Satellite Onboard Data Reduction Using a Risc-V Core Inside an RTG4-Based Data Processing Pipeline

Gasper Skvarc Bozic(✉), Thomas Unterlinner, Tanja Eraerds, Sabine Ott, and Markus Plattner

Max Planck Institute for Extraterrestrial Physics, Giessenbachstr. 1, 85748 Garching, Germany
gaskvarc@mpe.mpg.de

Abstract. The Wide Field Imager (WFI) is one of two scientific instruments onboard the next generation European x-ray observatory ATHENA. It will orbit Lagrange point L2 and send the acquired science data to a single ground station with a downlink that is available for several hours once per day. The data rate of the downlink is a bottleneck, which limits the amount of science data that can be transferred.

Measurement data of the eRosita satellite which is in operation since mid of 2019 shows that a high radiation background generates parasitic sensor data that adds to the science data. In order to remove the parasitic data from the science data stream onboard, a Risc-V softcore processor implementation in the RTG4 FGPA has been studied. Depending on the observation scenario, the data rate is reduced by a factor of more than 50.

Within this article, we describe the WFI onboard processing architecture, the sensor effects on space radiation and the hard- and software architecture of the Risc-V softcore that can be implemented to reduce the data rate on board. Three test cases are defined and executed to verify the performance of the data reduction scheme.

Keywords: ATHENA · WFI · Risc-V · RTG4 · Real-time · Onboard processing

1 Real-Time Data Processing Onboard ATHENA WFI

The Wide-Field-Imager (WFI) is one of two science instruments onboard the next generation x-ray space telescope ATHENA (Advanced Telescope for High ENergy Astrophysics) [1]. Its camera system consists of four large and one fast sensor, sensitive in the energy range from 0.2 eV up to 10 keV. The sensors are based on DEPFET (DEpleted P-channel Field-Effect Transistor) technology, 2-dimensional arrays of 512 × 512 (large), and 64 × 64 (fast) pixels, respectively. The sensors are operated in parallel, independent from each other, in rolling shutter mode. This means that 511 (63) rows are active and record incoming x-ray photons while one row is readout. Since the detection principle shall also resolve the energy of each incoming x-ray photon, the sensors have to be read out at rates of more than 50 Mega-Pixel per second to avoid a pile-up.

© Springer Nature Switzerland AG 2020
A. Brinkmann et al. (Eds.): ARCS 2020, LNCS 12155, pp. 228–238, 2020.
https://doi.org/10.1007/978-3-030-52794-5_17

In order to achieve the pixel read-out rate, each sensor is read out with eight channels operated in parallel. The eight data streams of one sensor are processed in real-time parallel processing pipelines that are implemented inside a Microchip RTG4 FPGA. The processing steps of each pipeline are:

- Offset correction: Every pixel has its individual offset value that is subtracted from the current measurement value.
- Common Mode Correction: Variations in the read-out ASICs cause an offset value common to all values of one read-out line. This offset is subtracted from all 64 pixels of one channel.
- Event and Pattern Filter: The pixel (energy) values are compared to thresholds that span the valid energy range. Pixels within the valid range are flagged as "valid". In case several neighboring pixels show valid values, the pattern is analyzed [2].
- Event List Generator: Dependent on the operational mode, the valid events are selected and spatial coordinates, as well as timestamps, are added to them.

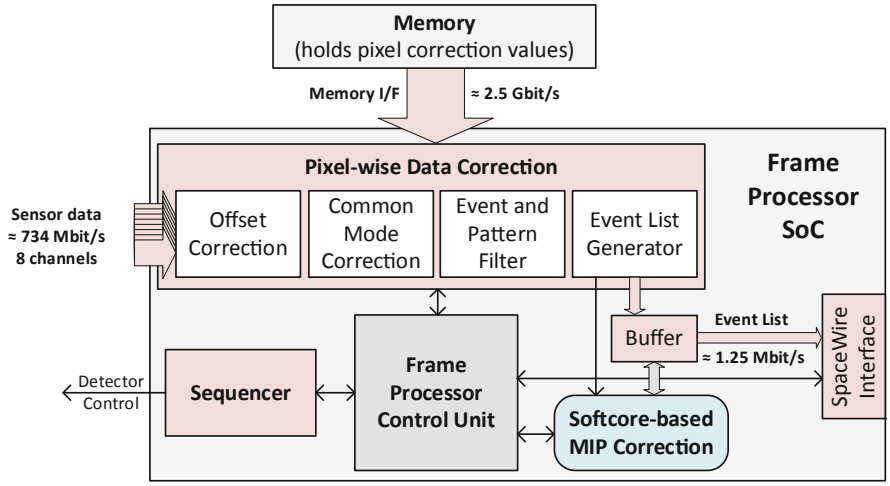

Fig. 1. Block diagram of FPGA (red: real-time processing pipeline, blue: softcore). (Color figure online)

All of these pipeline stages are based on pixel-wise data processing, i.e. each stage executes operations on the data stream of its detector channel without dependency on neighboring channels, see [3] for details. The output of the data processing pipeline is an Event List that includes all event data of the frame, i.e. coordinates of an event (pixel that was hit by x-ray photon), the energy (ADC value) and the time stamp (arrival time of the x-ray photon sampled with the frame rate). The event list of a frame is stored in an FPGA internal buffer (block RAM).

In addition to the real-time data processing pipeline, the FPGA accommodates

- a Sequencer block that generates all dynamic signals required for detector control in rolling shutter mode
- a Frame Processor Control Unit based on a finite state machine
- a Risc-V softcore that processes the event list (see Sect. 3)

The downlink data rate for a given satellite is limited. On the other hand, we want to transfers as much relevant science data to a ground station as possible. It is beneficial if the system allows it to do some data processing on the satellite itself and thus reduce the required downlink data rate. The described RTG4 FPGA processing pipeline already significantly reduces the science data rate. However, since RTG4 FGPA provides a flexible implementation environment, we investigated if the science data rate can be further reduced by implementing a Risc-V softcore for executing a Minimum Ionizing Particles (MIPs) exclusion algorithm.

2 Science Data Disturbance by Ionizing Particles

In the ideal case of observation, the processing pipeline can be adjusted to all the required observation scenarios. Additional events, however, are created in reality due to radiation that is existing in the halo orbit around L2. The particles that cause such events are called Minimum Ionizing Particles (MIPs) and include mainly protons. Pixels directly hit by a MIP receive energies above the threshold of maximum energy range and are removed within the data processing pipeline.

However, a MIP not only creates a direct detector hit but can also generate secondary radiation due to interaction with the structure surrounding the detector system. These secondaries, in turn, create events that cannot be distinguished from valid events generated by x-ray photons. Because of these effects, an additional processing step has to be implemented that identifies events from the event list that are located within a certain distance from a MIP event and remove them from the list. The area around the MIP event that could contain secondary events caused by the MIP depends on several parameters: MIP energy, incident angle, material (type and thickness) crossed by the MIP, etc. Geant4 simulations are carried out, taking into account these characteristics of incoming particles and the detector surroundings.

This simulation approach has been verified using data from the eRosita mission [4]. Although eRosita has a different type of sensor, the sensitivity regarding MIPs is comparable. Measurement results obtained by the eRosita cameras in orbit around Lagrange point L2 have been used for model correlation and yield an agreement within 10% between simulation and measurement results. Applying the same approach to ATHENA WFI with an adapted simulation model yields an average rate of 2 MIP/cm^2/s and an average count of 10 pixels that are hit by one MIP directly. This results in an average value of 0.41 MIP per frame of a large sensor and 10^{-4} MIP per frame of the fast sensor.

3 RISC-V Softcore Architecture and MIP Removal Algorithm

As described in Sect. 1, the real-time data processing is implemented in FPGA-based pipelines. Data from all pixels flow through the pipeline stages and additional information based on the processing stages is added in the form of flags. At the end of the pipeline, that Event List Generator selects the valid events and forwards them into an FPGA internal buffer memory. This is the place, where the raw data has been reduced by two to three orders of magnitude. Based on the nature of the task responsible for identifying the events within the defined region of a MIP as described in Sect. 2, it would be difficult to implement the required logic in the FPGA fabric. Therefore, a softcore microprocessor is used to perform the required task.

3.1 Softcore Microprocessor Architecture

Figure 2 depicts the softcore microprocessor architecture where blue and green blocks represent components implemented in the FPGA fabric. Whereas, red blocks represent components external to the FPGA. Green blocks indicate fabric interfaces with the physical world. The architecture is based on a 32-bit Risc-V CPU with a 32-bit AHB internal interconnect. For the design presented in this paper a Risc-V soft IP core (MiV_RV32IMA_L1_AHB) from Microsemi was used. This Risc-V soft IP core has a separate bus for memory and memory-mapped peripherals. The memory bus is connected to a DDR3 memory controller which interfaces the external DDR3 memory where program data is stored.

Several different peripherals were implemented. The most important is the dual-port SRAM with APB wrapper. This is the interface between the fabric parallel data pipeline and softcore microprocessor as depicted in Fig. 1. Other peripherals such as GPIO and UART are used for debugging purposes and provide an interface between the microprocessor and a PC. All peripherals are connected via the APB bus and through the AHB to APB bridge to the main interconnect. A second APB bus is used as an interface between the CoreABC processor and the memory controller. The CoreABC is a small co-processor used for external memory configuration and initialization. After power-up, certain configuration registers in the DDR3 external memory have to be configured.

The softcore microprocessor implementation was designed for the RTG4 target device with a system clock running at 50 MHz. The JTAG component included in the architecture enables the programming of the microprocessor and advanced debugging capabilities.

As an interface between fabric and microprocessor two options are possible. One is the dual-port SRAM and the other is FIFO buffers. At this point, dual-port memory was chosen as it simplifies testing. Currently, we do not have a dummy data generator that could fill the FIFO buffer and mimic the parallel data pipeline, because the interface between parallel data pipeline and softcore microprocessor is not yet completely defined. With SRAM we can write the generated event list at the beginning of program execution to the SRAM. The dual-port memory is connected to the APB peripheral bus

for simplicity. In case the memory penalty is significant then the SRAM can be moved directly to the AHB bus and improve the performance as long as the burst transfer mode of the AHB bus is utilized.

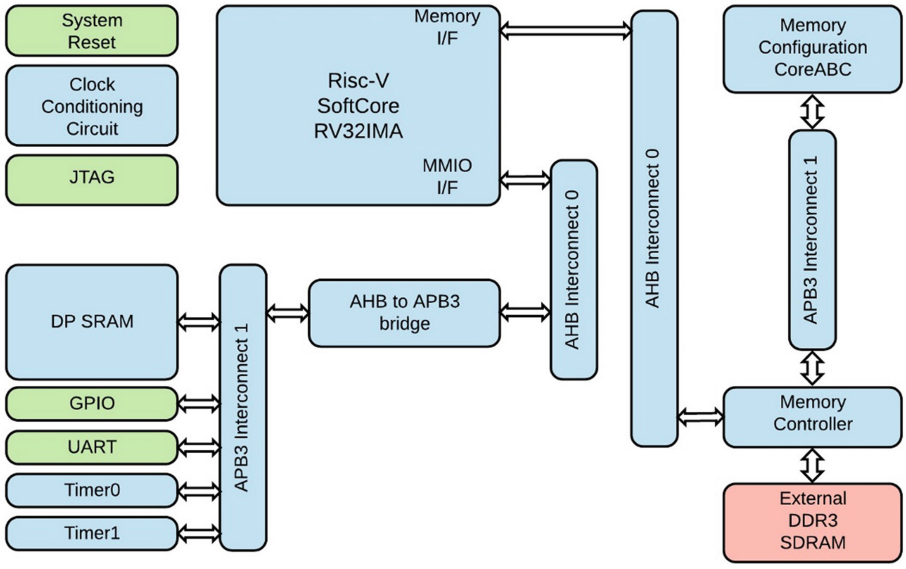

Fig. 2. Softcore microprocessor architecture with AHB internal interconnect. (Color fgure online)

3.2 Event List Structure

The input to the algorithm running on the Risc-V microprocessor is an Event List, which is the output of the frame processing pipeline. Its structure is shown in Fig. 3. For each frame, the Event List header is generated that contains the time stamp in the form of the frame counter and additional housekeeping data e.g. threshold values used in the pipeline stages. Each event is represented as one line that contains amongst others the following data:

- Energy value (bits number 36 down to 23): The 14-bit output of the ADC represents the energy of the pixel
- Line address (bits number 22 down to 14): The 9-bit value corresponds to line number 0–511
- Pixel address (bits number 13 down to 5): The 9-bit value corresponds to the column number 0–511
- Flags (bits number 4 down to 0): Information gained by the pipeline stages and added to each event indicating, for example, the results of threshold comparison.

Figure 3 illustrates the event list and shows a frame. The blue dots represent pixels illuminated with x-ray photons.

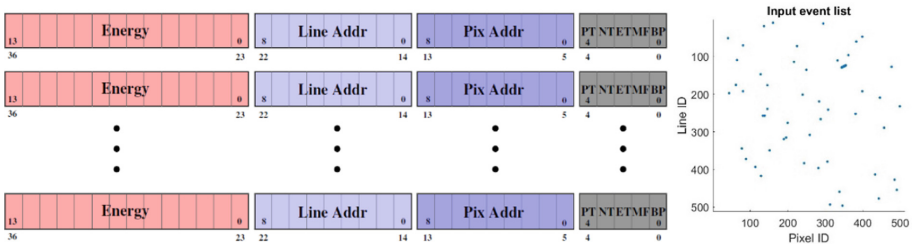

Fig. 3. Event list format and example frame. (Color figure online)

3.3 MIP Pixel Exclusion

Since the input to the algorithm is already a reduced dataset in from of an Event List it is not beneficial to reconstruct the frame in the microprocessor as this introduces substantial memory penalty and usage. Therefore, the generated Event List is considered as data point dataset. Each MIP event is represented by multiple pixels where each pixel corresponds to one data point in the generated Event List. When the Event List is passed to the processor, the processor does not know which event pixels belong to one MIP event even though this would be obvious if one would plot the data points. Therefore, it is necessary to perform a clustering algorithm in order to identify different MIP events and the number of them. However, because prior to clustering the number of clusters is unknown a hierarchical clustering algorithm or some other form of non-parametric clustering algorithm is needed.

The algorithm presented in this paper uses the DBSCAN clustering algorithm [5] since it is a non-parametric density-based clustering algorithm and suits this application well. Once all the clusters or rather MIP event groups (tracks) are identified it can be determined if other events occurred within the MIP event region. In order to do this, the region around each MIP event has to be defined. In our case, an elliptical exclusion area around the MIP tracks was chosen. First, a centroid is calculated for each cluster and other ellipse parameters based on mathematical equations presented in [6]. Based on these parameters an ellipse border is computed and with it the region around the MIP. Afterward, the non-flagged events can be checked if they fall into any of the computed regions if they do, they are flagged which indicates that they belong to a MIP event as explained in Sect. 2.

The algorithm depicted in Fig. 4 was first tested as a MATLAB script where one can also visualize all the results and prove the algorithm correctness. After successful test with MATLAB scripts the algorithm was rewritten in C to test it on the Risc-V microprocessor.

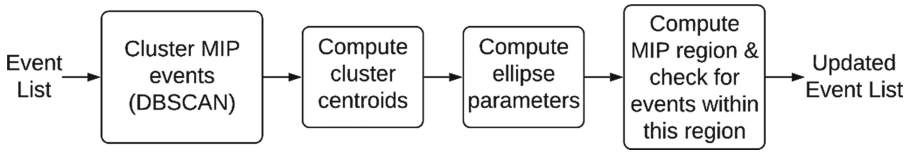

Fig. 4. Secondaries detection algorithm flow diagram.

Prior to described algorithm above, we used a simpler algorithm for detecting MIP events. It is based on an insertion sort algorithm with time complexity of $O(n^2)$, which is comparable to the worst-case time complexity $O(n^2)$ of the DBSCAN algorithm. Based on the results from Tables 1 and 2 it was determined that an algorithm with time complexity $O(n^2)$ is a feasible solution for our target application. However, the algorithm based on insertion sort had a significant drawback as it could not distinguish between two MIP events with either the same Line addresses (y coordinate) or the same Pixel addresses (x coordinate). Therefore, a new algorithm for identifying MIP events was needed.

Table 1. Performance for processing one event list with one MIP event (insertion sort)

Number of events	Flagged events	Cycles	Execution time [ms]
750	189	382500	7,65
500	113	256700	5,10
350	22	171100	3,42

Table 2. Performance for processing one event list with multiple MIP events (insertion sort)

Number of events	Flagged events	Cycles	Execution time [ms]
780	206	515900	10,30
505	101	328100	6,97
350	22	202400	4,04

As can later be seen from results in Sect. 4, the current C implementation of the new algorithm is nowhere near required timing constraints. However, we are confident that with optimized range query function we can achieve similar results as we did with the insertion sort based algorithm, if not better.

4 Experimental Results

Three different scenarios have been tested for the new algorithm:

- Test 1 has been performed with a constant number of ~50 events with an increasing number of MIP events (with an average of 10 pixels per MIP).
- Test 2 always included 3 MIP events per frame and the number of events has gradually been increased.
- Test 3 has been performed with a constant number of ~50 events, a constant number of MIP events (3 MIPs per frame), and an increasing number of pixels per MIP event.

Test 2 and test 3 demonstrate the timing complexity of the DBSCAN clustering algorithm which is expected to be $O(n^2)$ in the worst case, result of using a linear search. Algorithm is affected by the Event List growth and it does not matter which number of events increases be it either MIP or regular events. However, test 2 has worse performance as significant number of regular events introduces additional timing penalty because of another linear search in the last block of the algorithm depicted in Fig. 4.

It should be pointed out that these results can be affected by the size of a MIP region since more events can fall into a region. However, this is only significant when a large number of events are present in a frame, and more events need to be processed. Moreover, for each test case, a random Event List was generated.

Figure 5 depicts the result from the measurement of the execution time of the function executing algorithm as described in Fig. 4. The execution time linearly increases with the increasing number of MIP events. The dotted trendline (linear approximation) also confirms this behavior.

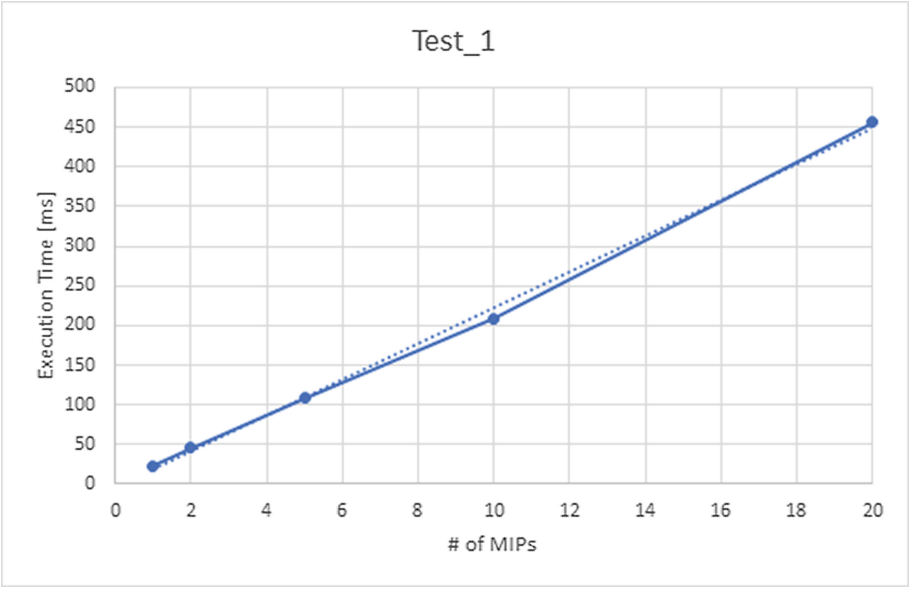

Fig. 5. Execution Time of the algorithm compared to the increasing number of MIP events per frame

Figure 6 depicts the relation between the execution time and the increasing number of events per frame. As seen by the dotted trendline the execution time follows an $O(n^2)$ the worst-case timing complexity characteristic in the case of the DBSCAN algorithm.

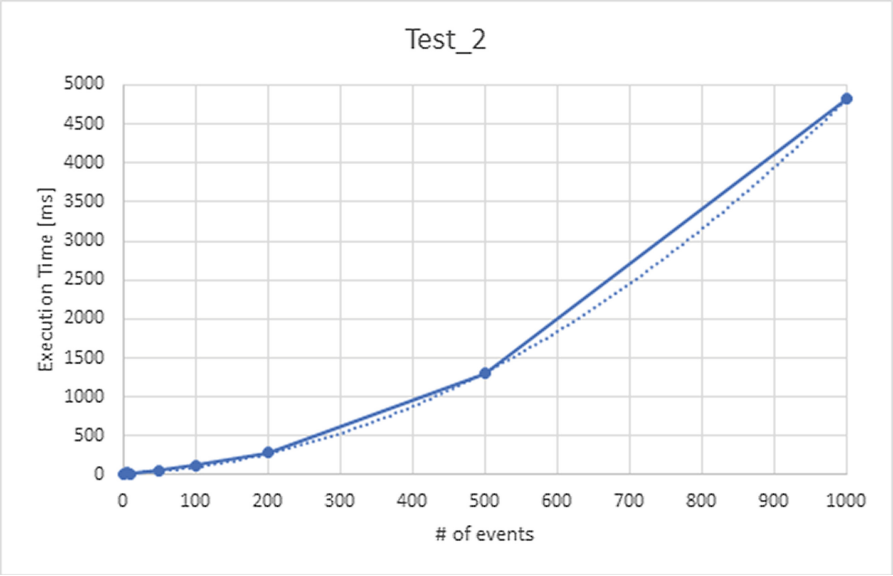

Fig. 6. Execution Time of the algorithm compared to the increasing number of events per frame

Figure 7 depicts the measurement result when the number of pixels per MIP event is increased. As seen by the dotted trendline the execution time follows an $O(n^2)$ the worst-case timing complexity characteristic in the case of the DBSCAN algorithm.

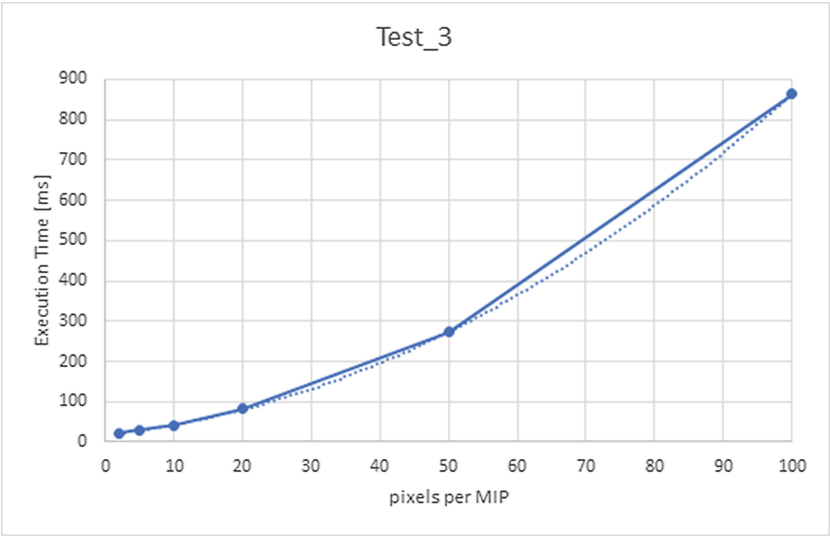

Fig. 7. Execution Time of the algorithm compared to the increasing number of pixels per MIP

Figure 8 depicts visual results from the algorithm described in Sect. 3.3. Generated Event List is shown in the left figure where events are represented by blue dots. Generated Event List is plotted in a 512 × 512 pixel frame in order to have a visual representation of the sensor array. The figure on the right depicts detected MIP events with their elliptic regions. The color of MIP events is not important and it is there just to visually distinguish between different MIP events. Two sizes of ellipse regions are shown, where smaller regions marked with the red color include fewer events from the Event List as opposed to larger regions marked with blue color.

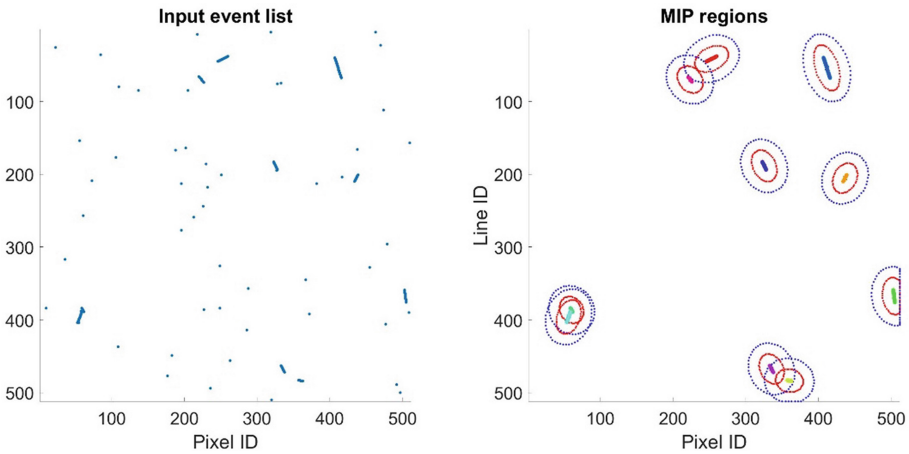

Fig. 8. Visualization of the algorithm results described in Sect. 3.3. (Color figure online)

5 Conclusion

In this paper, a use case of a softcore microprocessor in space application was presented. In particular, a MIP pixel exclusion algorithm running on a Risc-V softcore microprocessor as part of the WFI onboard processing architecture. The softcore microprocessor architecture was based on a 32-bit Risc-V RV32IMA CPU with a 32-bit AHB internal interconnect.

Two different approaches were tested for the MIP pixel exclusion algorithm. One based on an insertion sort algorithm, which proved that algorithms with time complexity $O(n^2)$ are a feasible solution for our application. However, it turned out it is not suitable for all scenarios. Therefore, a second more robust algorithm concept based on data clustering DBSCAN algorithm was presented. Its executions times were far off from required timing constraints. The most significant drawback of the current DBSCAN based algorithm is the range query function. Currently it is implemented as naïve linear search of the Event List which results in saver timing penalty and thus making the DBSCAN algorithm of timing complexity $O(n^2)$. Mover, another search for elements is performed in the final stage of the new algorithm which introduces additional timing penalty.

Nevertheless, with DBSCAN having a worst-case time complexity of $O(n^2)$ we are confident that with an optimized range query function based on r*-tree or kd-tree data indexing structure we can achieve similar results, if not better, as with the insertion sort based algorithm. With the use of data indexing structure the expected timing complexity of DBSCAN is O(n * log(n)).

Our next steps are to write an optimized C code for the range query, run the test on the Risc-V softcore microprocessor, and see if our speculations are correct.

References

1. Nandra K., et al.: The hot and energetic universe – a white paper presenting the science theme motivating the ATHENA + Mission. http://www.the-athena-x-ray-observatory.eu
2. Schanz, T., et al.: A fast one-chip event-preprocessor and sequencer for the Simbol-X LowEnergy detector. Nucl. Instrum. Methods Phys. Res. A **624**, 392–395 (2010)
3. Plattner M., et al.: WFI electronics and on-board data processing. In: Proceedings SPIE 9905, Space Telescopes and Instrumentation 2016: Ultraviolet to Gamma Ray, 99052D, 11 July 2016. https://doi.org/10.1117/12.2235375
4. Meidinger, N., et al.: Development of the focal plane PNCCD camera system for the X-ray space telescope eROSITA. Nucl. Instrum. Methods Phys. Res. A **624**, 321–329 (2010)
5. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters. In: KDD-96 Proceedings, pp. 226–231. AAAI (1996)
6. Haralick, R.M., Shapiro, L.G.: Computer and Robot Vision, vol. 1. Addison-Wesley Publishing Company, Boston (1992)