DEVELOPMENT OF CLUSTERING ALGORITHM FOR PIXEL DETECTORS FOR FPGA

A. Lapkin[a]

[a]*Joint Institute for Nuclear Research, 141980 Dubna, Russia,*

e-mail: lapkin@jinr.ru

Abstract – Modern pixel particle detectors allow possibility to determine coordinates of hit with high accuracy. Some of them, for example Medipix series detectors, provide possibilities to evaluate particle energy deposit. However, pixel detectors have a difficult problem of a dividing of the charge, appeared by interaction with hitting particle, into neighbor pixels. To compensate this effect the clustering procedure is used. The clustering procedure is the process of union of neighbor non-zero pixels. A cluster is a group of pixels with common borders. Sum of energy deposit in pixels of a cluster provide estimation of particle energy with higher accuracy. The clustering procedure usually operates with ready data saved on a computer what required a long time and big amount of memory. Here the clustering algorithm for FPGA to include in DAQ systems of pixel detectors is presented. The including clustering procedure in DAQ reduces amount of memory and time required for data processing.

Аннотация – Современные пиксельные детекторы позволяют с высокой точностью определять координаты попадания. Некоторые из них, например детекторы серии Medipix, предоставляют возможность оценить ионизационные потери частиц. Однако у пиксельных детекторов есть сложная проблема разделения заряда, возникающего при пролете частицы границы пикселей. Для компенсации этого эффекта используется процедура кластеризации. Процедура кластеризации — это процесс объединения соседних ненулевых пикселей. Кластер — это группа пикселей с общими границами. Сумма энерговыделения в пикселях кластера позволяет оценить энергию частиц с более высокой точностью. Процедура кластеризации обычно работает с готовыми данными, сохраненными на компьютере, что требует длительного времени и большого объема памяти. Здесь представлен алгоритм кластеризации для ПЛИС для включения в системы сбора данных пиксельных детекторов. Включение процедуры кластеризации в DAQ уменьшает объем памяти и время, необходимое для обработки данных.

INTRODUCTION

Modern pixel particle detectors provide the ability to determine the coordinates of particle with high precision. Some of them, for example, Medipix family detectors [1, 3], allow to measure the energy of registered particles. However, for all pixel detectors, there is a difficult problem of separating charge carriers arising from one event between neighboring pixels [2, 4, 5].

After the impact of a charged particle, free charge carriers appear in the volume of the reverse-biased semiconductor. The cloud of charge carriers, under the electric field, drifts to the pixel pads and expands by diffusion. As a result, free charge carriers from one particle are collected in different pixels, which leads to the need to combining the amplitude of signals from adjacent pixels to correctly estimate the particle energy deposit. The problem of charge separation is especially relevant for electrons, which can produce a long track in the volume of a semiconductor sensor, and charged particles with an energy deposit greater than 60 keV.

To compensate this phenomenon, clustering procedure is performed. Clustering is the process of combining neighboring non-zero pixels. Typically, clustering is performed on data stored on a computer [7]. This task requires time to process data and a large amount of memory. In this paper, a clustering algorithm on FPGA is proposed for embedding into DAQ of pixel detector. Integrating clustering into a DAQ system allows to reduce the amount of data occupied and speed up its processing.

A cluster is a group of non-zero pixels with common boundaries, surrounded on all sides by pixels with zero data. Non-zero pixels that have a common angle are considered to belong to the same cluster [6].

ALGORITHM

The main feature of clustering on FPGAs is that not all data is received at any given time. A new received pixel can join up to three clusters. For this reason, it is impossible to implement the clustering algorithm as a complex demultiplexer. It is necessary to make matching between already recruited clusters.

In the simplest case, clustering can be done by brute force, pairwise matching, and merging a pair of clusters if it happens that they are part of the same cluster. Newly received pixel data is converted to cluster data format and sequentially matched to all clusters that are in memory. When clusters are merged, the resulting cluster continues to be matched to remained clusters in memory. Then the new cluster is written to memory. Thus, in the cluster memory at each moment of time there are completely pairwise matched clusters that cannot be joined.

Cluster matching require a relatively simple mechanism that allowed cluster merging quickly and could be implemented based on bit operations. One of the simplest solution is to use a detector bitmap for each cluster. If a cluster contains a pixel with coordinates (x, y), then the bits corresponding to pixels (x, y), (x+1, y), (x, y+1), (x+1, y+1) are equal to 1 Then, to match clusters, it is needed to perform a bitwise AND operation between the bitmaps of the clusters, and then an OR operation between all the bits of the result. To calculate the bitmap of a combined cluster, it is needed to perform a bitwise operation OR between bitmaps of clusters.

The proposed algorithm is not suitable for execution on FPGAs because it consumes too much memory and time. The execution time will depend on the number of clusters as n^2. Each cluster will occupy a minimum of 8 KB of memory.

A partial bitmap can be used to reduce the memory size for one cluster. A partial bitmap is not an image of the entire detector, but only a part of it. Since a partial bitmap does not show in which area of the detector the cluster is located, it requires additional parameters, for example, the coordinates of the bitmap center relative to the entire detector. Matching and merging operations in this case require preliminary alignment of bitmaps clusters relative to the entire detector. Applying a partial bitmap does not guarantee correct result.

It is possible to reduce the complexity of the algorithm by setting conditions on the input pixel stream. One of the most obvious conditions might be sorting pixels by {y, x} value. This condition can be quite simply implemented in an FPGA and it is automatically satisfied for some types of pixel detectors. Clusters can add new pixels only from the side of largest y. It is not needed to store information about positions of other pixels. It means that cluster must have only two row bitmap. The highest y row to connect with next row pixels and lower row to connect with current row pixels. If a cluster have not connected with pixel of current row, it means that cluster is completed and there are no pixels which are possible to be joined with the cluster. Using two row bitmap provides reducing necessary memory for each cluster without losing the reliability of clustering. Additionally, the criteria of complete clusters can output ready clusters during clustering. It reduces maximum possible number of clusters in memory. It makes receiving output data faster because next modules do not have to wait until the end of dataflow to get clustered data.

SIMULATION

To estimate time consumption of the algorithm a simulation experiment was used. The simulation were done on Saphir Data Center, which has AMD Alveo U200 cards. The program automatically generated frames pixel by pixel. The probability that a pixel will have data was the same for the entire frame. The probability started at 0.001 and increased every 100 frames by 0.005. A total of 10,000 frames were generated. The obtained data was processed using ROOT software. The dependence of clustering time on the number of pixels in the frame is shown in Figure 1.

The total number of incorrectly clustered frames is 386, that is, 3.86%. In most cases, these errors are caused by a violation of the data reading. The graphs show that with a small number of pixels or clusters, the time increases quadratically. However, when the number of pixels exceeds 5000, the time growth slows down. After a peak at 17000 pixels, time consumption decreases as the number of clusters increases. This happens because with a large number of pixels, each new pixel does not increase the number of clusters, but combines

existing ones. Thus, when adding a new pixel to a frame, the processing time change and the number of clusters decreases.

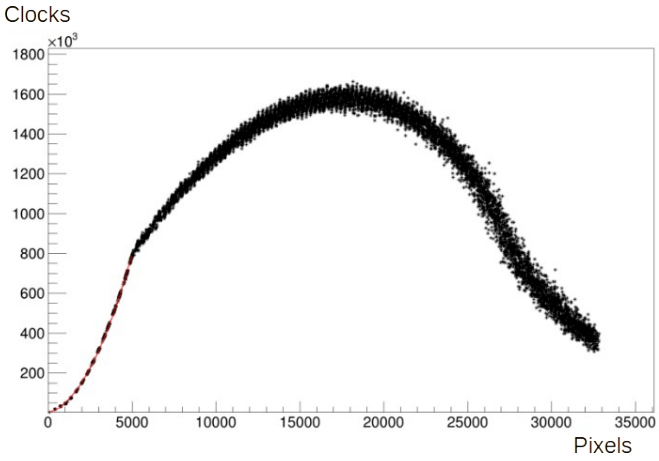

Fig. 1. Dependence clustering time in clock periods on amount of pixels. Range of pixels where the dependence is quadratic was fitted by red line.

From the simulation we can conclude, that the algorithm is suitable for particle flow more than 5000. Such flow can be encountered in high-energy physics experiments and in some other applications, such as multi-energy tomography. In any case, transferring clustering from computers to FPGAs will speed up the clustering, if only because in FPGA clustering begin during readout procedure.

TESTING ON READY-MADE READING SYSTEMS

Two readout systems were used for two different detectors. The first detector had one serial output with the order of pixel packets sorted by (y, x). The second detector had 8 pins and unsorted data order. Both readout systems were based on Intel Cyclone 5 FPGA of different versions. A clustering algorithm was installed in both reading systems and the control software was upgraded. After upgrading, the reading system stores pixel and cluster data. Noise data and data with the Am-241 source were used for the test. The measurements were carried out in the mode of saving pixels and clustering results. After measurements, these pixels were clustered on a computer. The result of clustering on a computer and on an FPGA turned out to be indistinguishable. The only difference was in the calculation of weighted average coordinates, which arose due to sampling errors in the FPGA. Such errors cannot decrease spatial resolution because no one pixel detector can provide spatial and energy resolution that such error would be essential.

CONCLUSION

An FPGA clustering algorithm for pixel detector reading systems was presented. Its functionality was substantiated and confirmed. Using simulation, the dependencies of the calculation time on the amount of input data were obtained. The presented algorithm was built into existing reading systems. The operation of the algorithm in reading systems was tested and its performance was proven. The algorithm is suitable for high-energy physics experiments and multi-energy tomography.

ACKNOWLEDGEMENT

This research was partially supported by the computing infrastructure of the SAPHIR Millennium Institute (ANID, Millenium Program, ICN2019_044)

REFERENCES

1. Medipix Collaboration. URL: https://medipix.web.cern.ch/home (Дата обращения 27.09.2022). – Текст: электронный.

2. Bassi G. A FPGA-Based Architecture for Real-Time Cluster Finding in the LHCb Silicon Pixel Detector / G. Bassi, L. Giambastiani, K. Hennessy, F. Lazzari, M. J. Morello, T. Pajero, A. Fernandez Prieto, G. Punzi // IEEE Transactions on Nuclear Science. - 2023. - 70. - p. 1189-1201

3. Campbell M. An introduction to the Medipix family ASICs / R. Ballabriga, M. Campbell, X. Llopart // Radiation Measurements. – 2020. – 136. – 106271.

4. Mathieson K. Charge sharing in silicon pixel detectors / K. Mathieson, M.S. Passmore, P. Seller, M.L. Prydderch, V. O'Shea, R.L. Bates, K.M. Smith, M. Rahman // Nuclear Instruments and Methods in Physics Research. – 2002. – A487. – p. 113-122.

5. Krzyzanowska A. Measurements of charge sharing in a hybrid pixel photon counting CdTe detector. 22nd International Workshop on Radiation Imaging Detectors (Belgium, Ghent, June 27 – July 1, 2021).

6. Bimatov M.V. Charge collection in X-ray pixel detectors based on SI-GaAs doped with Cr / G.I. Ayzenshtat, M.V. Bimatov, O.P. Tolbanov, A.P. Vorobiev // Nuclear Instruments and Methods in Physics Research. – 2003. – A509. – p. 52-55

7. Meduna L. Real-time Timepix3 data clustering, visualization and classification with a new Clusterer framework / L. Meduna, B. Bergmann, P. Burian, P. Manek, S. Pospsil, M. Suk // Connecting the Dots and Workshop on Intelligent Trackers (CTD/WIT 2019) (Spain, Valencia, 2-5 April, 2019).