FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Computational module for on-line data processing for miniaturized particle tracker for radiation monitoring in space
Student:	Václav Lepič
Supervisor:	Ing. Jan Jakubek, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2020/2021

Instructions

The state-of-the-art radiation imaging detector Timepix3 developed in CERN is a position, energy, and time-sensitive: For each ionizing particle, it digitally registers its position, energy, time of arrival and track shape. Since ionizing particles create type-specific patterns they can be often identified including some additional properties (e.g. direction, polarization, electric charge). This detector presents an ideal tool for monitoring and analysis of mixed radiation fields in various environments such as nuclear power plants, hadron therapy or space.
The effective data processing is a key component for the proper utilization of Timepix3 technology. The ultimate goal is to design the on-line data processing tool-box and prepare it for porting to miniaturized platform of MiniPIX TPX3 device with ARM cortex M7 CPU. This student project will be solved in the frame of ESA project MIRAM (Miniaturized Radiation Monitor for communication satellites) by CTU and ADVACAM.

Electronically approved by doc. Ing. Jan Janoušek, Ph.D. on 5 January 2020 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Computational module for on-line data processing for miniaturized particle tracker for radiation monitoring in space

Václav Lepič

Department of Theoretical Computer Science
Supervisor: Ing. Jan Jakůbek, Ph.D.

May 13, 2021

Acknowledgements

I would like to thank my supervisor for his patience with me. My failure to finish this is purely my fault.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on May 13, 2021 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Lepič, Václav. *Computational module for on-line data processing for miniaturized particle tracker for radiation monitoring in space.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Tato práce není a nebude dokončená. Rád bych to zkusil znovu s novým zadáním.

Klíčová slova Nedokončené, MIRAM, Vesmír, Timepix, Medipix, Timepix3

Abstract

This work is not finished and it never will be. I would like try again with a new assignment.

Keywords Unfinished, MIRAM, Space, Timepix, Medipix, Timepix3, Particle tracking

Contents

List of Figures

Motivation

Imaging

Being able to identify individual particles can be very useful in imaging. For example background radiation can be filtered in software before creating final image. Also because of effect called charge sharing many particles, even photons, can take up multiple pixels, we can calculate it's position high precision, thus significantly increasing resolution.[5]

Filtering the data in software has some advantages, even if it could be done by physical filters. Since the unwanted signal is not blocked during the measurement, multiple different settings can be tried and chosen the best of them, without the need for redoing the measurement.

Particle tracking

Timepix detectors can intercept many kinds of particles, including X-ray photons, muons, beta radiation, alpha radiation and heavy ions. All of these light up a track of pixels on the detector.Some of them are long and only a few pixels thick, some are round. Because individual particles can arrive around the same time, it is necessary to identify individual particles before trying to classify them.

dosimetry

with Timepix and Timepix3 it is possible to not only calculate total energy deposited, but also to identify almost every particle and thus give accurate information about the dose someone in that area would receive. Simple example of place, where this is useful is the International space station, where they use Timepix detectors for radiation monitoring.[6]

MIRAM

Short for Miniaturised Radiation Monitor, MIRAM is a Project of ESA. While technical proposal talks about using Timepix, or Timepix2 [7], which was being developed at the time of proposal, current prototype uses Timepix3 [**?**], which actually finished its development sooner (see in which year were [8] and [9] published).

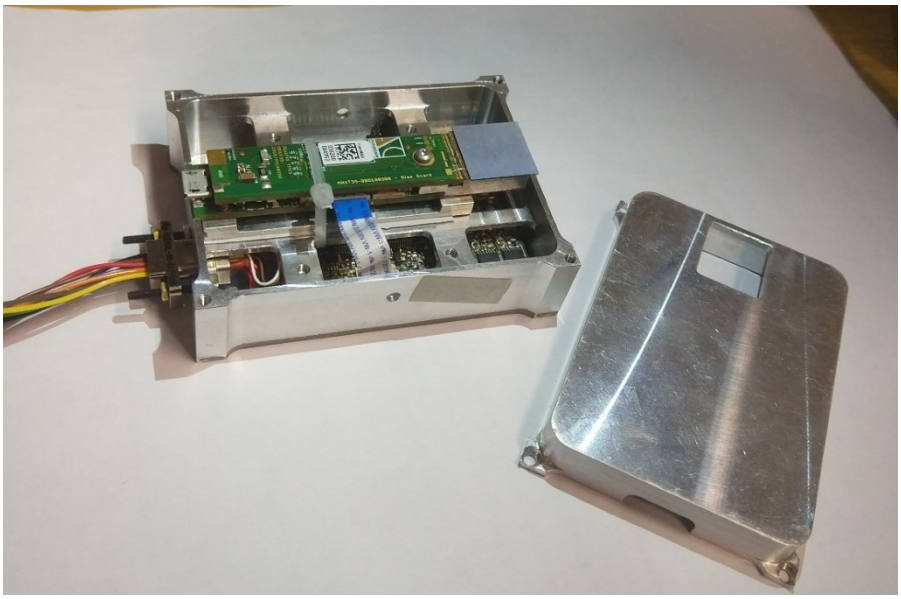

Figure 0.1: MIRAM current prototype [1]

Chapter 1

State-of-the-art

1.1 Indirect conversion

Indirect detection systems first convert the photon into visible light using a scintillator and then visible light is detected by photodiodes. This degrades the spatial resolution and detection efficiency.[10]

1.2 Direct conversion

In a direct detection system, photons are converted directly to electrical signal, without using visible light as an intermediate step.

1.2.1 Photon counting detectors

Photon-counting detectors (PCD) are sensitive enough to allow for detection of individual particles. Unlike conventional imaging devices which integrate charge and measure it during frame readout, PCDs are constantly comparing charge with set threshold and incrementing counter, present in each pixel, whenever threshold is reached. This has few key advantages over traditional methods which measure charge at the end. One of the undesired properties of PCDs is so called dead time. It is the time that has to be between events to avoid being counted as one event. To archive good results, dead time must be significantly smaller than mean time between events. [10]

Another disadvantage for purposes of this thesis is that I don't have access to them.

1.2.2 Hybrid pixel detectors

HPD (hybrid pixel detector) consists of 2 parts - readout chip and sensor, which are bump-bonded together. Huge advantage of this approach is the

ability to connect sensor and readout chip made of different material, meanwhile monolithic devices have to be made out of silicon. [11] Even if silicon is to be used, different sensor thickness can be used without making major change to the rest of the device.

1.2.3 Medipix Timepix

Timepix is a Hybrid pixel

Timepix can operate in 3 different modes [12]

- Event count
  - Measures how many times has pixel been hit
- ToA
  - Measures when the pixel was hit for the first time
- ToT
  - Measures energy deposited

Readout is done via 256 rows, where one bit at the time from each column is read and placed into a 256 bit fast shift register. During readout counters in pixels behave like shift registers, where single bit from each column is placed into 256-bit register, which is then read out and the cycle continues.[13] This approach is simple, but requires entire matrix to be read out even if most of it are zeros.

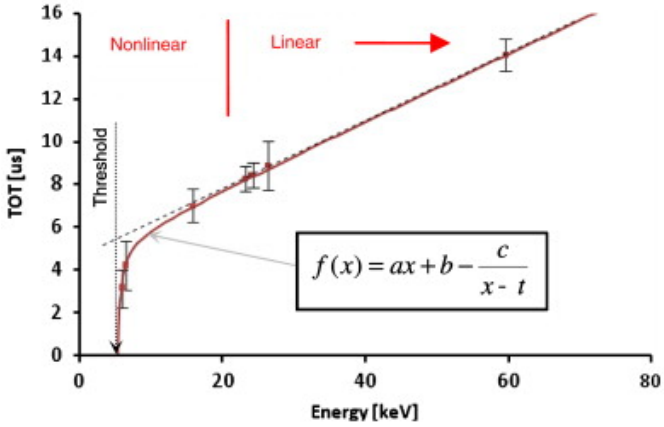

Figure 1.1: Relation between ToT and energy [2]

1.2.4 Timepix3

Timepix3 is a successor to to the Timepix. By simply looking at the number, it might seem that it is successor to the Timepix2, but it is not, otherwise it wouldn't be possible for [8] to introduce Timepix3 in 2014, while [9] which introduced Timepix2 was published in 2020. Just like it's predecessor it has 256x256 grid of pixels, each of size 55×55 m[12][8].

Main improvements of Timepix3 over Timepix[12] are as follows:

- It supports data driven mode
  - individual pixels can be read out independently while keeping rest of the detector sensitive
- There are 3 counters in each pixel, compared to 1 in Timepix
  - This allows for storing ToA and ToT at the same time
-

There are 2 readout modes and 3 pixel modes. Readout mode affects how will individual pixels be accessed, pixel acquisition mode affects meaning of counters in individual pixels. It can operate in zero-suppressed frame-based readout mode, but it's main new feature is data-driven read- out mode. In this mode every pixel hit is processed individually, without the need for readout of the entire matrix. One of the advantages of this approach is that it allows for readout to be done simultaneously with data acquisition, thus keeping the detector sensitive at all times.[8]

1.2.4.1 Chip description

There are 3 pixel acquisition modes.[14]

- ToA & ToT
  - contains information about when was pixel hit and energy of the event
- Only ToA
  - as above, without storing ToT
- Event count & integral ToT
  - useful in matrix readout mode
  - contains information about number of hits and total energy

Since Timepix3 has more features than original Timepix, it is significantly more complex. Its pixel matrix is divided into double-columns, which are then subdivided into SuperPixels. Each SuperPixel is 2 pixels wide and 4 pixels tall. As you can see in ??, individual pixels are quite similar to original Timepix[13], however there are few key differences. One key difference is that in Timepix3 there are 3 counters in each pixel.[8]

Counter sizes are 14, 10 and 4 bits. The 14bit register which can be used for ToA using Gray encoding, or for iToT, in which case it operates as LFSR. The 10bit register is used for ToT or photon count and operates as LFSR. The 4bit register can contain either fine ToA, as a binary number, or photon count, in which case it operates as LFSR.[14] What is significantly different than the original Timepix is the readout mechanism. For every 8 pixels, there is one SuperPixel, which communicates with the End-of-Column block. Depending on mode, data from SuperPixel can be sent as soon as they arrive, or wait for external readout command. If operating in data-driven mode,[14]

1.3 Timepix 3 readout systems

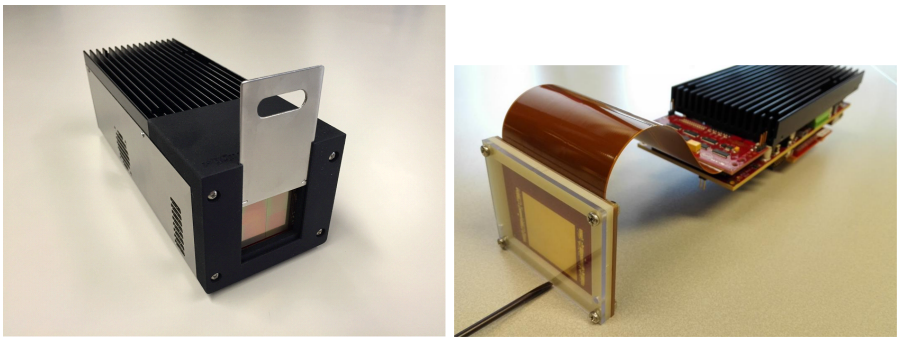

Figure 1.2: SPIDR with and without housing[3]

1.3.1 SPIDR

SPIDR (stands for speedy pixel detector readout) is powerful readout system for Timepix3 chip, developed by NIKHEF. It can read out Timepix3 at it's full data rate of 80 Mhits/s/chip. It connects to C with either 10 GbE or 1 GbE. It also supports Medipix3 chip and other chips might get support in the future by reprogramming the built-in FPGA. [3]

1.3.2 Katherine

Katherine is Ethernet readout interface for Timepix3 developed by IEAP CTU and Faculty of Electrical Engineering, University of West Bohemia. It's main

benefit is operation at up to 100m from computer. It features 1 GbE connection, which on one hand limits it's maximal readout speed to 16MHits/s, on the other hand it makes it compatible common laptops. It also contains reasonably powerful ARM Cortex A9 processor, making it capable of operating as stand-alone radiation detector. [4]

Figure 1.3: Katerine readout [4]

1.3.3 MiniPIX TPX3

MiniPIX is low power radiation camera developed by Advacam, featuring timepix3 chip. It can be equipped with Si or CdTe sensor. It's advantages include small dimensions (80mm 21mm 14mm), weight of only 30g and comparatively low cost. It connects to a PC via USB 2.0 which is also used for power delivery. Being limited to USB 2.0 speeds means that it is limited to readout speed of 2.35MHits/s. [15] If better performance is required, AdvaPix TPX3 can be used instead.

1.3.4 AdvaPIX TPX3

AdvaPIX TPX3 is high performance camera from Advacam featuring Timepix3 chip. It has readout speed of 40MHits/s and connects to PC via USB 3.0 connection. [16]

1.4 Data processing

1.4.1 PIXet

PIXet is data acquisition and processing software developed by ADVACAM. It allows for making measurements, configuring the devices and viewing the data. It allows for python scripting and can be extended with plugins.[17] It contains plugins for track processing and clustering which I will use to check if I have correct results.

Chapter 2

Outputs

2.1 Clusters

2.1.1 Cluster properties

There are many different properties that could be calculated for each cluster. Here are some examples of cluster properties:

- ToA
- Total energy
- Center of gravity
- Size
- Height
- Roundness
- Polar angle
- Elevation angle
- Standard deviation of energy
- Standard deviation of distance along axis
- Linear energy transfer

2.2 [*.clog]

In clog file, everything is separated into frames. Frames can either be actual frames when operating in the frame mode, or just sets of clusters that are close enough in time. This format is great for experimenting with new cluster

properties. I used this format for testing if I get correct results and it worked great for that. However, many uses (e.g. filtering noise from image) require to calculate cluster properties histograms Output of embedded version are just histograms of some values. Which values will be used is subject to change. Reason why histogram was chosen as the output of the embedded version is simple: it takes up much less space compared to formats that save information about individual clusters. It does sacrifice a significant amount of information and I am aware of it, however there are significant issues with data storage todo: insert info about ram size and communication bandwidth.

2.3 [*.elist]

As an output file format was chosen elist. todo describe this differently File begins with comments and metadata on lines starting with . The first line not starting with contains list of value names e.g. time, position, second line contains units these parameters are in. Some of these values are mandatory and some are optional. Each of these values has its own strict definition and should be the same, no matter which software was used to generate the file. Having option to define units is also important.Having clear definition of each value might not seem like something worth highlighting, but it is really important. Even something as simple as ToA for cluster can be defined in multiple ways. One of them is to take pixel which recognized signal first, someone might use the pixel which arrived last and another option would be to take average or weighted average. This approach allows for having all these values, just under different names. Main advantage of this format is it's simplicity, which is really important, because it is intended to be read from scripts written by people who have high understanding of physics, but programming is not their main specialization. Also scripts accessing these files might not be run too many times and reducing runtime by 1 minute at the cost of increasing time of writing the script by an hour is not a good tradeoff when the script is only going to be run a few times. There are a few disadvantages of this approach, one of them being that adding additional values will cause this format to become bloated. This effect will be reduced by having multiple people approve each value before adding it into specification. There are also issues associated with every formated text format, like inefficient use of disk space, slower read/write performance. To combat this issue, binary version might be created in the future for working with large amounts of data.

Chapter 3

Implementation

3.1 Input file formats

- [*.t3pa]
  - text file with one pixel per line
- [*.t3p]
  - binary version of t3pa
- [*.t3r]
  - raw detector data data
  - it's documentation is basically nonexistent
  - also contains empty packets, when detector isn't fully utilized

If speed doesn't have top priority, t3pa file is preffered. Working with it is slower in terms of CPU time, but not . Since it is human-readable, it is possible to find which part of the data causes problems and assess if the problem was with data acquisition, or with data processing. In the end, time saved on debugging can easily outweigh by time spent on runtime.

3.2 Software architecture

Software is divided into few major building blocks. First of them just reads data from file or directly from the detector and converts them into internal representation. When reading directly from detector, it would be vital to to create noisy pixel detection Second part is the clustering engine responsible for grouping pixels into clusters each corresponding with single particle. Final part takes clusters corresponding to each particle, computes values required for particle identification. Last optional step is to take computed values and

convert them to final form. This could be just writing them to file as they are, or greatly reduce its size by creating e.g. histogram.

3.3 Clustering

Algorithm 1: Overview of clustering

```
Result: Pixels are grouped into clusters that are close in both time
        and space
sort pixels by ToA;
forall Pixels do
    look at pixels position in the matrix;
    if there is existing cluster then
        Mark the cluster as complete and remove it from matrix;
    else
    end
    Create new cluster containing this pixel;
    forall Neighboring clusters do
        if Clusters are close in time then
            Merge the clusters into one;
        else
            Mark the cluster as complete and remove it from matrix;
        end
    end
end
return Clusters in the order they were created
```

Main clustering engine contains buffer of pixel events, 2D matrix and collection of clusters. Buffer of pixel events is required for multiple reasons, most important of them being that these events need to be sorted before clusters can be identified. To be absolutely sure that front of the buffer is sorted, it would have to have at least the number of pixels in the matrix (65 K), or time difference between pixel being added and the oldest pixel is longer, than the time it takes to read out full matrix i.e. for AdvaPIX TPX3 which has readout speed 40 MHits/s[16] it would be $\frac{65536}{4e7}s = 1.6ms$. This scenario wouldn't yield any reasonable results from this algorithm anyway, because it would result only in one big cluster, therefore we divide these numbers by some reasonable constant, knowing that in edge case scenario it could yield incorrect result.

My first option was to keep buffer of pixel events in binary heap for time complexity over n insertion and removals with buffer of size k of $\mathcal{O}(n \times ln(k))$. However, with real world data, sorted linked list turned out to be faster even though it has time complexity of $\mathcal{O}(n \times k)$. This is most likely because there is still some order in data coming from the detector. Also size of the buffer is in

reality not that large, because it's minimum required size is limited not only by number of pixel-events, but also by time between newest and oldest. There is interesting possibility for improvement, where pixel events would be added in batches that would be sorted by the fpga in the detector itself. Assuming size of this batch is k where k is buffer size and c is some constant. Inserting each batch would be merging of two sorted lists which has complexity of O(k + k) which simplifies to O(k). This means n pixels added in n $\times$ c batches will take O(n $\times$ c $\times$ k) time, which simplifies to (n).

Main element of this engine is the matrix. Each element of this matrix corresponds to one pixel and contains reference to last event and cluster containing it. When a pixel event is being processed it is first looked at the corresponding place in the matrix. If the space is occupied, the original cluster is removed from the matrix. Otherwise surrounding pixels are checked. If occupied pixel is found and time is lower than specified interval, current event is added to it's cluster. If two or more clusters satisfy this condition, they are joined. If There is no cluster satisfying condition, new cluster is created. Every pixel-event contains one byte variable used to mark which neighboring pixels were also activated. Doing this at this point makes it really fast, since I am already checking neighboring pixels. It also adds unnecessary code complexity, since this can be done afterwards and these features would be isolated. If the performance wasn't a big problem, i would probably do this while computing other properties of clusters. If no clusters are joined or removed time complexity is (1). If cluster is removed, time complexity gets to (n) where n is number of pixels in given cluster, because entry for each of its pixels must be cleared from the matrix. However each pixel will be removed from the matrix only once which means that the amortized complexity is only (1).

3.3.1 Memory usage

There is a 256 $\times$ 256 matrix with pointers. Assuming 2 pointers per point in matrix (one for cluster, other for pixel), assuming 8 B per pointer, that would use $256 \times 256 \times 8 \times 2 =\sim 1MB$ even before any data is added.

Modern computers have enough memory so I wouldn't have to worry about this e.g. laptop I am writing this on has 16GB. Unfortunately, this is already more, than the 0.5 SRAM is available at CPU used in MIRAM prototype.[**?**] There some structure like hash table can be used instead to save memory (unfortunately at the cost of speed, which is also at a premium)

Pixels can take only little amount of space, if needed, after all Timepix3 packet has 48 bits.[8]. However, this could grow quite easily, after all you might want to give each pixel a unique id and if you make it 64 bit integer, the original size of pixel just doubled. Other places where the size could grow include storing energy and time as 64bit floating point numbers, storing duplicated values e.g. both calculated time and original ToA, not discarding ToT after calculating calibrated energy.

3.3.2 Time complexity

3.4 Parallelization

To harness more power from modern CPU, parallel version was done. Before clustering begins, pixels are divided by some criteria between threads. I was considering splitting them by time and by position. Split by time has few advantages - it is easy to do, relatively easy to do, checking if clusters should be joined can be done brute force without big performance issues, since there won't be too many clusters close enough in time to be joined. It is required to sort pixels near the time of split to be sorted, or at least have its time compared to constant. Discarting those is also a valid option, that is trivial to implement.

Algorithm 2: Parallel clustering

```
Result: Split clustering into multiple threads
Divide pixels into two sets based by their coordinates;
Add pixels on the boundary to the second set too;
Find clusters in both sets separately;
Create 2 queues for each boundary pixel, one for each set;
forall Clusters do
    if Contains boundary pixel then
        Push the cluster to corresponding queue;
    else
        Mark the cluster as finished;
    end
end
forall Boundary-pixel queues do
    while both queues for boundary pixel contain cluster do
        Pop and merge the clusters;
        if Cluster has all boundary pixels resolved then
            Mark the Cluster as finished;
        end
    end
end
Sort clusters on the boundary by ToA;
Merge lists of clusters, keeping them sorted by ToA;
```

Split by position on the other hand is more difficult to implement, but it also has some advantages. Unlike splitting by time, which requires having large amount of data before starting, makes it more suitable for processing in real time.

While it can be easily generalized to more than 2 threads, I will describe it's workings only on 2. First pixels are divided whether they are on the top or bottom part of the matrix, keeping one pixel overlap. Since data is split

before sorting events by time, sorting step can, at least in theory, take less time. Then Clusters are created in each half independently. Each cluster now keeps track of pixels, that are touching borders to allow for later joining of clusters and unfortunately increasing memory usage.

There are 2 queues for each pixel on the dividing line - one for each side. These queues contain reference to cluster containing given pixel. These are sorted by time, so on the same position in both queues are clusters containing the same pixel, thus they need to be joined. If resulting cluster doesn't contain any cluster on the border, it is put into buffer, which will have to be sorted.

As I stated, this approach can easily be used for more than 2 threads, but after looking at performance compared to sequential I don't see the point in doing so.

Finding clusters isn't the only part where doing thing in parallel can help. There is no reason to read to load data in the same thread used for processing. Calculating most cluster properties can also be done in parallel, since most properties don't require to look at other clusters.

3.5 Minimal version

Even though approach above does provide good results, it needs reasonably powerful computer. When the data rate is low, there is also an option to look for packets that don't correspond to packets being hit and call packets that consist of pixels being hit surrounded by dummy packets a cluster.

This might skew data, if some situations cause cluster to be discarded more often, than others, However it does provide some aditional information, compared to just taking sum of energy, which is one of the modes MIRAM uses [1]

Chapter 4

Functional Verification

4.1 Benchmark data

To measure performance I planed on using 2 main sets of data. one of them is background radiation from the environment and the other one with high amount of gamma radiation. It is necessary to have both of these, because there is suspicion, that performance wouldn't depend only on number of pixels, but also on number of clusters. Number of hits per second could also play a significant role since it can influence how much out of order pixels will be. This is because if there is enough time between events to read it, these events can't be mixed together

Background radiation data was measured with detector simply lying on

Figure 4.1: ADVAPIX TPX3 with Am241 source from smoke detector

the table.

To get data with a high amount of gamma radiation I used Americium241 source from a smoke detector. As a detector I used ADVAPIX TPX3 with 1mm thick Si sensor, because that's what was available to me at the time. I simply placed the source onto kapton tape covering the sensor of the detector and ran the measurement.

4.1.1 Comparison with PIXet plugins

4.1.2 Further possibilities

Conclusion

This work isn't finished. I would like to try again with a new assignment.

Bibliography

[1] ESA. MIRAM - Miniaturised Radiation Monitor. 2021. Available from: https://artes.esa.int/projects/miram

[2] Jakubek, J. Precise energy calibration of pixel detector working in time-over-threshold mode. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, volume 633, 2011: pp. S262–S266, ISSN 0168-9002, doi:https://doi.org/10.1016/j.nima.2010.06.183, 11th International Workshop on Radiation Imaging Detectors (IWORID). Available from: https://www.sciencedirect.com/science/article/pii/S0168900210013732

[3] Visser, J.; Van Beuzekom, M.; et al. SPIDR: a read-out system for Medipix3 & Timepix3. *Journal of Instrumentation*, volume 10, no. 12, 2015: p. C12028.

[4] Burian, P.; Broulím, P.; et al. Katherine: Ethernet embedded readout interface for timepix3. *Journal of Instrumentation*, volume 12, no. 11, 2017: p. C11001.

[5] Khalil, M.; Dreier, E. S.; et al. Subpixel resolution in CdTe Timepix3 pixel detectors. *Journal of synchrotron radiation*, volume 25, no. 6, 2018: pp. 1650–1657.

[6] NASA. NASA, CERN Timepix Technology Advances Miniaturized Radiation Detection. 2019. Available from: https://www.nasa.gov/feature/nasa-cern-timepix-technology-advances-miniaturized-radiation-detection

[7] Technical Proposal IEAP.CTU/17/ESA/MIRAM.01. Technical report.

[8] Poikela, T.; Plosila, J.; et al. Timepix3: a 65K channel hybrid pixel readout chip with simultaneous ToA/ToT and sparse readout. *Journal of Instrumentation*, volume 9, no. 05, may 2014: pp. C05013–C05013,

doi:10.1088/1748-0221/9/05/c05013. Available from: https://doi.org/10.1088/1748-0221/9/05/c05013

[9] Wong, W.; Alozy, J.; et al. Introducing Timepix2, a frame-based pixel detector readout ASIC measuring energy deposition and arrival time. *Radiation Measurements*, volume 131, 2020: p. 106230, ISSN 1350-4487, doi: https://doi.org/10.1016/j.radmeas.2019.106230. Available from: https://www.sciencedirect.com/science/article/pii/S1350448719305165

[10] Ballabriga, R.; Alozy, J.; et al. Review of hybrid pixel detector readout ASICs for spectroscopic X-ray imaging. *Journal of Instrumentation*, volume 11, no. 01, jan 2016: pp. P01007–P01007, doi:10.1088/1748-0221/11/01/p01007. Available from: https://doi.org/10.1088/1748-0221/11/01/p01007

[11] Dudák, J. *ENERGY SENSITIVE X-RAY RADIOGRAPHY AND TOMOGRAPHY OPTIMIZED FOR SMALL ANIMAL IMAGING*. Dissertation thesis, Czech Technical University in Prague, 2019.

[12] Llopart, X.; Ballabriga, R.; et al. Timepix, a 65k programmable pixel readout chip for arrival time, energy and/or photon counting measurements. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, volume 581, no. 1, 2007: pp. 485–494, ISSN 0168-9002, doi:https://doi.org/10.1016/j.nima.2007.08.079, vCI 2007. Available from: https://www.sciencedirect.com/science/article/pii/S0168900207017020

[13] Llopart, X. *TIMEPIX Manual v1.0*. CERN, 2006.

[14] Llopart, X.; Poikela, T. *Timepix3 Manual v2.0*. CERN, 2015.

[15] ADVACAM. *MINIPIX TPX3 datasheet*. Available from: https://mk0advacamcom3oi5gxh.kinstacdn.com/wp-content/uploads/2019/04/MNXT3S-Xxx190925-MiniPIX-TPX3-Standard-Datasheet-2020-04-03.pdf

[16] ADVACAM. *ADVAPIX TPX3 datasheet*. Available from: https://mk0advacamcom3oi5gxh.kinstacdn.com/wp-content/uploads/2017/08/APXT3M-Xxx200128-AdvaPIX-TPX3-Datasheet-2020-05-26.pdf

[17] ADVACAM. *PIXet Pro user's manual*. Available from: https://mk0advacamcom3oi5gxh.kinstacdn.com/wp-content/uploads/2017/08/APXT3M-Xxx200128-AdvaPIX-TPX3-Datasheet-2020-05-26.pdf

Appendix A

Acronyms

PCD Photon counting detector

GbE Gigabit Ethernet

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format