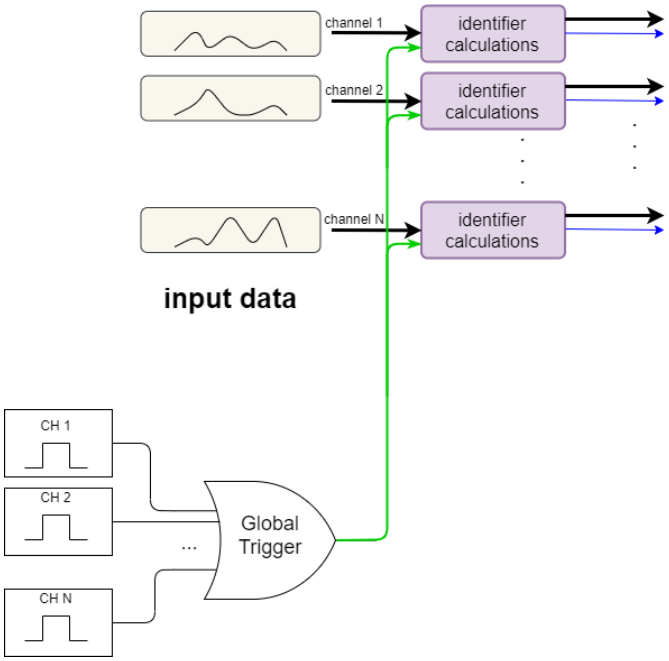

Fig. 5. Global Trigger Architecture.

Triggers of all channels are connected to the OR logic gate. Valid trigger signal on any channel triggers the Sequencer for all of them. The Sequencer receives valid data from all channels in the same time (Fig. 6). Measurement data is processed from all channels, not only from channels where the valid trigger signal appeared. Global Trigger allows to verify data, channels, dependencies between channels. Also this mode is compatible with fast processing software on the PC side. In a high rate experiments usability of such trigger mode is limited, because it process and sends to the PC not only valid data events, but data from all channels, even there were no registered events.

B. Local Trigger Mode

Concept of the Local Trigger Mode is depicted in Fig. 7. In that mode all channels' triggers work independently and simultaneously. Every channel waits only for its individual trigger. The incoming trigger on a specific channel starts processing only its own data and forwards it to the Sequencer (Fig. 8). The risk of overloading Sequencer in a such mode is much lower than in Global Trigger, where in the same time always appear at every channel data to process. Local Trigger is optimised for transferring only valid events data, so it is applicable for high rate experiments where quality of data and its content is crucial. The Local Trigger Mode will block sending any data without valid event.

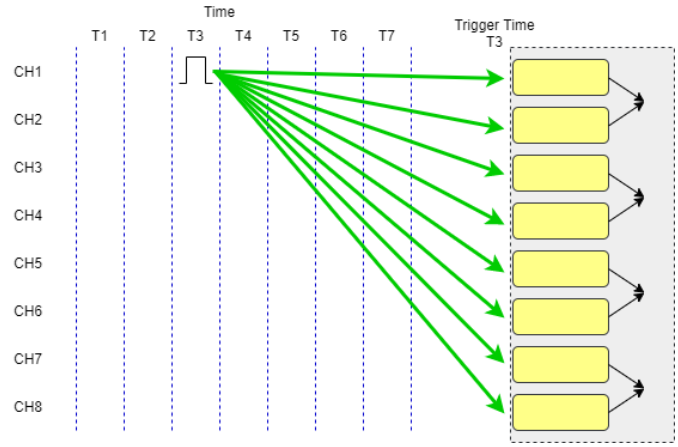

Fig. 6. Global Trigger Signal Propagation.

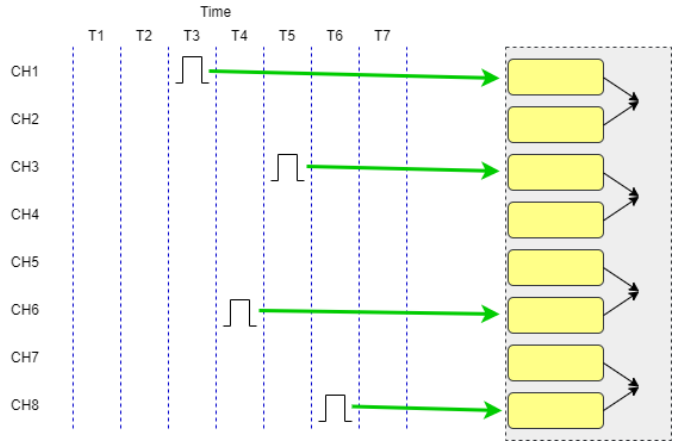

Fig. 8. Local Trigger Signal Propagation.

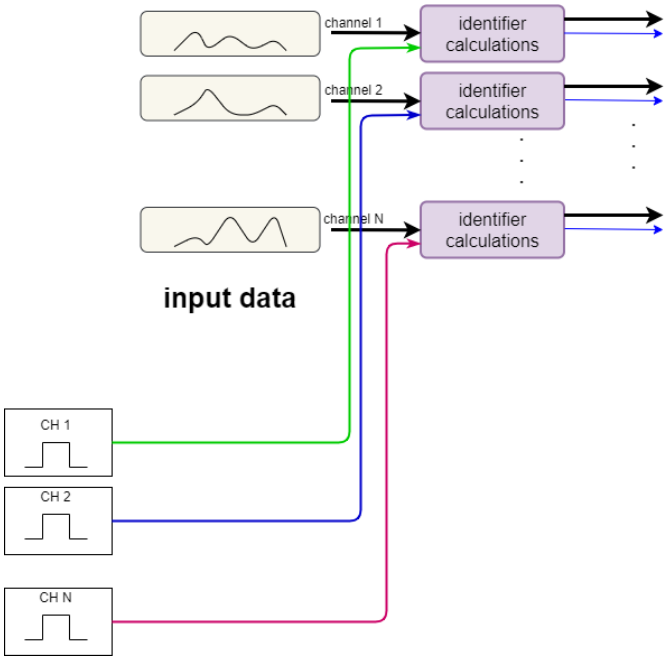

Fig. 7. Local Trigger Architecture.

C. Zero Suppression Triger Mode

Zero Suppression Trigger Mode is depicted in Fig. 9. This is advanced trigger which combines functionality of the Global and Local Trigger Mode. In comparison to existing trigger modes, this one adds a new module – Trigger Control Unit. Every channel has defined pre-trigger FIFOs where incoming data can be stored. It should be able to store at least 2 full events. Therefore for WEST experiment, where event contains 40 samples, the deep of the pre-trigger FIFOs is set to 128 words. Because writting and reading to all FIFOs is being done in the same cycles, all FIFOs are controlled by only one Finite State Machine for writing and one for reading. This approach allows to simplify control complexity and saves resources.

When trigger appears on any channel, data from all channels is written to the pre-trigger FIFOs (Fig. 10). During the event, the trigger controller saves information on which additional channels appeared triggers. All that information is stored in the registers called Release Register. When the event is finished and all 40 samples were written to the pre-trigger FIFOs, the trigger controller starts procedure of releasing registered data to the Sequencer. Sequencer, in the same time, receives only data from channels where trigger occurred during the event (Fig. 11). The timestamp is equal to the time when the first trigger appeared. Output data in the Zero Suppression Trigger Mode is compatible with the PC software which is responsible for collecting the data, but it also sends only the valid data (which would not be possible with Globatl Trigger Mode).

The performance of the proposed new trigger mode, does not depend on the performace and bandwidth of other blocks in the Sequencer. The Trigger Control Unit stores every new incoming data, but bewteen next events has to be at least break for one clock cycle. As long as all channels are processed simultaneously, it does not matter, if the trigger appears on one channel or all of them in the same cycle. When the event writing process to the pre-trigger FIFOs is finished, two cycles later starts process of releasing the data. If the Identifier Sorter (which is a next processing block) is ready to take over the released data, it starts processing it, if not, the event data is lost.

Zero Suppression mode combines advantages of global and local trigger modes. It is compatible with the rest of the system, but also sends only data with valid events.

V. Simulation

The block was simulated in a Modelsim to verify correctness of implementation (Fig. 12). The simulation process was divided into two stages. During first stage, only new Trigger Control Unit was tested with unity tests. Test cases included various inpusts data and trigger vectors. Examples of trigger vectors used during unity tests:

- set to HIGH triggers of all channels in the same clock cycle
- set to HIGH triggers of random channels in the same clock cycle
- set to HIGH triggers of channels accrodingly to patterns in various clock cycles

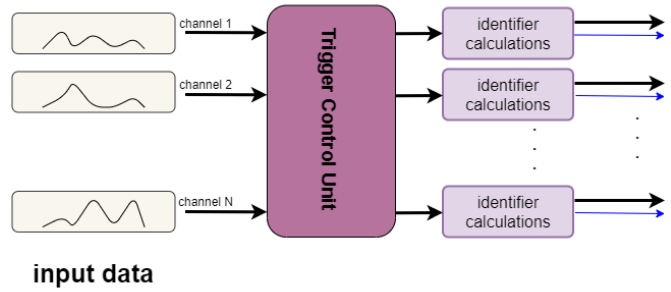

Fig. 9. Zero Suppression Trigger Architecture.

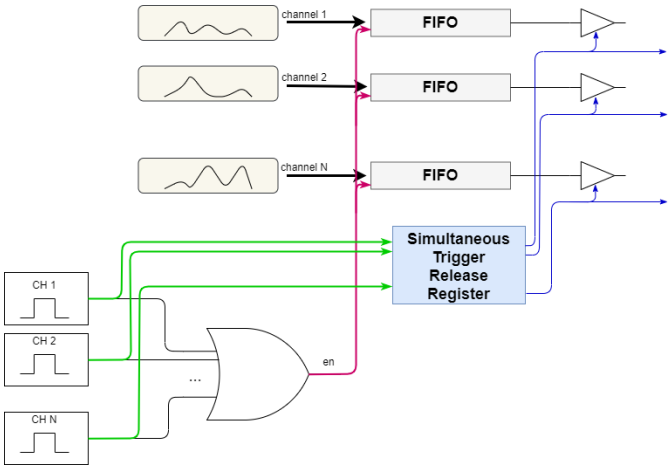

Fig. 10. Detailed Zero Suppression Trigger Architecture.

Additionally were tested cases to verify maximum events/channel frequency.

Second stage of the simulation tests new trigger mode whith whole existing structure of the Sequencer. In such tests also various trigger vectors, listed before, were processed

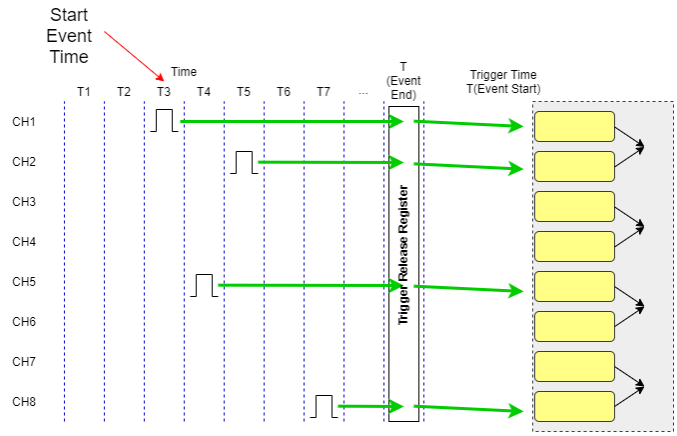

Fig. 11. Zero Suppression Trigger Signal Propagation.

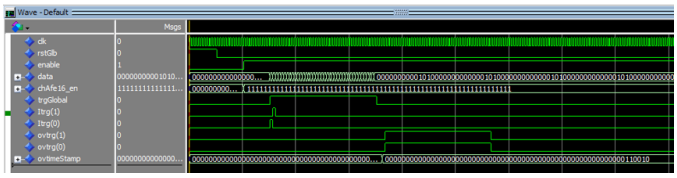

Fig. 12. Results of the simulation of the Zero Suppression Trigger Mode.

and checked (triggers, events frequency). Additionally were checked cases with full and blocked Identifier Sorter.

The Sequencer and the new Trigger Mode are quite complex modules. In case of triggering all channels in the same time, there would be to check 2560 (64x40) samples and order of 64 channels. Due to so much data it would be difficult to create test patterns and then verify results manually. Therefore all tests were performed fully automatically in the VHDL environment.

To generate data patterns, function was used. During the tests, the data values do not matter. Therefore they were used as a reference to check results of the tests. For every event, value of a sample is increased in every cycle. It means that if the first sample in the event has value X, the second will have X+1, the third X+2, and the last X+39. Every new event will start with the last used value incremented by 1. E.g. if in the same cycle are generated more events for various channels, the first event will have samples with values in range X - X+39, the second will have values in range X+39+1 - X+39+39 etc. For every new event, are generated different values of samples. That approach simplifies verification of the output data and ensures that no event is repeated. The testbench starts with 1, as a first value in the first event.

Beside data generation for samples, it is also important to verify if the information about source (channel) is stored and transfered to the output properly. During generating triggeres for channels, this information is stored in the FIFO. The FIFO keeps order of triggerred channels.

To verify if the output data and channel order is correct, the another function was developed. After receiving the event from the Sequencer the functions starts processing it. It checks correctness in two ways:

- check if the first sample in the event is just greater by one from the last chacked value. Additionally check if all next samples in the event are greater by one from the previous one.
- read from FIFO the next expected source of the event and compare it with received value. If they match, it means that the source is correct.

As long as all input vectors are generated with described algorithm, the testbench can contain many various tests. Output functions will always find any disfunction and incorrect behaviour of the Sequencer. In such verification, the is no need to check anything manually.

VI. Results of Implementation

Zero Suppression Trigger Mode was implemented in xc7a200tffg1156-2 chip with VHDL language. Block works

with frequency of 80 MHz. Xilinx IP cores were used for generating memory blocks. Details about used resources are presented in Fig. 13. Every pre-trigger FIFO uses half of the Block RAM. Whole New Zero-Suppression Trigger module uses 1900 Slices (almost 3000 LUT as logic and almost 4600 Registers). After adding new Trigger used resources were increased by 10%. It is worth to notice, that exisiting triggers, due to their simplicity, have negligible impact on the resources. Developed architecture is optimised for WEST experiment.

Name	Slice (33650)	Slice LUTs (134600)	Slice Registers (269200)	Block RAM Tile (365)	LUT as Logic (134600)
muxt.muxl[63].mu...	32	57	55	2	57
zsInCtrl_i (sorter_z...	1900	2282	4570	32	2282
sorter_out_ctrl_inst (s...	748	1814	1245	0	1814

Fig. 13. Results of the implementation of the Zero Suppression Trigger Mode.

VII. Summary

The article briefly presents GEM detector system. It describes GEM detector, hardware readout system and module responsible for transferring data between FPGA and PC. Currently the system can work with two trigger modes, but both of them have disadvantages and limitations. Therefore the authors in the article presented the third trigger mode which combines functionality of two others. It is an important part in the actual development process, because it is still compatible with existing interfaces and software on the PC side, but also is designed to transfer only valid event. With the same bandwith, much more information can be sent. Comparing to the other triggers, Zero-Suppression Trigger is much more complex and uses more resources. Global and Local triggers designs are based on a glue logic, while the Zero-Suppression Trigger uses memories, control units, registers, Finite State Machines.

Currently the first version of the Zero-Suppression Trigger Mode is under tests. However there are already planned features improvements for next releases. Currently between events has to be at least clock cycle. In next releases this limit will be removed. Additionally, currently, if the Identifier Sorter is not able to process data from the Trigger block, the data is lost. Is some experiments it would be valuable, to not lose that data, and wait with the releasing it, untill the Identifier Sorter is ready to take it over. In such approach, the FIFOs in the Trigger module could work as additionall buffers (very useful in experiments with high rates). That behaviour of the module will be configurable.

In some cases also will be valueable if there is possibility to disable some channels in the Trigger module e.g during calibration or service processes.

The new trigger mode was designed, implemented and verified in the simulation. It is already merged with other sources and is added to the hw firmware in FPGAs. The next step will be a verification in hardware. It must be mentioned that this is only temporary solution used in actual state of system development. When the PC software will be updated to work with the Local Trigger Mode, which is still more efficient, it will become a default trigger in the measurement system.

References

[1] D. Mazon et al., "Design of soft-X-ray tomographic system in WEST using GEM detectors", *Fusion Engineering and Design* vol. 96-97, pp. 856-860 (2015).

[2] M. Chernyshova et al., "Design and development of soft x-ray diagnostics based on GEM detectors at IPPLM", *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2019* 111764F (2019).

[3] A. J. Wojenski et al., "Multichannel reconfigurable measurement system for hot plasma diagnostics based on GEM-2D detector", *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* vol. 364, pp. 49-53 (2015).

[4] D. Mazon et al., "GEM detectors for WEST and potential application for heavy impurity transport studies", *Journal of Instrumentation* vol. 11 (2016).

[5] W. Zabołotny et al., "FPGA and Embedded Systems Based Fast Data Acquisition and Processing for GEM Detectors", *Journal of Fusion Energy* vol. 38, pp. 480-489 (2019).

[6] P. Linczuk et al., "Measurement capabilities upgrade of GEM soft X-ray measurement system for hot plasma diagnostics", *International Journal of Electronics and Telecommunication* This volume.

[7] G. Kasprowicz et al., "Fast modular data acquisition system for GEM-2D detector", *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2014* 92902F (2014).

[8] A. Wojenski et al., "Multichannel measurement system for extended SXR plasma diagnostics based on novel radiation-hard electronics", *Fusion Engineering and Design* vol. 123, pp. 727-731 (2017).

[9] A. Wojenski et al., "FPGA-based GEM detector signal acquisition for SXR spectroscopy system", *Journal of Instrumentation* vol. 11 (2016).

[10] P. Kolasinski et al., "GEM detector charge signals sequencer implementation for WEST experiment", *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2019* 111764D (2019).

[11] P. Kolasinski et al., "Serial data acquisition for GEM-2D detector", *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2014* 92902H (2014).

[12] T. Czarski et al., "Serial data acquisition for the X-ray plasma diagnostics with selected GEM detector ", *Journal of Instrumentation* vol. 10 (2015).

[13] P. Kolasinski et al., "Modeling and implementation of the GEM detector charge signals sequencer for the hot plasma tokamak fast diagnostics system", *Przeglad Telekomunikacyjny + Wiadomosci Telekomunikacyjne* vol. 4, pp. 86–91 (2018).

[14] P. Kolasinski et al., "FPGA based fast, low latency serialization system for measurement data", *Elektronika : konstrukcje, technologie, zastosowania* vol. 58, pp. 4–45 (2017).