# Serial Data Acquisition for GEM-2D detector

Piotr Kolasinski[a], Krzysztof T. Pozniak[a], Tomasz Czarski[b], Maciej Linczuk[a], Adrian Byszuk[a], Maryna Chernyshova[b], Bartlomiej Juszczyk[a], Grzegorz Kasprowicz[a], Andrzej Wojenski[a], Wojciech Zabolotny[a], Pawel Zienkiewicz[a], Albrecht Herrmann[c], Didier Vezinet[c] and ASDEX Upgrade Team[c]
[a]Institute of Electronics Systems, 15/19 Nowowiejska Street, 00-665 Warsaw, Poland[1]
[b]Institute of Plasma Physics and Laser Microfusion, 23 Hery Street, 01-497 Warsaw, Poland
[c]Max-Planck-Institut für Plasmaphysik, Boltzmannstraße 2, D-85748, Germany

## ABSTRACT

This article debates about data fast acquisition and histogramming method for the X-ray GEM detector. The whole process of histogramming is performed by FPGA chips (Spartan-6 series from Xilinx). The results of the histogramming process are stored in an internal FPGA memory and then sent to PC. In PC data is merged and processed by MATLAB. The structure of firmware functionality implemented in the FPGAs is described. Examples of test measurements and results are presented.

**Keywords:** plasma diagnostics, GEM detector, FPGA, charge identification, data processing, fast histogramming, VHDL, Xilinx

## 1. INTRODUCTION

Plasma high intensity X-ray detection was tested in ASDEX Upgrade tokamak in Garching (Germany). A two dimensional GEM detector with hexagonal structure was used as 128-channels sensor (see Figure 1) 55. The FPGA based processing was performed to recognise fast energy and time characteristics 55. The serial DAQ as a new type of fast and accurate data acquisition was applied in the experiment.

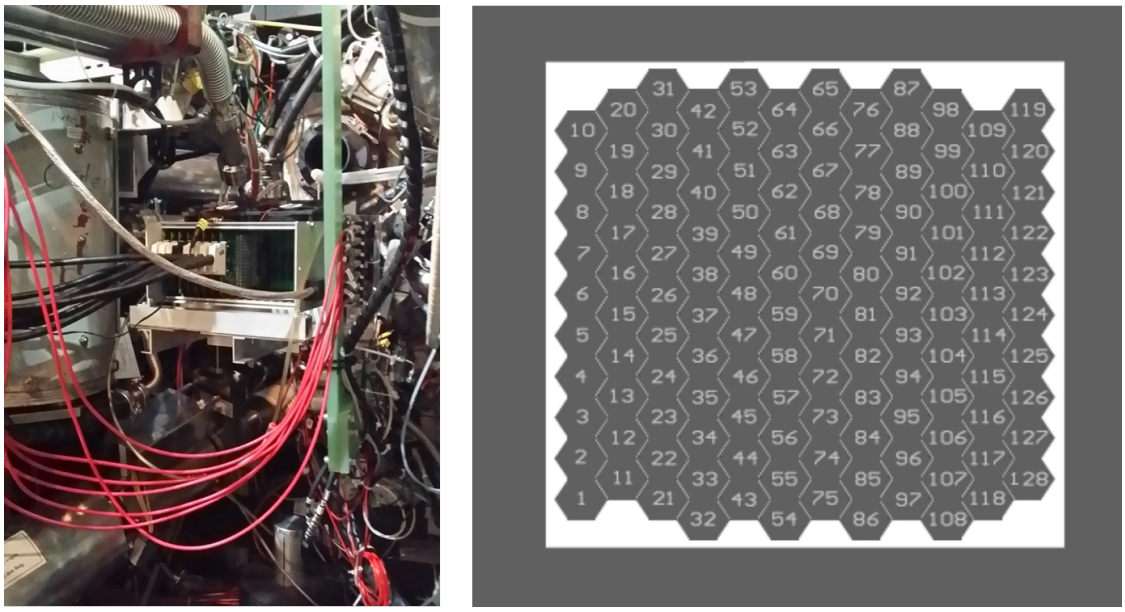

Figure 1. GEM detector and hexagonal structure of anode plate.

[1] pozniak@ise.pw.edu.pl; phone +48 22 234 79 86; fax +48-22 825 23 00; www.ise.pw.edu.pl

## 2. FPGA BASED SERIAL DAQ IMPLEMENTATION

The primary function of the Serial Data Acquisition (SDAQ) module is to collect data from many channels, segregate it and then send it sequentially to further analysis. ASDEX Upgrade experiment configuration of SDAQ is presented in Figure 2.

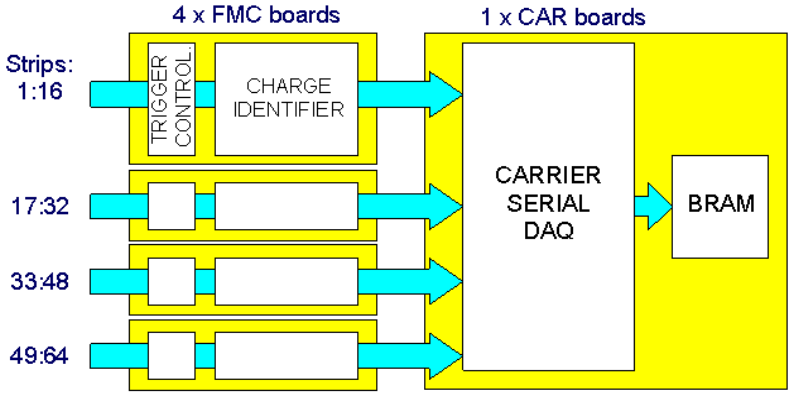


Figure 2. ASDEX Upgrade experiment configuration of SDAQ.

The SDAQ contains 64 channels (from 4 FMCs – FPGA Mezzanine Card) and has been implemented in Carrier FPGA (Spartan6 chip). Results from SDAQ were stored in BlockRAM of FPGA. Two Carrier boards and eight FMCs were used.

Data to this module can come concurrently and independently from all channels. Arrival time of every sample is stored and chronological order of incoming data is preserved at the output. If any sample can't be captured, module passes this information to the output.

Developed module contains four basic parts (Figure 3). The first part, which is the main timer, is responsible for counting clock cycles. Thanks to it, the module is able to calculate at what time every sample arrived. The second part is built from registers which are connected in pyramidal structure. Every next register is connected with two previous ones. Data flows only in one direction. Register can send data only if next register is not storing relevant data. The first level of the pyramid has as many registers as input channels. Samples from channels are shifted from this level through all middle levels to the last one. Every next level has twice fewer registers. This structure allows to catch samples from every channel in the same time and then release them in chronological order. The third and fourth parts are responsible for calculating relative time of every sample and preparing collected data to send to further analysis.

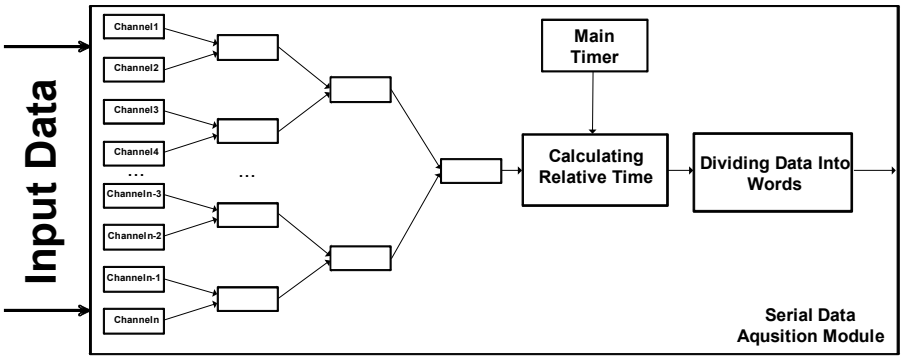


Figure 3. Serial Data Acquisition Module.

Every frame passed through the registers contains four fields: frame type, frame timer, channel number and ADC data (Figure 4). Channel number and ADC data are self-explanatory.

| Frame Type | Frame Timer | Channel | ADC Data |
|---|---|---|---|

Figure 4. Structure of frame.

Frame type gives information about what is stored in the frame. There are two types of frame. “Standard” frames store correct data without any exception. “Sample lost” frames store correct data too and also information that some samples from given channel weren’t caught. This type was created because the module won’t be able to log every sample (e.g. in case of high rate of incoming data) and this loss should be taken into consideration during analysis of data.

In Figure 5, an example is presented, where a “standard” frame is changed into a “sample lost” frame. In cycle *n* register A stores *data 2*, register B *data 1*, and register C *data 0*. In cycle *n+1*, new data – *data 4* is coming to register B, but previous one is still stored there. Thus register B can’t save *data 4*, which is lost and in cycle *n+2*, algorithm changes type frame to “sample lost”. From this moment this information is passed to the output of module.

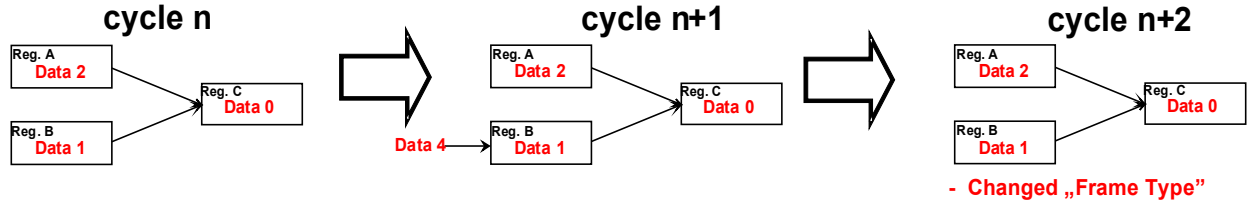


Figure 5. Description of "sample lost" frame.

Field “frame timer” stores information about how long data is being processed in registers. It behaves as a counter. When a sample comes, the counter is set to zero and is then incremented with every cycle. When the sample is sent to the next register, the counter is incremented too. Bigger value stored in this field means that the sample spent a longer time in registers. This information is very important due to proper chronological synchronization of all samples. By checking the “frame timer” values in registers, the module can decide which data should be passed first.

In Figure 6 it is shown how it works. Register A stores *data 0* and register B stores *data 1*. Register C in cycle *n* is empty. Both values in “frame timer” field are different. *Data 0* contains value 18 and *data 1* value 12. It means, that *data 0* came earlier to the module than *data 1*. Therefore *data 0* should be served first and in the next cycle goes to the register C. *Data 1* is waiting in register B till register C is empty again. If *data 0* moves forward then *data 1* will be taken by register C. This procedure gives certainty, that all samples are processed in right order.

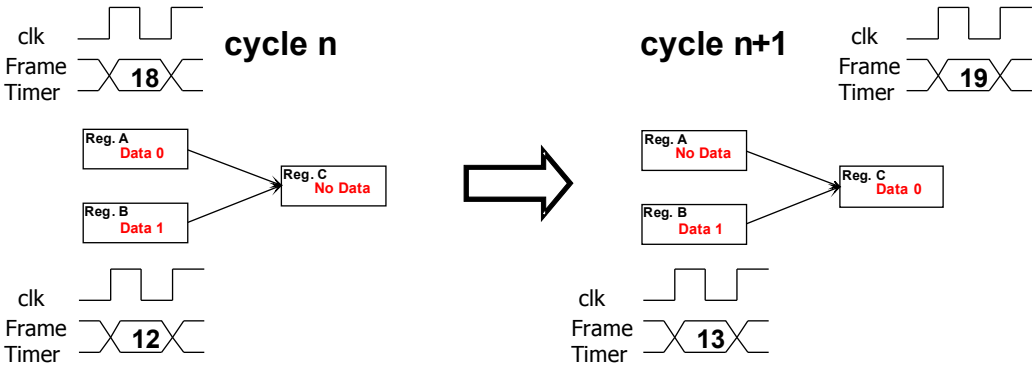


Figure 6. Passing data through registers.

At the output of last register data is chronologically segregated. “Frame timer” gives information how long data is being processed in the structure, but there is no information about time of arrival. In the next step of the algorithm, the relative time of arrival of every sample is calculated. Arrival time is calculated in relation to arrival time of the previous sample. To calculate arrival time, a simple operation is executed. From “main timer” value are subtracted “frame timer” value and absolute time of previous frame. The result of this operation is relative time. After this operation, frame which goes through the module is changed. Instead of “frame timer” field, in the next stages of module, relative time is passed (Figure 7).

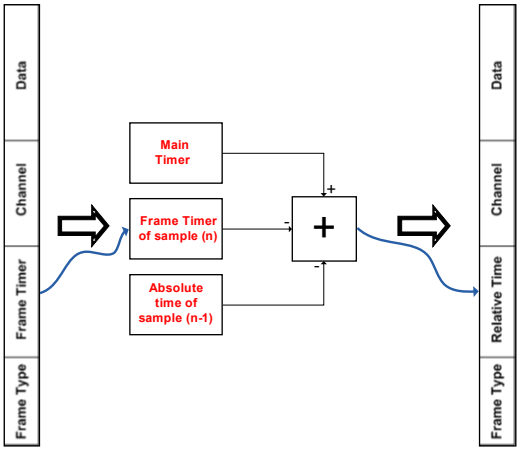


Figure 7. Calculating relative time.

The last step in the algorithm is to prepare data to send to further analysis. If width of output is lower than width of used frame, the frame is divided into smaller parts. Due to sending frames partially it is crucial to properly describe what every part contains and how they connect together again before next analysis. Therefore every created part contains a new field which stores the part number (Figure 8). In the first part are stored all information – ADC data, channel, frame type and LSB of relative time (the longest vector in frame). In the next parts are stored only MSB of relative time. This dividing has another advantage - if the relative time is small, there is no need to send all parts of frame to further analysis. Only significant bits of relative time are sent. Parts which don't contain valuable data are not sent. This solution improves using resources of FPGA and allows to store much more data.

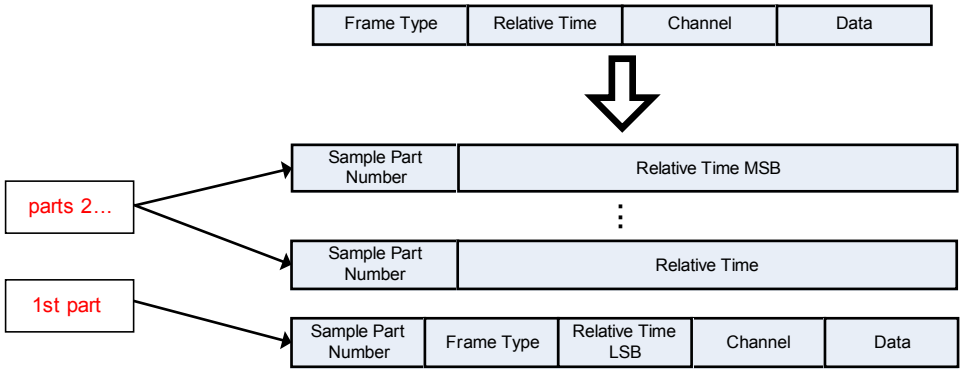


Figure 8. Dividing frames into parts.

## 3. DATA PROCESSING IN MATLAB

A part of the whole system is proposed and developed for processing and merging the data from the GEM measurement of the radiation intensity with a multichannel measuring device. Proposed division of the following blocks:

- Data processing block for a data from one single board,
- Data merging block of data series from two boards into one series of data,
- Chanel renumber block for renumbering channels,
- Time stamps block for changing time stamps.

A data processing system is achieved via combining the blocks with correct parameters. An example of combining these blocks is shown on Figure 9.

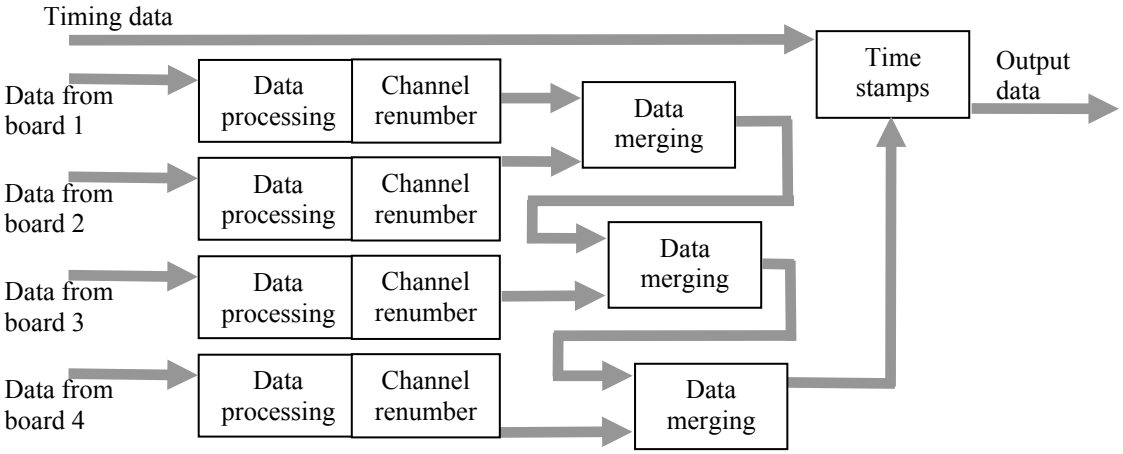


Figure 9. **Data processing system**.

The whole system was verified in MATLAB on real data. The system proposed in this paper may be transferred into other platforms, for example, software systems implemented in an FPGA or graphics card using CUDA technology. Moreover, data formats for the exchange of information between the blocks is proposed. In the case of MATLAB software these formats are two dimensional matrices. For FPGA this format will be a data bus with transmitted signals using them. In the case of CUDA technology data format will be a structure transferred from device memory to memory of graphic card.

First, the data processing block checks input data. Input data must be correct. In case of corrupted data from one or more acquisition boards, this data can't be used for further computation – output data must be empty matrix. It is very important, because corrupted data will corrupt final results of whole physical experiment. Moreover, this block detects an empty data – no events detected on whole detector. This situation is currently represented by empty matrix.

The channel number block is responsible for changing the channel number depending on the physical location and connection of the measuring board. Currently, the block is reduced to convert the channel number. This conversion is being developed – final version will depend on physical detector board.

The time stamp block is responsible for a conversion of time stamp inside the data. The input time is measured in system clock ticks. The output time is measured in seconds. The output time is computed based on timing data from an external clock.

The data merging block is responsible for combining the two data streams into a single one. The main idea of this block is that output data are in correct order – next output data must represent event in detector, which was in the same time as previous event or later events. There is no possibility, that a previous event is represented by data after the next one.

Combining data blocks are connected in cascade. In the case of MATLAB simulations this connection does not matter for efficient calculations. For software algorithm in VHDL suggested another connection, allowing parallelization of calculations.

Inside one time interval, the whole system works according to FIFO idea. Each input data have their time stamp. In case of correct order at the input – next data not represent previous events, output data are also in correct order with the same rule.

## 4. EXPERIMENTAL RESULTS

Serial DAQ event is triggered by ADC samples synchronized by 77.7 MHz frequency, so the time resolution is about 13 ns. The charges are calculated within the given time window of 20 samples for the activated channels. The resulting data structure forms table of vectors: [charge value, channel number, event time] related to the events. Data packages are loaded sequentially to the FPGA RAM memory and finally are conveyed to PC. Several technical characteristics are

initially considered to verify data acquisition reliability. Typical cycles of ASDEX Upgrade plasma sequential shots represented by event counts for nearly one hour monitoring and one second time resolution are presented on Figure 10.

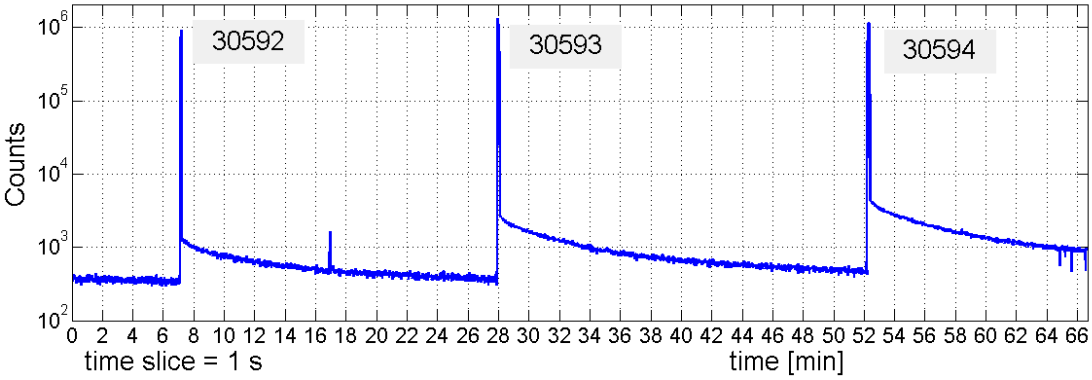


Figure 10. Plasma sequential shots displayed by event counts vs. time.

An individual plasma shot (no. 30601) for interval of 25 s and time resolution of 1 ms is considered at Figure 11. Data structure is represented by points corresponded to the events. Set of events is modeled by the relation (time, charge) and (time, channel) respectively. The actual plasma shot is proceeding during nearly 6 s but due to high intensity radiation less than 0.1 s is a time long enough to fill up the available FPGA RAM memory. Subsequently a gap less than 1 s is required to convey data and to clear out the memory for a next package. After the plasma shot, a residual, low intensity radiation is detected and corresponding data flow fills up the memory during several seconds. Some different memory properties are observed for two carrier boards with specific time of data acquisition. The investigated characteristics allows to estimate required memory size for continuous plasma data acquisition for the final implementation.

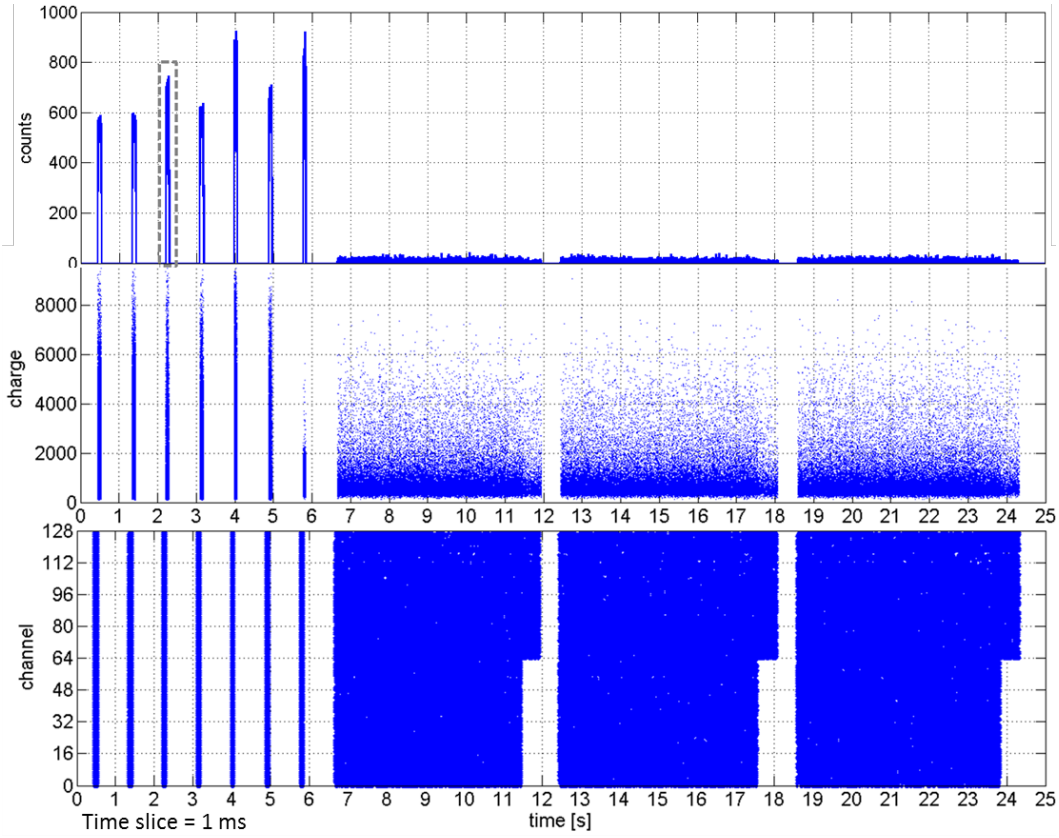


Figure 11. Total counts vs. time (up) and data structure for plasma shot decomposed as (time, charge) relation (mid) and (time, channel) relation (low) for individual plasma shot no. 30601.

Relative frequency of events is presented by time intervals or difference distribution within the range of 1 μs. in Figure 12. Discrete time interval is a multiple of time resolution ~13 ns. The highest counts value for zero interval corresponds to coincides and charge events for the same cluster. Accurate characteristics require cluster identification by investigation of time and space charge distribution. For the given hexagonal structure most of the cluster events are deposited on a single pixel sensor. Therefore charge distribution presented in Figure 13 corresponds to the energy distribution for X-ray radiation. Consequently planar distribution corresponding to the charge events position is presented in Figure 14.

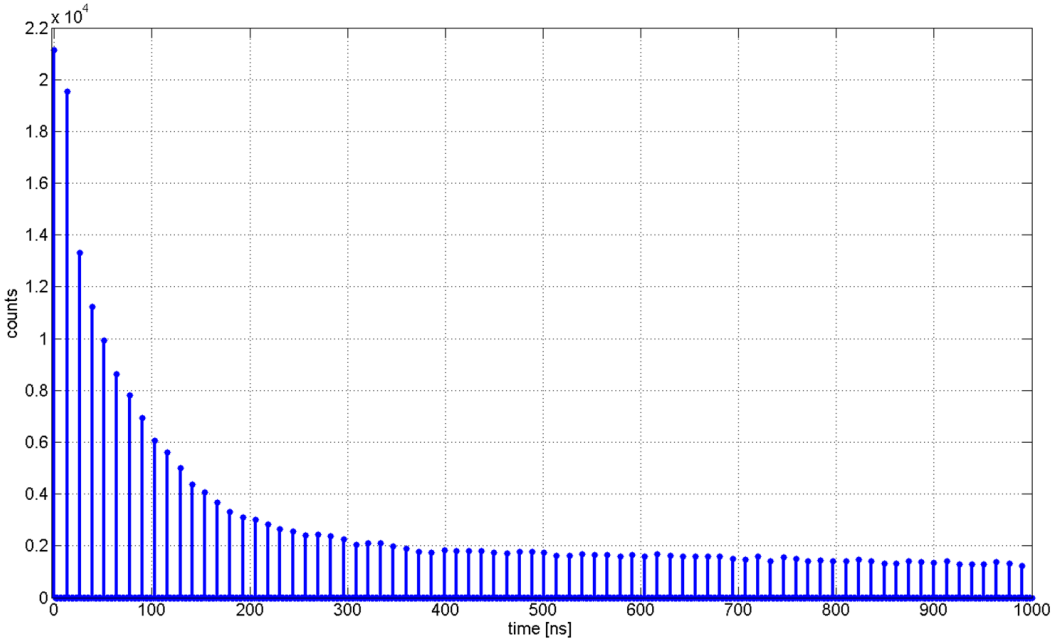


Figure 12. Time intervals distribution.

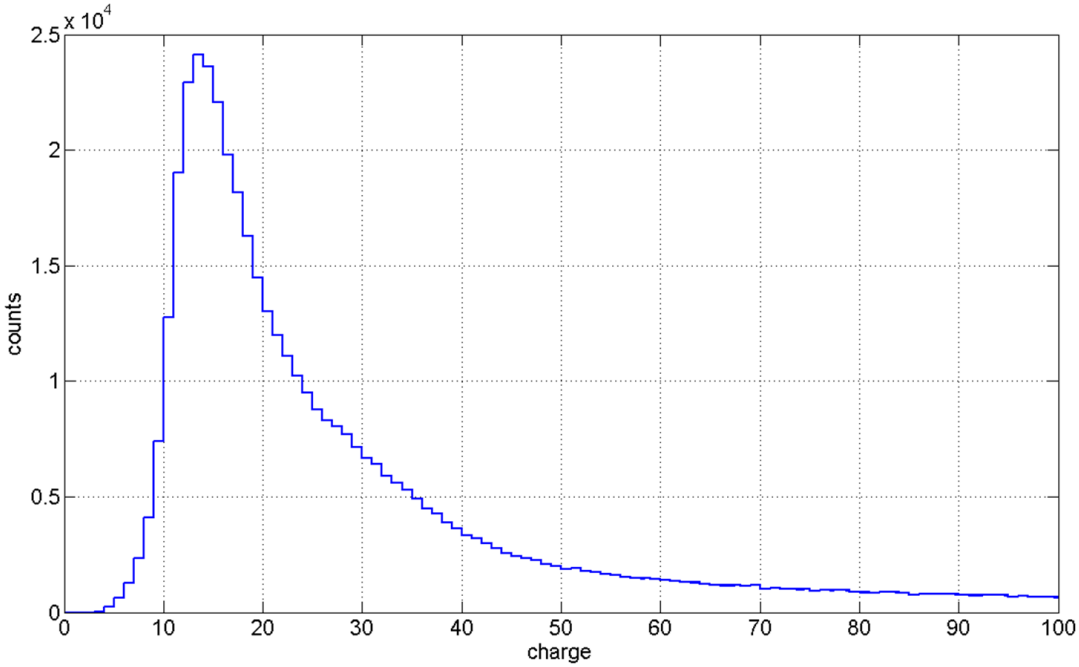


Figure 13. Charge value distribution corresponding to the plasma X-ray radiation energy.

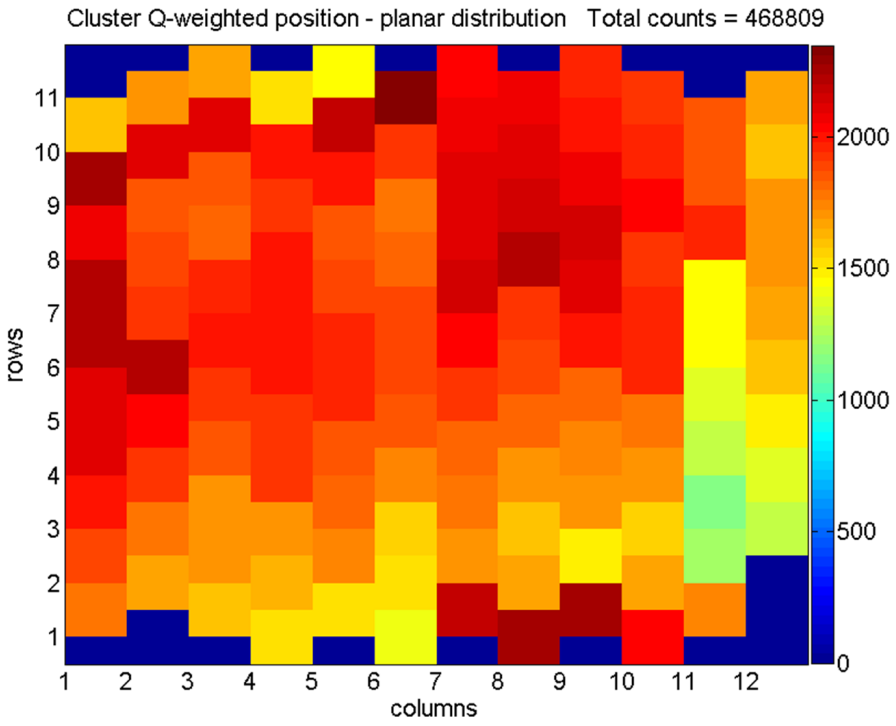


Figure 14. Planar distribution corresponding to the charge events position.

## 5. SUMMARY

A complete firmware for fast data acquisition method and histogramming of charges of the T-GEM detector discussed in the paper has been completed for FPGA chips. The firmware was prepared as a set of configurable components. Components have been written as VHDL behavioral descriptions form. Firmware has been implemented and running for Spartan-6 FPGA family (Xilinx). Collected data was analyzed by MATLAB. Invented and implemented method allowed to acquire and store packages of valid data from detector. Result shows that data can be captured at a very high rates. Described method was used and tested in ASDEX Upgrade tokamak in Garching (Germany).

## ACKNOWLEDGMENT

This work was performed within the strategic research project "Technologies supporting the development of safe nuclear power", financed by the National Centre for Research and Development (NCBiR). Research Task "Research and development of techniques for the controlled thermonuclear fusion", Contract No. SP/J/2/143234.

## REFERENCES

[1] Rzadkiewicz, J. et al., "Design of T-GEM detectors for X-ray diagnostics on JET" Nucl. Instr. Meth. A, 720,36–38 (2013)
[2] Chernyshova, M., et al., "Development of GEM gas detectors for X-ray crystal spectrometry", Journal of Instrumentation, 9 (3), art. no. C03003, 2014
[3] Kasprowicz, G., et al., "Fast ADC based multichannel acquisition system for the GEM detector" Proc. SPIE 8454, 84540M (2012)
[4] Pozniak, K.T., et al., "FPGA based charge fast histogramming for GEM detector", Proc. SPIE 8903, art. no. 89032F, 2013