Calibration of Timepix4

Arda Ünal, Boğaziçi University, Turkey

Supervisors: Jonathan Correa, David Pennicard and Alexandr Ignatenko

DESY Summer Student Program

September 6, 2022

Abstract

Research with photons provides insights into fundamental material properties and shows how they can be controlled. Researchers examine materials and biomolecules with extremely powerful and short pulse x-rays. Therefore, detector technology is continually developing to meet the needs of these researchers. Timepix4 is a versatile photon detector readout chip with 55 µm pixels, recently developed by CERN[1] on behalf of the Medipix collaboration[2]. It can operate both in a photon counting mode with frame readout, and a timestamping mode with event readout. Both of these modes offer higher performance than existing Medipix3[3] and Timepix3[4] chips. Control software of Timepix4 is being developed at DESY[5] by the Detector Science group. The aim of this report giving a brief overview of calibration studies of Timepix4 which consists of testing chip under different conditions, developing visualization tools, and debugging control software.

Contents

1 Introduction

1.1 DESY

DESY[2] is one of the world's leading accelerator centers. With the DESY large-scale devices, researchers are exploring the microcosm in all its diversity – from the interplay of the smallest elementary particles to the behavior of novel nanomaterials to those vital processes that take place between biomolecules. The accelerators as well as the detection instruments that DESY develops and builds are unique tools for research: They generate the strongest x-ray light in the world, bring particles to record energies and open completely new windows into the universe.

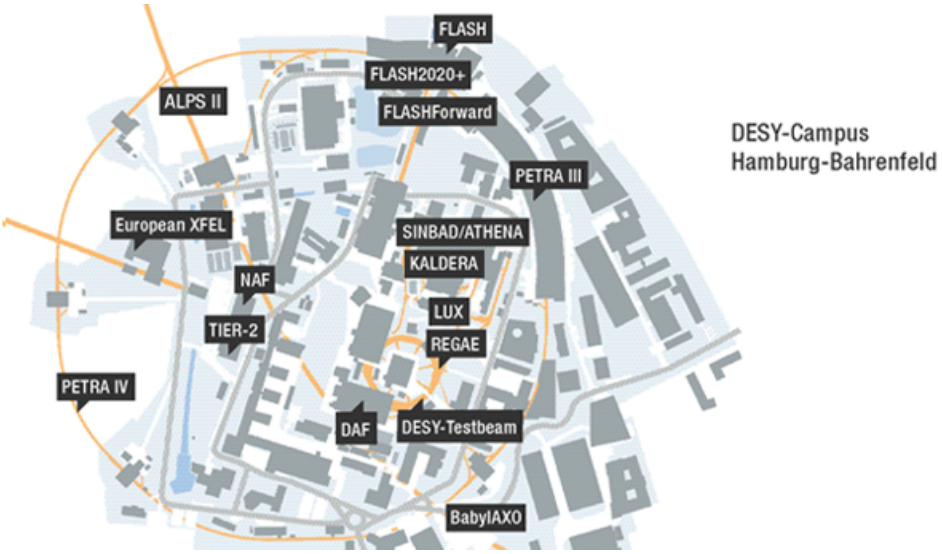

Figure 1: FLASH facilities stand for free-electron laser sources and PETRA facilities are storage rings which can be used as synchrotron radiation sources.

1.1.1 Accelerators

Accelerators are among the most important tools in research. They bring electrons, protons, or high atomic number atoms to relativistic speeds – up to almost the speed of light by using RF cavities and powerful magnets. A wide variety of research disciplines benefit from the fast particles: particle physicists have them collide in order to observe subatomic particles(such as Higgs boson). Chemists, materials scientists, and biologists use accelerators to produce x-ray light to take a close look at an extensive variety of materials – from aircraft turbines to microchip semiconductors to vital proteins. There are numerous different types of accelerators such as linear accelerators, synchrotrons, rhodotrons, and so on. Accelerators can be operated as photon sources using 2 different methods: synchrotron radiation and free-electron laser

1.1.2 Photon Science

Conventional x-ray tubes like those used in hospitals are generally not suitable for today's scientific applications, because the x-rays they generate are too weak for the experiments. Particle accelerators, on the other hand, generate extremely powerful and focused radiation. These x-ray beams are so intense that they can reveal even the finest details – for instance, the tiniest cracks and pores in a turbine blade, minute impurities in a semiconductor, or the positions of individual atoms in a protein molecule. Moreover, when researchers fire extremely short x-ray flashes at various samples, they are able to observe ultrafast processes such as those that occur in a chemical reaction. There are different methods for observation of materials and molecules: x-ray crystallography, x-ray diffraction and so on.

1.1.3 Synchrotron Radiation

Relativistic charged particles emitted radiation when they are accelerated perpendicular to their velocity. Synchrotron radiation is the electromagnetic radiation emitted when charged particles travel in curved paths(synchrotron). Because in most accelerators the particle trajectories are bent by magnetic fields, synchrotron radiation is also called Magneto-Bremsstrahlung radiation. The emitted spectrum is broadband from the microwave (harmonics of the driving RF field) to x-ray spectral regions.

1.1.4 Free-Electron Laser(FEL)

The special x-ray laser light is generated in a FEL according to a sophisticated principle: During the slalom course through a periodic magnetic arrangement (undulator), the electron bunches emit light (photons) of a fixed wavelength. The photon beam propagates in a straight line and overlaps with the electron bunch. It imprints the regular structure of the electrons, which means that after some time, the initially uniform charge density distribution has become a series of individual charge slices, each separated by a wavelength of light. Now all electron discs radiate in unison – the light can intensify into intense laser radiation.

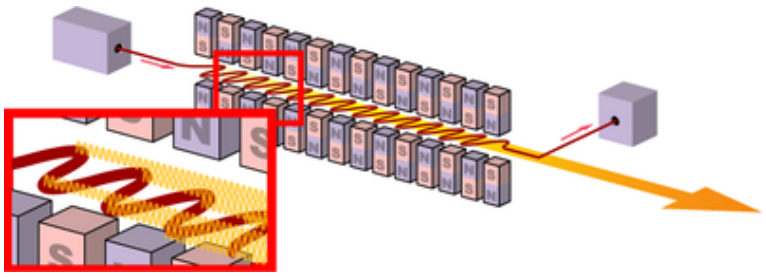

Figure 2: Electron motion in an undulator which consists of series of permanent magnets.

1.1.5 Detectors

Photon-detectors are crucial devices for many applications in photon science. In x-ray detectors, the energy transported by the radiation is converted into forms that can be recognized electronically. Generally, the photons are absorbed by the detector material and energy transfer takes place by ionization. Typically, 3 types of x-ray detectors are used: gas detectors, scintillation counters, and solid state detectors.

2 Timepix4

Timepix4 is a 24.7 × 30.0 mm2 hybrid pixel detector readout ASIC that has been designed to permit detector tiling on 4 sides. It consists of 448 × 512 pixels which can be bump bonded to a sensor with square pixels at a pitch of 55 m. Like its predecessor, Timepix3, it can operate in data-driven mode sending out information (Time of Arrival, ToA, and Time over Threshold, ToT) only when a pixel has a hit above a pre-defined and programmable threshold. In this mode, hits can be tagged to a time bin of fewer than 200 ps and Timepix4 can record hits correctly at incoming rates of 3.6 MHz/mm2/s. In photon counting (or frame-based) mode it can count incoming hits at rates of up to 5 GHz/mm2/s. In both modes, data is output via between 2 and 16 serializers each running at a programmable data bandwidth of between 40 Mbps and 10 Gbps.[6]

Edgeless technology was used in the design of Timepix4 in Figure 3. TSVs are high-performance interconnect techniques used as an alternative to wire-bond. This technology makes it possible to combine Timepix4 chips without having blind areas. Although it has edgeless property, it is still possible to use wire-bounds. In the current system that we're testing, wire-bounds are used because it is technically much simpler and quicker.

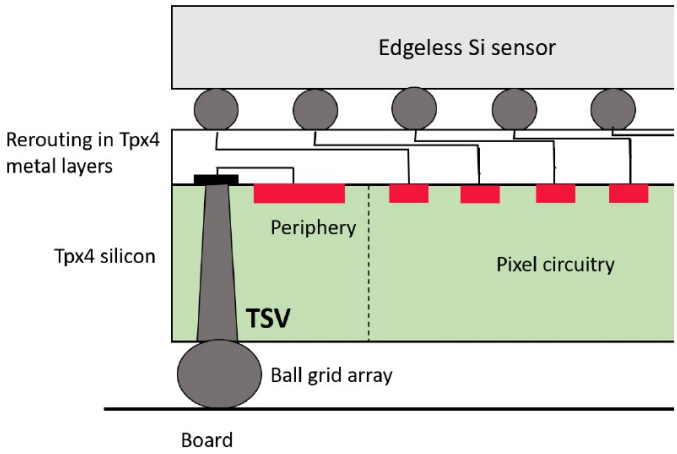

Figure 3: TSVs are high-performance interconnect techniques used as an alternative to wire-bond to create 3D packages.

2.1 Analog Front-end of Timepix4

The analog front-end of the Timepix4 chip consists of a charge-sensitive amplifier with a programmable feedback capacitance. The pixel gain can be adjusted according to the detection requirements. In the high gain mode, a 3 fF metal-to-metal capacitance is used. An additional capacitance of 3 fF can be added in parallel to the main capacitance to decrease the gain of the amplifier by enabling the low gain mode bit. A MOS gate capacitance can also be enabled in the signal path to program the chip in the adaptive gain mode. The DAC(digital analog converter) can be used change threshold value of the signal. Therefore it can be used to eliminate low power signals at the end of the charge-sensitive amplifier. In Figure 4, block diagram of the Timepix4 front-end circuitry is shown.[6]

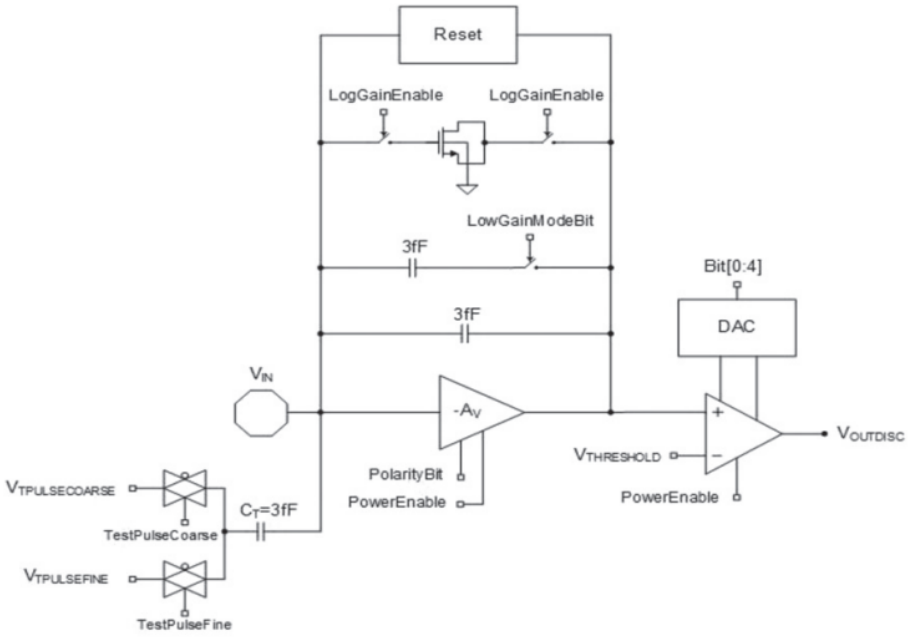

Figure 4: Silicon sensor signal acquired via Vin input pad. According to given amplifier mode, signal is amplified. There is a DAC at the end of the amplifier which can be used as suppressor for unwanted signals. Additionally, there is test pulse functionality which can be used to test configuration of chip.

2.2 Timepix4 Pixel Configuration

In Figure 5, each pixel matrix is composed of 224 $\times$ 256 double columns. The pixels in each double column are grouped into superpixels containing 2 $\times$ 4 pixels. The ends of a column at the top and bottom of the ASIC are split into 2 groups, left and right, called data fabrics.[6]

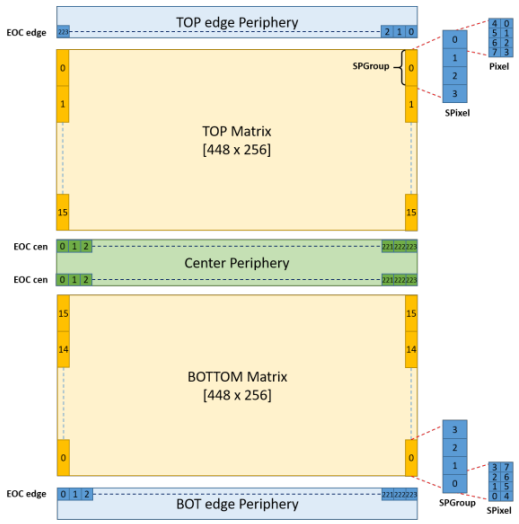

Figure 5: Timepix4 pixel matrix organization. Top and bottom peripheries are used for data acquisition while the center periphery is used to configure pixels.

2.3 Readout Modes of Timepix4

Timepix4 can be used in 2 different modes:

- Frame Based Mode(Photon Counting Mode): In the photon counting mode, for each incoming hit above the threshold, a pixel counter is incremented. Each pixel contains 2 counters which have a programmable depth of either 8 or 16 bits. As there are 2 counters per pixel continuous read/write operation is used. One counter is being read out while the other is counting. In Figure 6(a), there is an illustration of this mode.

- Data Driven Mode(Event Mode): In the event mode, every hit above the threshold of pixel will create a data packet which consists of coordinates of the pixel, hit energy, and hit time as in Figure 6(b).

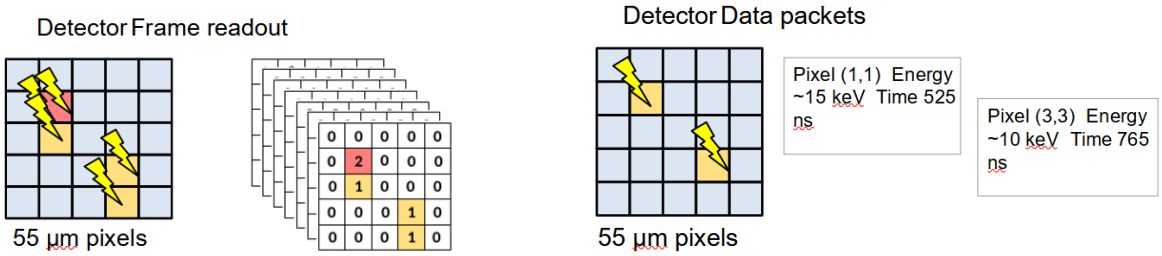

Figure 6: Frame and event mode principles.

2.4 Laboratory Setup

Timepix4 is a high-speed ASIC chip, therefore it is necessary to use FPGA for acquisition. The Xilinx Zynq Ultrascale FPGA development board is used for test purposes. For communication and powering up the Timepix4 chip, a PCB was designed(see Figure 7) which is directly connected to the Xilinx Zynq FPGA board via firefly optical connectors. Xilinx Zynq FPGA board has a CPU which runs a Linux OS. By using a computer, one is able to connect this Linux OS via SSH and control the firmware of the Timepix4 chip by using some high-level programming language such as Python.

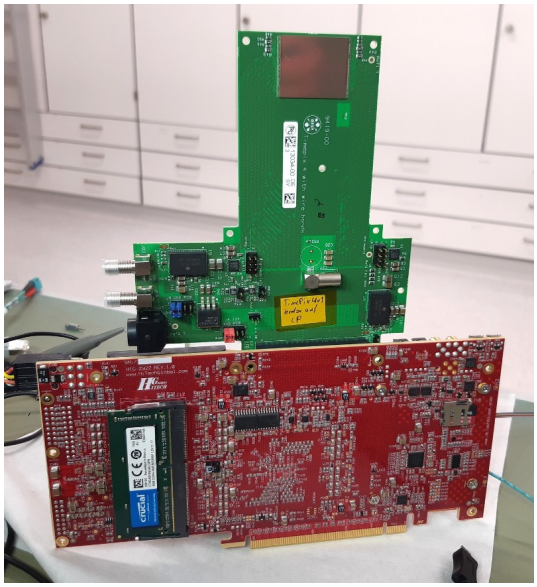

Figure 7: Laboratory setup, green board is PCB of Timepix4 sensor which is connected to Xilinx Zyqn FPGA board(red).

3 Measurement

The control software of Timepix4 is currently being developed, so my work focused on testing functionality for taking images and electronically masking pixels to test whether the chip and software behave as expected and to find any bugs. In the current system, that we're testing, we only have a bare chip, rather than one with a sensor. However, if the threshold in the pixel is close to the baseline level, then we will still measure hits due to noise causing changes in the pixel voltage. The baseline level varies from pixel to pixel, so with a given threshold setting, different pixels will be noisy. It is possible to individually configure each pixel to adjust their baseline level, and this needs to be done to get the chip performing well. For now, we have focused on testing the function to mask pixels, to make sure that our software for configuring the pixels works.

3.1 24-Bit Raw Data Analysis

I started to analyze data with 24-bit readout mode because there are some bugs with 16-bit and 8-bit readout modes. I wrote a script which creates a detector pixel map matrix after processing raw data and visualizing it as an image. While analyzing the data, I found some unknown data packets and some conflicts in the decoding code. Images reveal some anomalies. Most of the pixels are cumulated at the corners of the detector and most of the pixels have no packages as indicated in Figure 3.1. Therefore, I changed the script in such a way that one can see pixels which have 0 count value in the different images. The results demonstrated that 0 count pixels are also cumulated at the corners of the image as given in Figure 3.1. We expected to see data from the whole chip rather than just specific regions. These works showed many of the pixels did not read out.

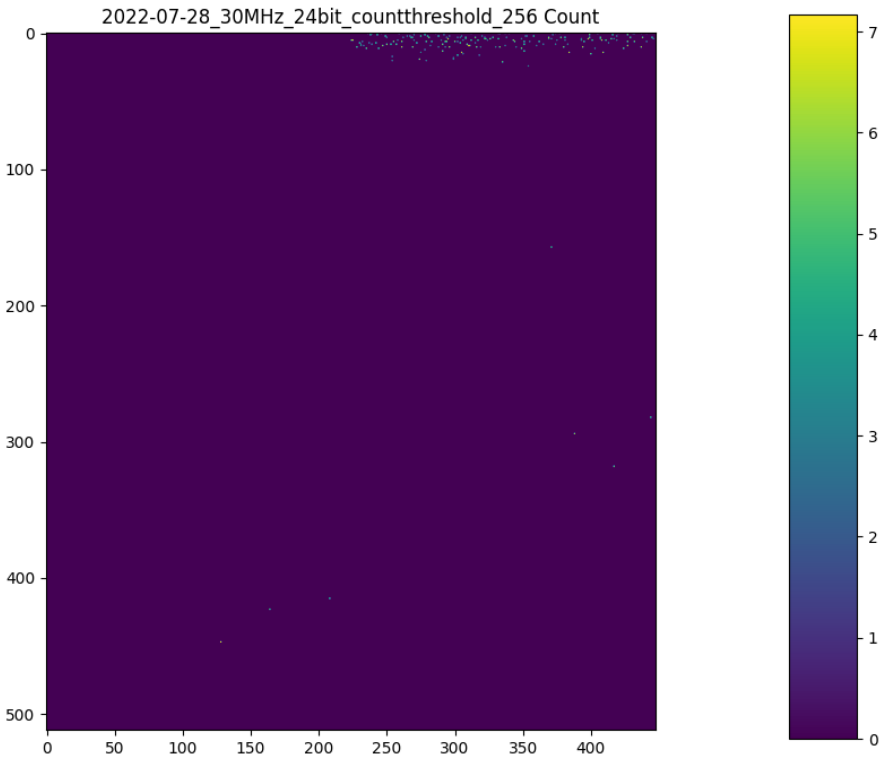

Figure 8: 24-bit count value image taken in nominal conditions.

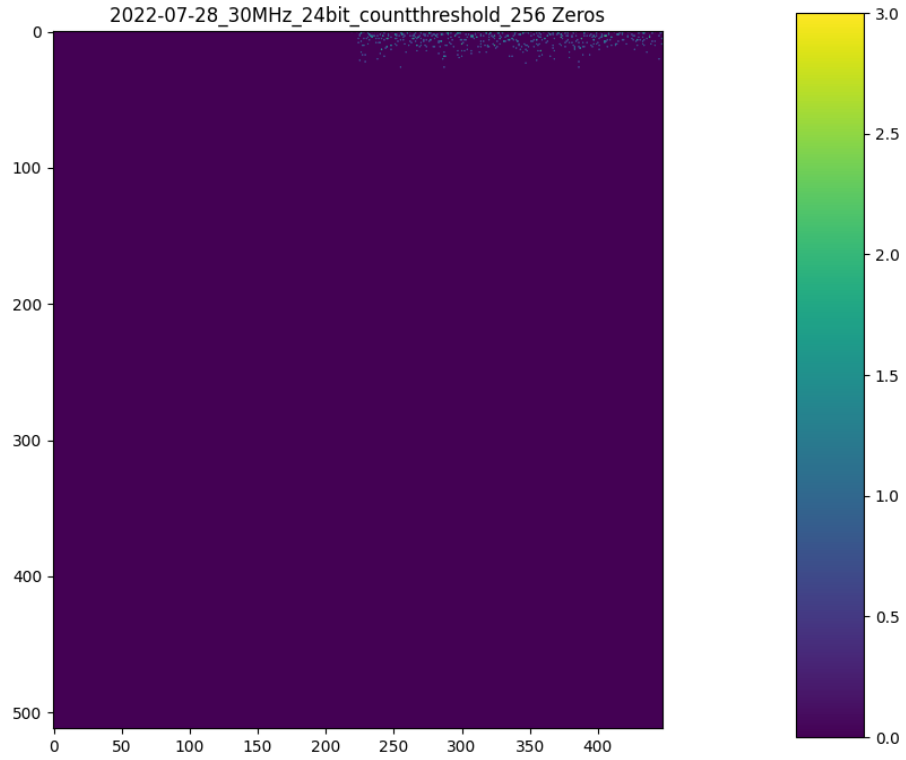

Figure 9: 24-bit zeros image taken in nominal conditions.

3.2 Pixel Noise Analysis

After 24-bit raw data analysis, a new data set was obtained which includes 100 raw data files with different pixel threshold values. A threshold value of a pixel basically specifies the incoming signal energy threshold. If the signal is higher than the threshold value, then it is counted. First I wrote a script which read these files and convert them into a 3D matrix. Then it creates count value plots for each pixel to visualize the evolution of pixel count value according to threshold value as in Figure 10. This program was unuseful because there are 512x448 pixels in the detector and analyzing one by one is basically pointless. Therefore, instead of analyzing individual pixels, I created a histogram-like bar chart that shows how many pixels have count values for the corresponding threshold value as in Figure 10. The aim of this experiment was to understand what are the nominal threshold values of the pixels. This knowledge can be used while tunning the chip for the experiments.

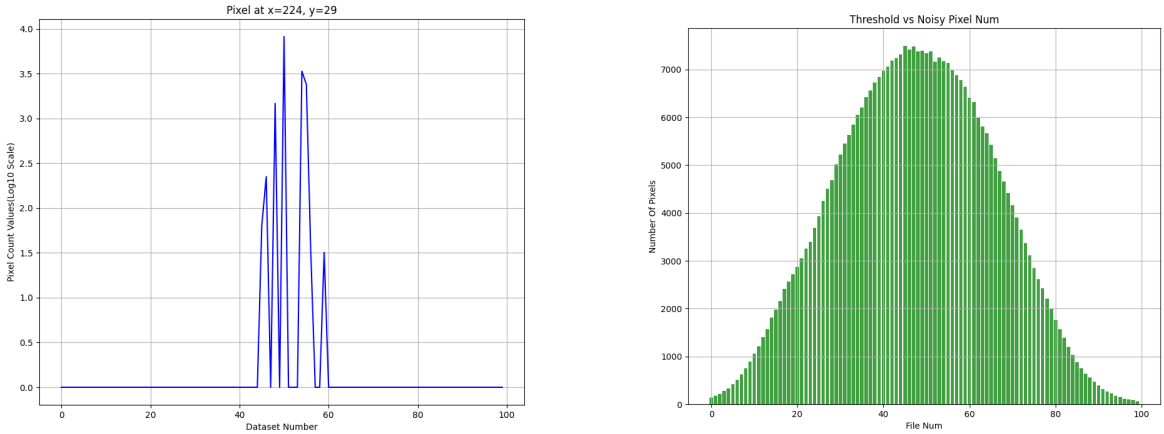

Figure 10: (a) Indicates pixel count values versus threshold values.(b) Histogram of number of noisy pixels versus threshold value.

3.3 16-bit Frame and Event Mode Raw Data Analysis

Analysis of the 24-bit data did not give us a clue to find the source of the problem, so I started to work on 16-bit event readout mode. I wrote another script to visualize detector pixels if there is an event and I found only a box of pixels send data as illustrated in Figure 11. When we increased the duration of acquisition time square at the corner of the detector expanded as depicted in Figure 12, but there were still no data at the top side of the detector.Thus I started to analyze 16-bit Frame readout mode. Firstly, I took a raw data and try to decode it, but I could not do that because there is a known bug in the chip. Packages does not include frame-end information so that visualization of the images become unworkable. To analyze this issue, I changed the frame decoding script in such a way that images were separated into top and bottom parts. After that top side readout problem was solved, I started to work on 16-bit frame mode. While I was working with the 16-bit frame readout mode, I realized that there is no function for 16-bit frame readout mode in the control script of the detector. Hence, I added 16-bit frame readout acquisition functionality to the control code. With using the new function, after I learned how to control the Xilinx Zynq FPGA board, I acquired data using the computer at the laboratory and I visualized the data, but there were no big differences.

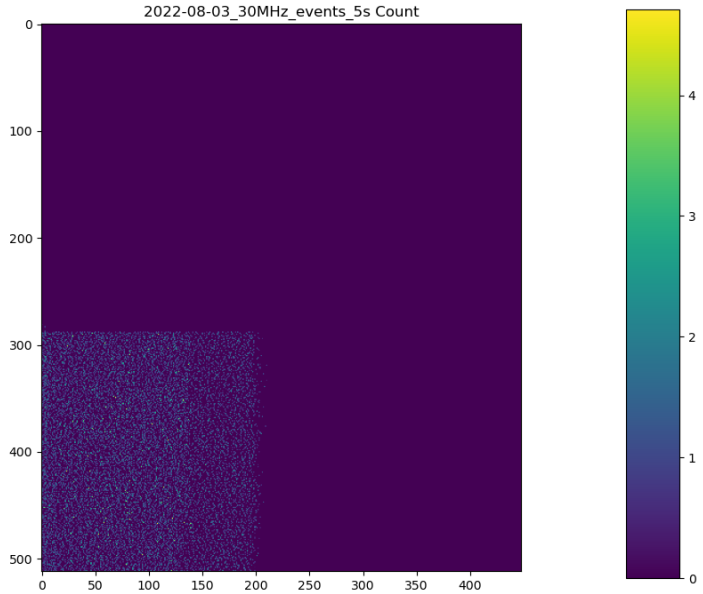

Figure 11: 5 seconds readout duration in nominal conditions.

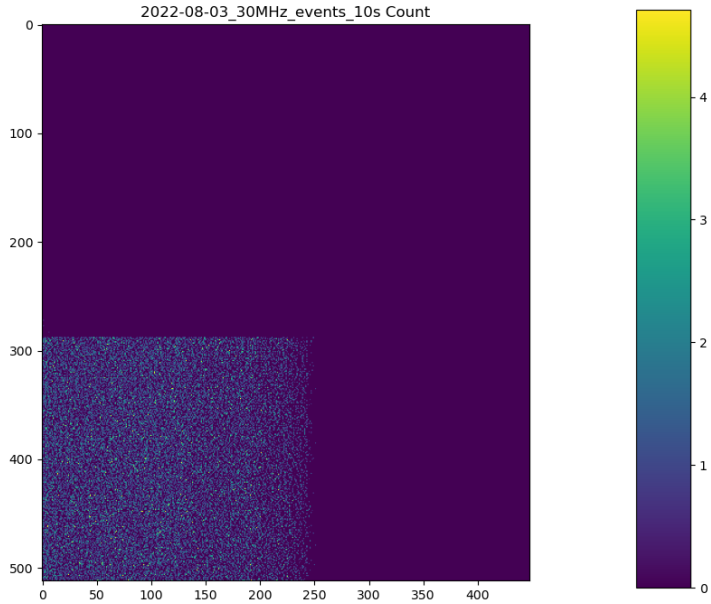

Figure 12: 10 seconds readout duration in nominal conditions.

3.4 Pixel Masking

The problem with the top part of the chip acquisition was solved, yet noisy pixels were still cumulated at the corner of the detector as given in Figure 13. One basic explanation of this issue comes from pixel matrix organization. For the top part of the chip, the pixels coordinate starts from the right top, on the other hand, the bottom part coordinates start from the left bottom side of the chip. As illustrated in Figure 13, we are able to read almost half of the chip. We noticed that by changing pixel threshold values, our readout area increases.

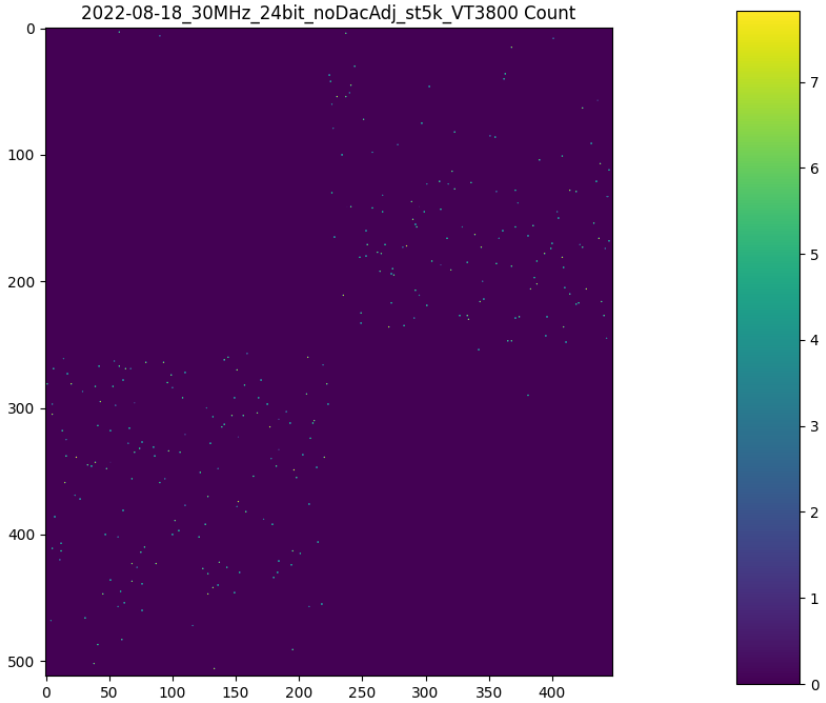

Figure 13: 24-bit frame mode noisy pixel distribution image with 5 seconds readout duration.

Therefore, we decided to mask pixels that send more than 2 packets to see all the pixels because we thought buffers are insufficient to acquire all packets for every pixel which can be done by masking those noisy pixels. I start masking with large areas such as masking the bottom and top parts of the chip. Then I systematically decreased the masked area but as I decreased we saw some strange behaviors. For instance; when I masked the area given in the Figure 14, masked area in acquired image is given in Figure 15.

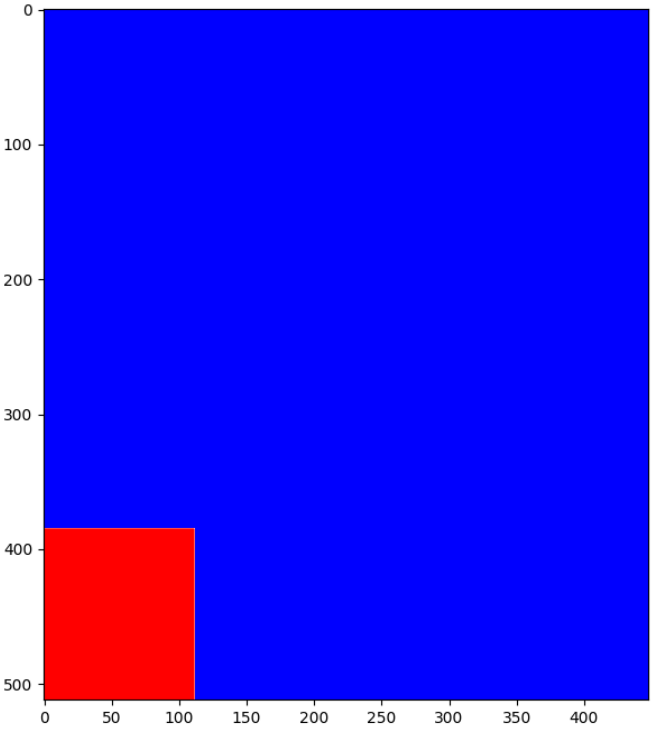

Figure 14: The masked area is red colored.

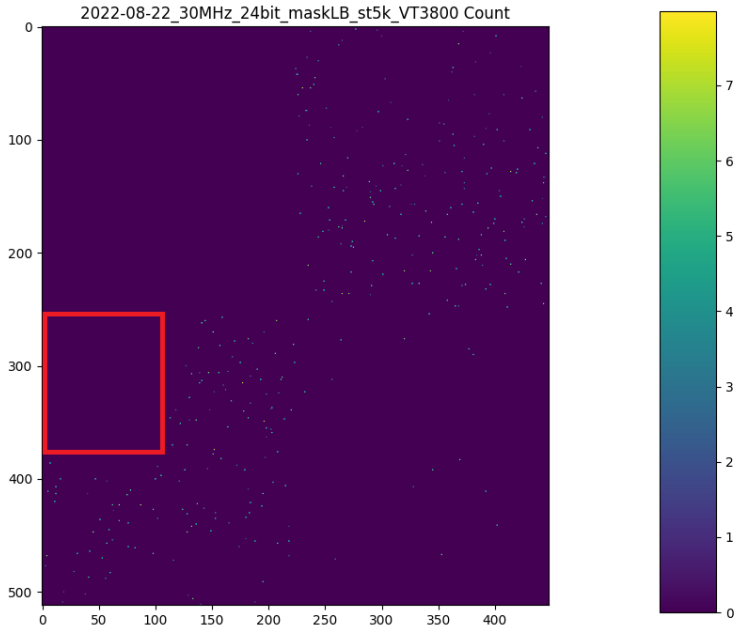

Figure 15: Red rectangle represents masked area in the acquired data.

After that, we decided to mask only noisy pixels. Before masking only noisy pixels, I created a histogram which illustrates the number of packets which is sent by pixels(x-axis) versus the number of pixels that sends that amount of data (y-axis) in Figure 16. Next, I wrote a script that finds these noisy pixel coordinates and creates a mask matrix. Using this mask matrix, I acquired a frame, but there were no differences. Whence I wrote a script which validates if the acquired data is masked properly. The results revealed masking process does not work accurately. After the performed experiments, it was found that the pixels are addressed differently when data is read out, compare to configuring pixels which was not documented in the chip manual.

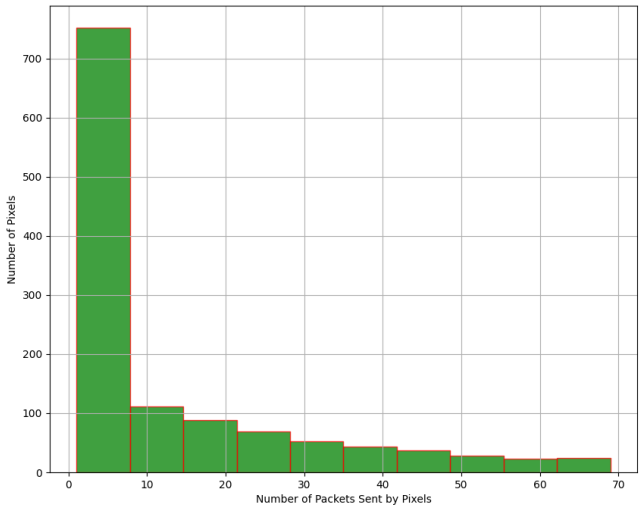

Figure 16: Distribution of number of readout packets.

3.5 Results

In this prototype system, we read out data from the chip at a very low rate, because high-speed readout is not yet implemented. As a result, hits occur in the chip more quickly than we read them out. These experiments indicate that pixels closer to the edge of the chip are given higher priority during readout, and so pixels further away might not get read out, depending on the bit rate. The solution of the problem is waiting enough time to readout all the pixels as shown in the Figure 17. I repeated the 24-bit frame mode masking noisy pixels which send more than 2 packets of data experiment with the new control script. Masked pixels can be seen in Figure 18. It successfully masked pixels according to given mask matrix as illustrated in Figure 19.

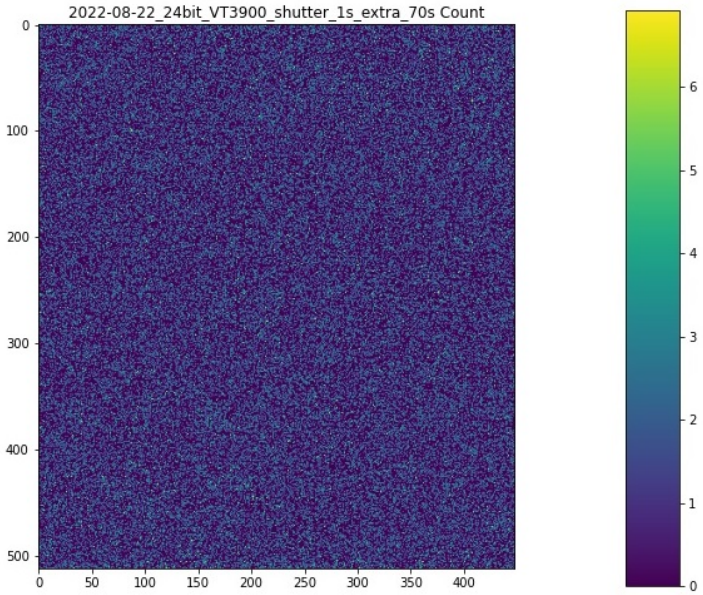

Figure 17: 24-bit frame mode image with 70 seconds extra readout duration in nominal conditions.

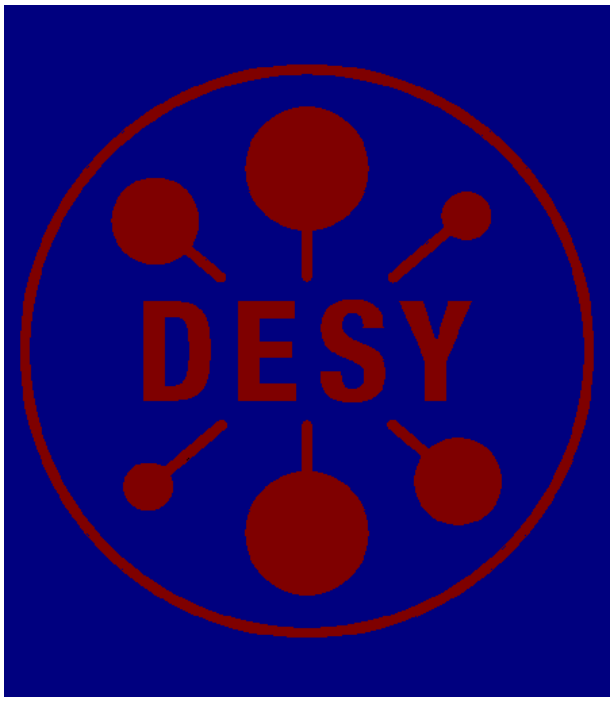

Figure 18: Masked pixels are colored red.

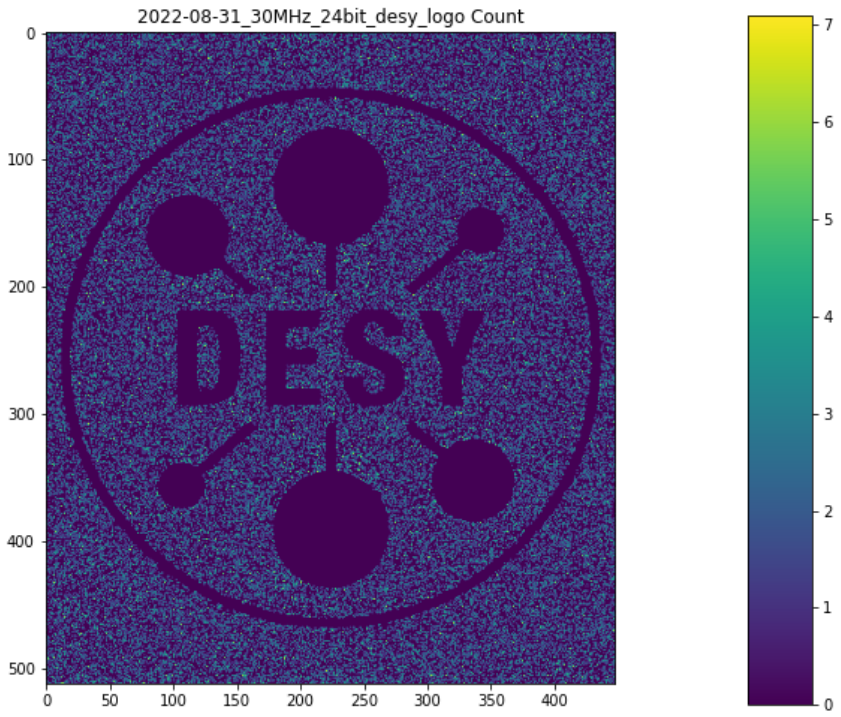

Figure 19: Masked pixels image with extra time 100 seconds in nominal conditions.

3.6 Graphical User Interface

I designed a GUI program that has functionality that both event and frame mode images can be visualized. The event data includes the energy level of the incoming particle, the timestamp of the particle when it hits the detector, and the coordinates of the pixel which is hit. The algorithm behind the showing images in event mode is that after processing raw data, it sorts events according to their timestamps behind and creates images for every unique timestamp. The GUI has a slider on the left side which can be used the visualize images according to the timestamps. On the right side, there is a video label that makes a video of this list of events (see Figure 20). Additionally, I also added an option combobox which can be used to see only the count values of pixels during the acquisition time and it will ignore the timestamps. On The second tab of the GUI has frame mode visualization which can be used to illustrate frames from multiple files. It is possible the change mode of files with using another combobox in the second tab. As the images progress, it prints file names on the list and visualizes them. After uploading process finishes, if one left click 2 times to the file names, it is possible to see image at the right side which belongs to this file. There is a save button that can be used to save all images in a sub folder. I also added a progress bar to illustrate the situation of the program (see Figure 21).

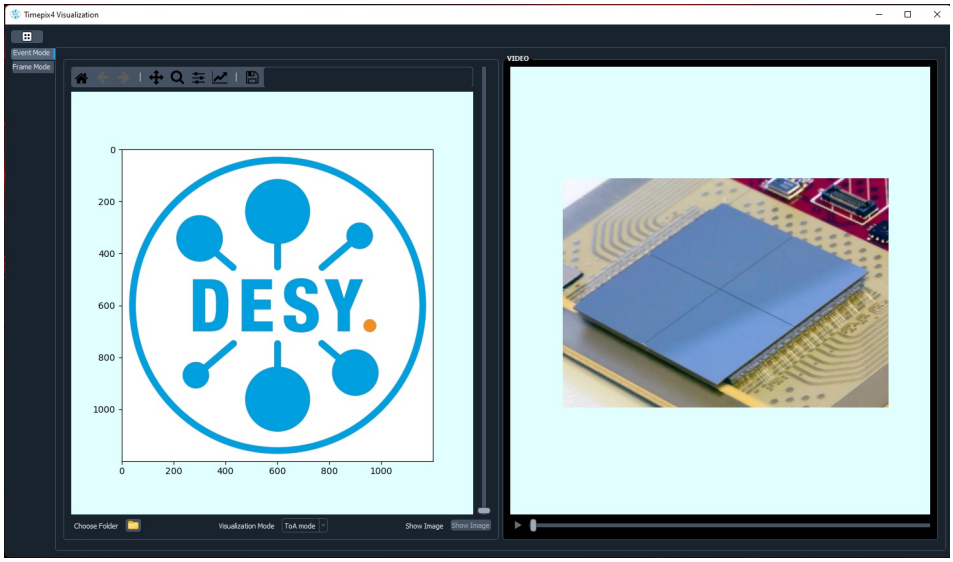

Figure 20: Computer application for visualization of Timepix4 raw event data.

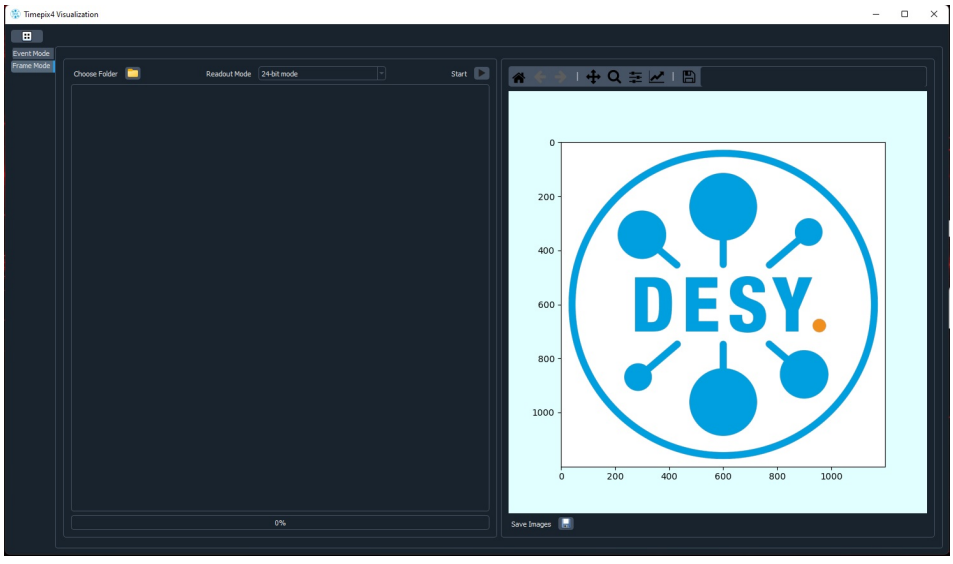

Figure 21: Computer application for visualization of Timepix4 raw frame data.

4 Conclusion

Analysis and visualization of Timepix4 data showed that currently known bugs in the chip make it inappropriate to work with 8/16-bit data acquisition modes. In addition to that, using these tools, some problems in the data acquisition and configuration software of the Timepix4 were solved. Through the Timepix4 calibration project, I developed various tools for both visualization and analysis of raw data as well as tools for control software of Timepix4 firmware which will be useful in future tests on the chip and in scientific experiments. A brief list of the designed tools are:

- 24-bit Raw Data Analysis Software : It is a Python script that can be used to visualize multiple raw data files.
- 8/16-bit Raw Data Analysis Software : I designed 2 different software both visualization of the event and frame mode. I also change the frame decoding code so that images are separated as top and bottom parts of the chip.
- Pixel Noise Variation Software: It can be used as a visualization of acquired package distribution according to the threshold configuration of the chip.
- Pixel Masking Software: It consists of 2 different scripts. One can be used to visualize noisy pixels as a histogram and the other can be used to create a mask matrix as a numpy file to mask noisy pixels.
- GUI: It combines 24-bit frame readout mode with 8/16-bit event mode in an application. The GUI makes it possible to visualize event mode with time stamps as an interactive plot and video. Everything is fully automated when there is more than 1 raw data file.

5 Acknowledgements

I would first like to thank my supervisors. A lot of thanks to Dr. Jonathan Corea, for excellent project leadership, invaluable advice, and support both technical and personal. Many thanks to Dr. David Pennicard for helping me from start to end with his golden technical and theoretical knowledge. I learned a lot of things while working with him and he never left me without an answer when I asked one. Many thanks to Dr. Alexandr Ignatenko for helping me with his datasets and teaching me how to use Timepix4. He introduced me laboratory and he always helped me with technical problems. I would like to send my thanks to all FS-DS group for their warm welcome and let us be part of the group easily. I would like to thank the lecturers who gave a lot of effort to introduce us to various trendy scientific experiments both theoretical and experimental. Lastly, I would like to send my thanks to the Summer student organizers for this excellent and fulfilling program.

References

[1] CERN, URL: https://home.cern/

[2] CERN Medipix Collaboration, URL: https://medipix.web.cern.ch/home

[3] T. Poikela et al, JINST 9 C05013 (2014)

[4] X.Llopart et al, A 633 S15–S18 (2011)

[5] DESY, URL: https://www.desy.de/

[6] X.Llopart et al, JINST 17 C01044 (2022)