Implementing an Asynchronous Port Driver for Data Analysis and Acquisition

Tanisha Aggarwal, Lukas Ho-Koziol, Patrick Nisperos

SLAC NATIONAL ACCELERATOR LABORATORY

U.S. DEPARTMENT OF ENERGY

Stanford University

Introduction

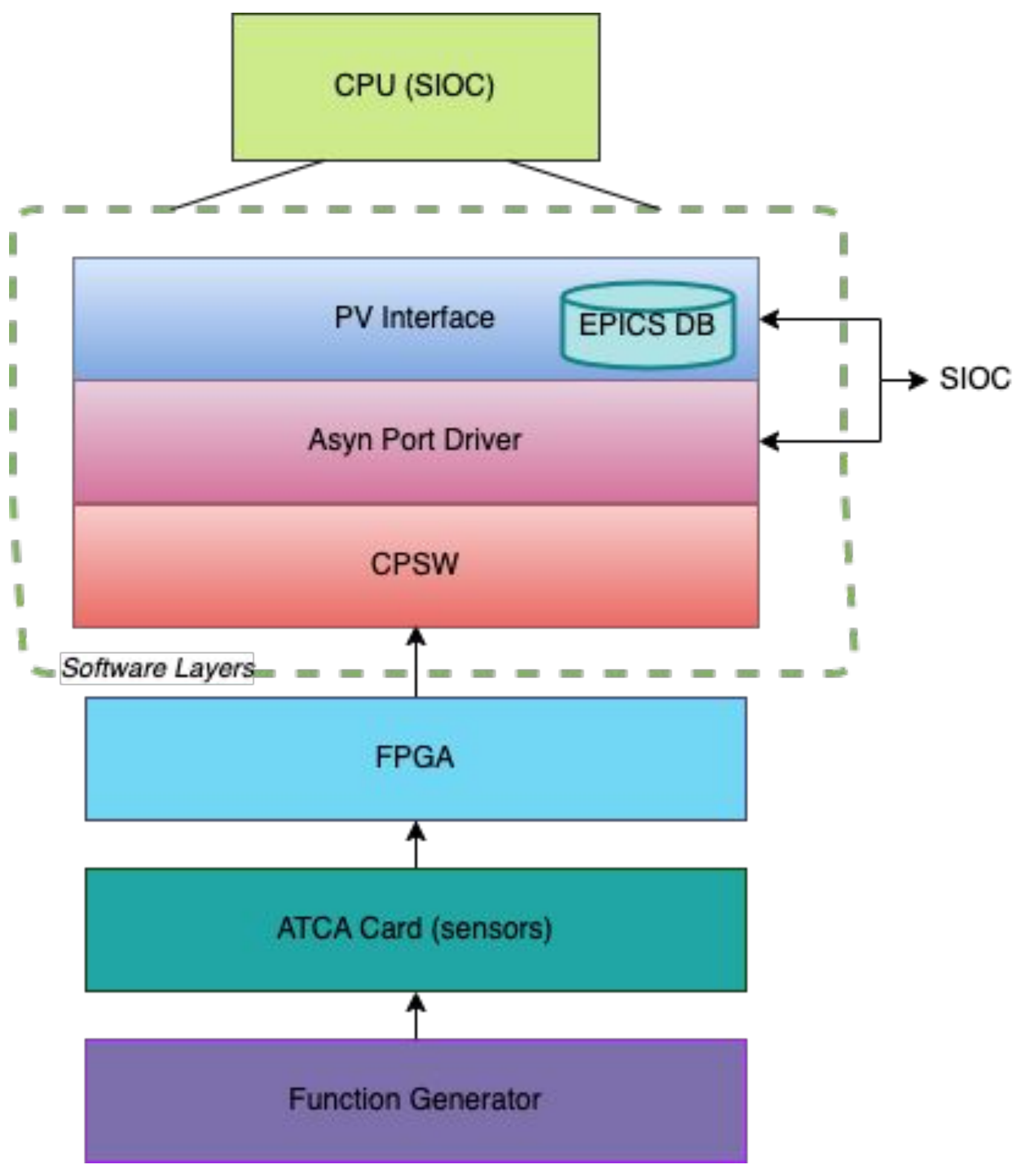
- An asynchronous port driver reads raw data from hardware sources and stores it into Experimental Physics and Industrial Control System (EPICS) records.
- This allows accelerator operators, physicists, and engineers to streamline the process of accessing and manipulating data from various sources using an Input/Output Controller (IOC).

Goals

- Facilitate communication between Advanced Telecommunications Computing Architecture (ATCA) cards and the IOC.
- Implement data analysis techniques like a fourier transform.

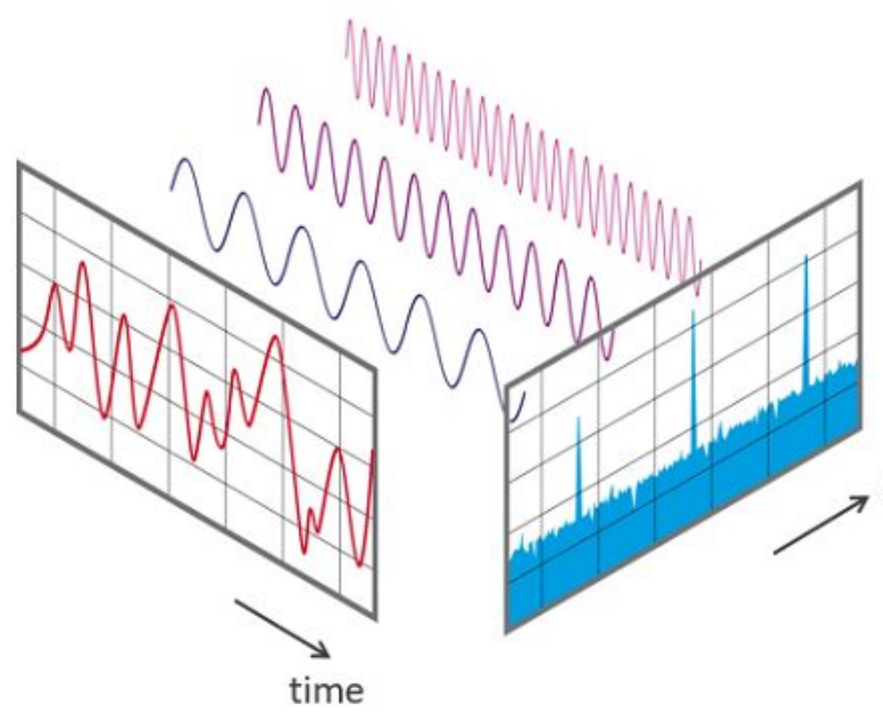
Process

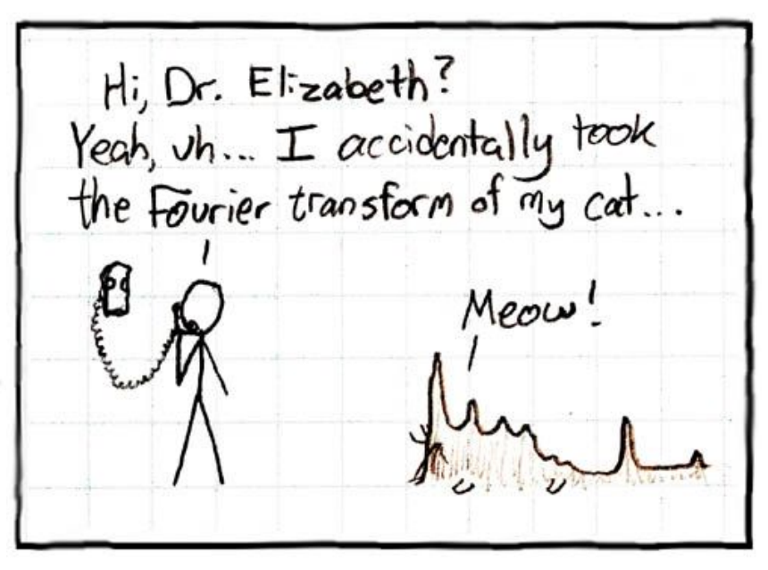
- ATCA cards have sensors connected to them which read data from the hardware.
- The asynchronous port driver connects to EPICS records in the IOC. The number of records it connects to is determined by the number of streams of data it will read.
- Then it launches a thread to stream the data to the corresponding EPICS records.
- An IOC is used for retrieving and updating the records and for running commands that execute additional general purpose functions like tracking the status of various streams and computing the fourier transform of a stream of data.



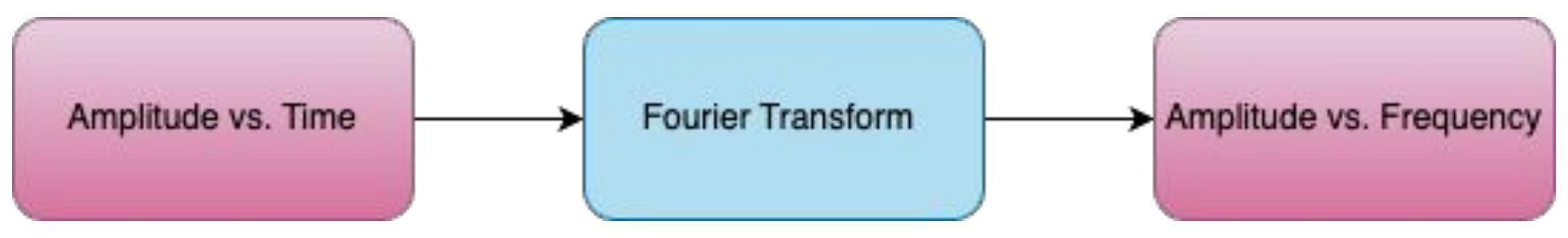
Fourier Transform

- The fourier transform is a mathematical tool that allows us to break down a waveform into its constituent sinusoids or frequencies.

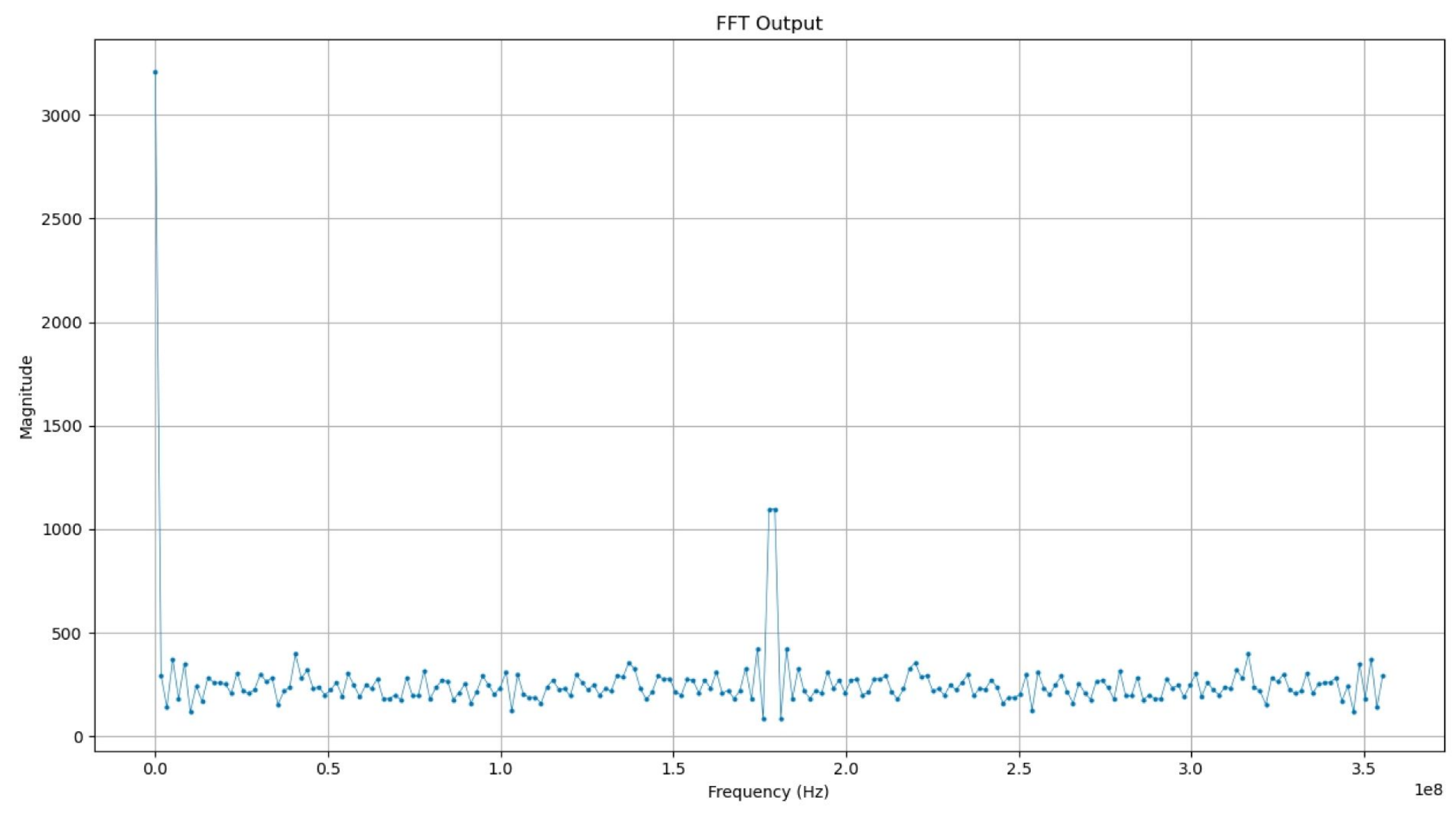


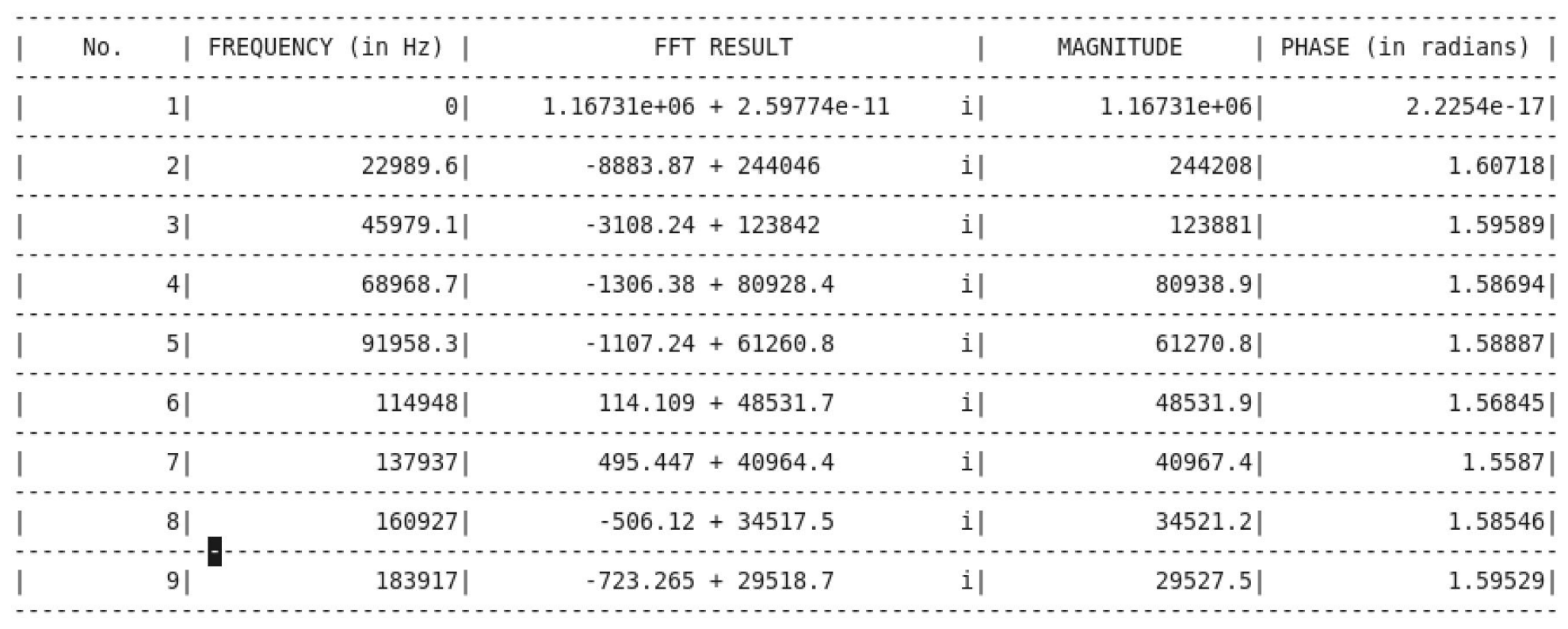

Hi, Dr. Elizabeth?
Yeah, uh... I accidentally took the Fourier transform of my cat...
Meow!

- The fast fourier transform (FFT) is an algorithm for efficiently computing the Discrete Fourier Transform of a waveform.
- In our asyn port driver, we have an input buffer that stores one million data points. We select a subset of those data points for our FFT by finding the global maximum of the waveform and using that to calculate a range of data points that can provide us information about the waveform.
- Then the FFT is computed for that range of values. The output of the FFT is in the form of complex numbers which we use to calculate the frequency bins or frequency components of the signal and their corresponding magnitudes and phases.



- We also added a python script which uses matplotlib to create a graph of the FFT output which represents the frequency domain of the

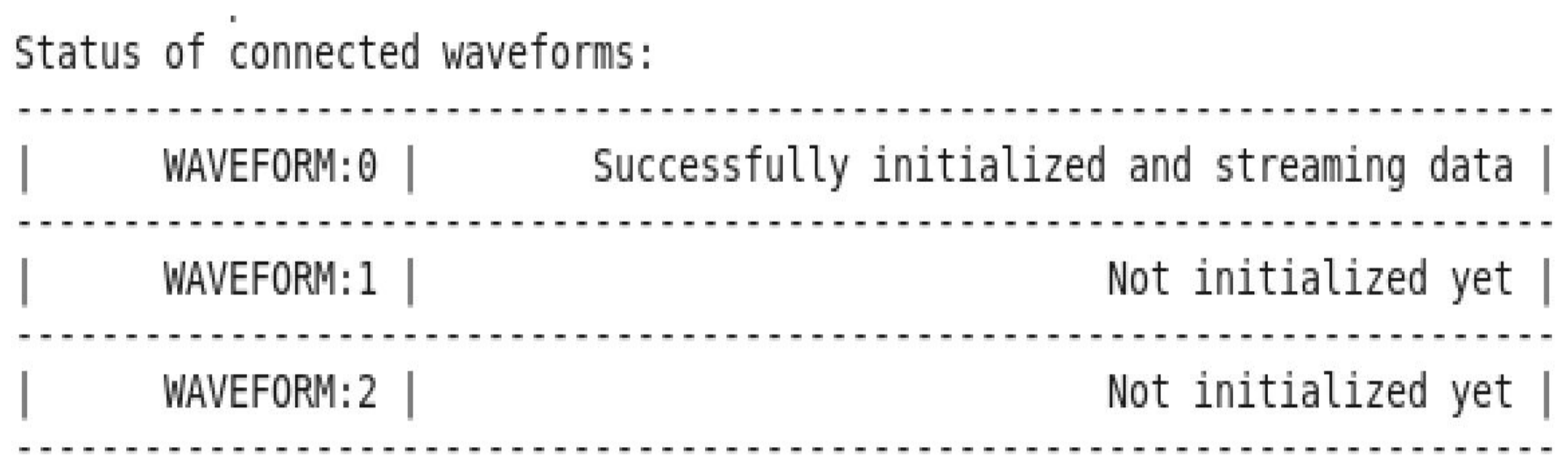




```
-----------------------------------------------------------------------------------------------
|    No.     | FREQUENCY (in Hz) |              FFT RESULT                |     MAGNITUDE     | PHASE (in radians) |
-----------------------------------------------------------------------------------------------
|           1|                  0|    1.16731e+06 + 2.59774e-11         i|        1.16731e+06|          2.2254e-17|
-----------------------------------------------------------------------------------------------
|           2|            22989.6|       -8883.87 + 244046              i|             244208|             1.60718|
-----------------------------------------------------------------------------------------------
|           3|            45979.1|       -3108.24 + 123842              i|             123881|             1.59589|
-----------------------------------------------------------------------------------------------
|           4|            68968.7|       -1306.38 + 80928.4             i|            80938.9|             1.58694|
-----------------------------------------------------------------------------------------------
|           5|            91958.3|       -1107.24 + 61260.8             i|            61270.8|             1.58887|
-----------------------------------------------------------------------------------------------
|           6|             114948|        114.109 + 48531.7             i|            48531.9|             1.56845|
-----------------------------------------------------------------------------------------------
|           7|             137937|        495.447 + 40964.4             i|            40967.4|              1.5587|
-----------------------------------------------------------------------------------------------
|           8|             160927|        -506.12 + 34517.5             i|            34521.2|             1.58546|
-----------------------------------------------------------------------------------------------
|           9|             183917|       -723.265 + 29518.7             i|            29527.5|             1.59529|
-----------------------------------------------------------------------------------------------
```

Waveform Status

- The waveform status function uses a map to keep track of the streaming status of the available waveforms.
- This is essential because the port driver will support multiple streams of data, and having access to their streaming status will help with health checking.
- Streaming data is a multi-step process, so there are various states the streams might be in, such as when the stream hasn't been initialized, when the stream has been initialized but isn't streaming yet, or when the stream has been initialized but there is no data in the buffer. Tracking these states helps ensure that we can monitor and manage the health of each data stream efficiently.



```
Status of connected waveforms:
----------------------------------------------------------------------
|      WAVEFORM:0 |       Successfully initialized and streaming data |
----------------------------------------------------------------------
|      WAVEFORM:1 |                                 Not initialized yet |
----------------------------------------------------------------------
|      WAVEFORM:2 |                                 Not initialized yet |
----------------------------------------------------------------------
```

Future Enhancements

- The port driver will be used in Beam Loss Monitors to detect the location and amount of beam loss.
- The next step would involve adding a function to the driver to find the maxima of the data buffer and convert that to a position in the beam line.
- We also need to ensure that the driver can simultaneously read data from multiple streams and save it to EPICS records.

Reflections

- This internship provided me with the opportunity to deepen my knowledge of C++ and introduced me to various technical concepts such as EPICS, Linux commands, and computer architecture.
- Additionally, this internship was unique because it allowed me to interact with professionals from different fields. I gained insights into the physics, chemistry, electrical engineering, and other disciplines involved in accelerators, enriching my understanding of the multifaceted nature of accelerator technology.

ACKNOWLEDGEMENTS

- I would like to thank my mentors, Lukas Ho-Koziol and Patrick Nisperos, for their constant guidance and infinite patience as I asked a million questions.
- I am grateful to the entire EED team for making this internship an extremely educational experience.
- I would also like to thank the LCLS team for giving me this amazing opportunity.

REFERENCES

- 26: Fourier - explain xkcd. https://www.explainxkcd.com/wiki/index.php/26:_Fourier
- FFT. https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft
- Slac-Epics-Modules. GitHub - slac-epics-modules/ycpswasyn: EPICS Module for communication with the SLAC's HPS based on CPSW + YAML. GitHub. https://github.com/slac-epics-modules/ycpswasyn