A simple vendor-neutral radio telescope backend software with real-time coherent de-dispersion

Yukai Zhou [1]
Supervisor: Rushuang Zhao [2,3], Youling Yue [2], Renxin Xu [1,4]

[1]School of Physics, Peking University

[2]National Astronomical Observatories, Chinese Academy of Sciences

[3]School of Physics and Electronic Science, Guizhou Normal University

[4]Kavli Institute for Astronomy and Astrophysics, Peking University

2023.7.5 [1]

[1]development of this software started from 2022.5.31

Contents

1 TL; DR . . . 3

2 Introduction . . . 4

3 Method . . . 11

3.1 Programming Framework . . . 12

3.2 Pipeline Structure . . . 14

4 Result . . . 15

4.1 Simulated data . . . 16

4.2 Example baseband data from dspsr . . . 17

4.3 Observation of Crab Pulsar . . . 19

5 Compatibility & Benchmark . . . 27

5.1 GPUs . . . 28

5.2 Optimizations for restricted FP64 capability . . . 29

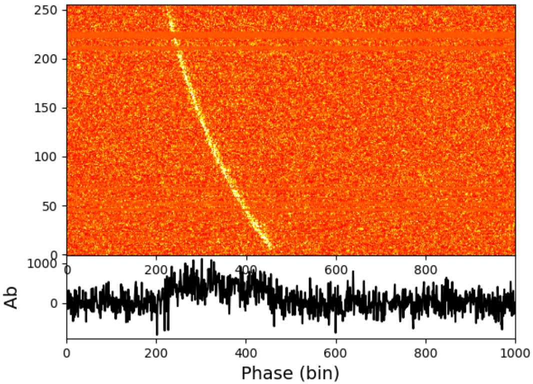
5.3 GPUs vs. CPUs . . . 31

6 Conclusion . . . 32

7 Reference . . . 34

TL; DR

(a) the 4.5 m antenna in Guizhou Normal University



(b) spectrum processed using dspsr, dedispersed with inaccurate DM, 1 bin = 512 ns

Figure 1.

- developed a simple backend that capable of real-time coherent de-disperion
- paved the way of developing GPGPU programs using SYCL, instead of CUDA
- detected giant pulses from Crab using 4.5 m antenna

Introduction
Pulsar

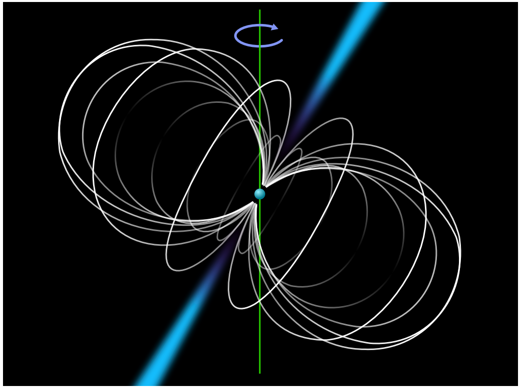


Figure 2. Schematic of pulsar. [2]

- A pulsar is a highly magnetized rotating neutron star that emits beams of electromagnetic radiation out of its magnetic poles.

[2] Credit: Mysid, Roy Smits, `https://en.wikipedia.org/wiki/File:Pulsar_schematic.svg`

Introduction
Giant pulses

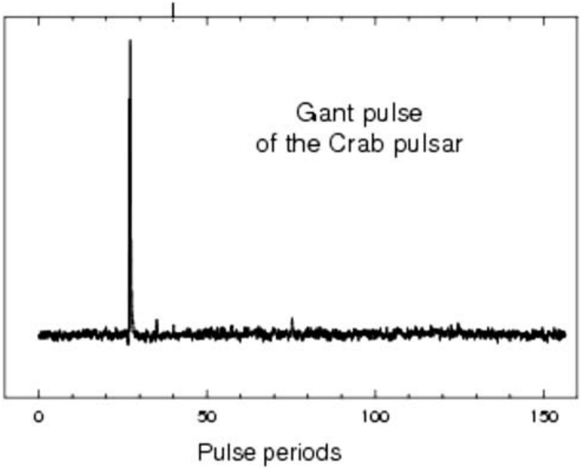


Figure 3. (Kuzmin (2007))

- Typical length of giant pulses from B1937+21 is 15 ns (Soglasnov et al. (2004));
- Pulse structure from Crab is as short as 2 ns (Hankins et al. (2003)).

Introduction
Dispersion

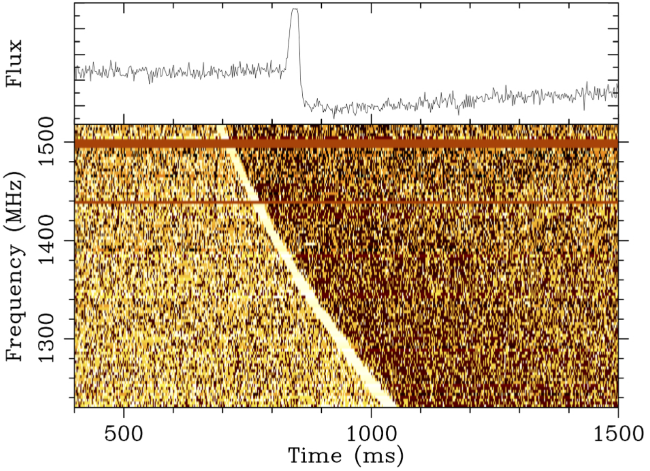


Figure 4. A typical signal with dispersion, FRB 010724 here (Petroff, Hessels, and Lorimer (2019))

- When electromagnetic wave travels through plasma, low frequency component is slower than high frequency component, so signal is stretched if look at time series.

Introduction

Incoherent dedispersion

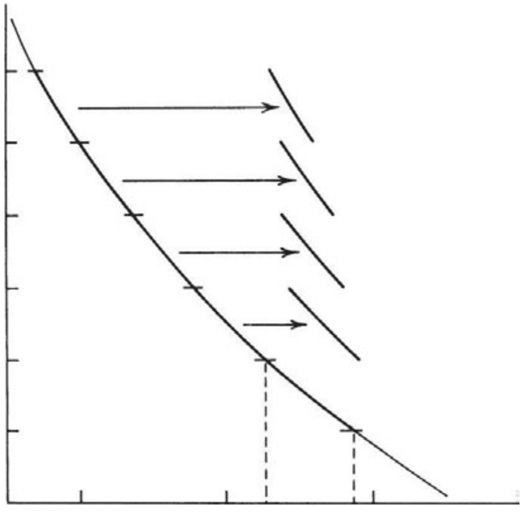


Figure 5. Schematic of incoherent dedispersion (Lyne and Graham-Smith (2012))

- Incoherent dedispersion: calculate time delay of each frequency channel and re-align them
    - signal delay in each frequency channel is not handled, hence time resolution of dedispersed signal is affected.

Introduction
Coherent dedispersion

- using some electrodynamics (Song (2022)), phase delay of given frequency f is

$$\Delta\phi = 2\pi D \frac{\mathrm{DM}}{f} = \Delta\phi = 2\pi D \frac{\mathrm{DM}}{f_0 + \Delta f}$$

  where f_0 is reference frequency
- in accordance with Lorimer and Kramer (2004) only the quadratic term has effect,

$$\Delta\phi = \frac{2\pi D}{(f_0 + \Delta f) f_0^2} \cdot \mathrm{DM} \cdot \Delta f^2$$

  should be corrected in frequency domain
- large amount of FFT is required

Introduction

Choosing Target Device

- Example input sample rate: $2 \times (1 \times 10^9)$ samples / second (1000 - 1500 MHz)
- FFT algorithm is $O(n \log n)$, other procedures $\sim O(n)$, so most time is spent on FFT (in theory)
- hence each FFT operation should be handled within at most 0.5 second

Introduction

FFT benchmark

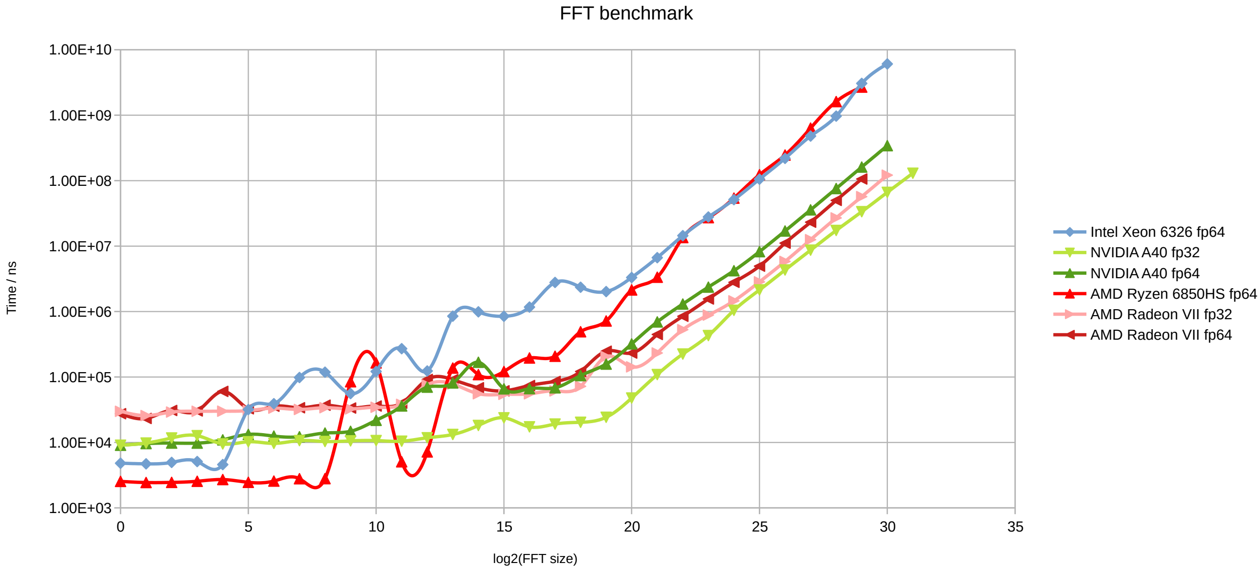


Figure 6. Benchmark of FFT on different devices. cuFFT for NVIDIA GPUs, hipFFT + rocFFT for AMD GPUs and FFTW3 (Frigo and Johnson (2005)) for CPUs.

- CPU cannot satisfy the requirement, GPGPU or other accelerator seems a must.

Method

1 TL; DR . . . 3

2 Introduction . . . 4

3 Method . . . 11

3.1 Programming Framework . . . 12

3.2 Pipeline Structure . . . 14

4 Result . . . 15

4.1 Simulated data . . . 16

4.2 Example baseband data from dspsr . . . 17

4.3 Observation of Crab Pulsar . . . 19

5 Compatibility & Benchmark . . . 27

5.1 GPUs . . . 28

5.2 Optimizations for restricted FP64 capability . . . 29

5.3 GPUs vs. CPUs . . . 31

6 Conclusion . . . 32

7 Reference . . . 34

Method

Programming Framework

- API: SYCL 2020
- reason:
  - vendor neutrality
    - ~~CUDA~~
  - wide compatibility
    - ~~CUDA~~
  - better if open source
    - ~~CUDA~~
  - relatively lower programming difficulty
    - ~~OpenCL~~
  - high performance
    - ~~CPU programming (including OpenMP on CPU)~~
  - choice of some other software
    - GROMACS adopted SYCL
    - Blender 3.0 dropped support of OpenCL, in favour of HIP and oneAPI
- although in practice only NVIDIA GPUs are common.

Method

Programming Framework

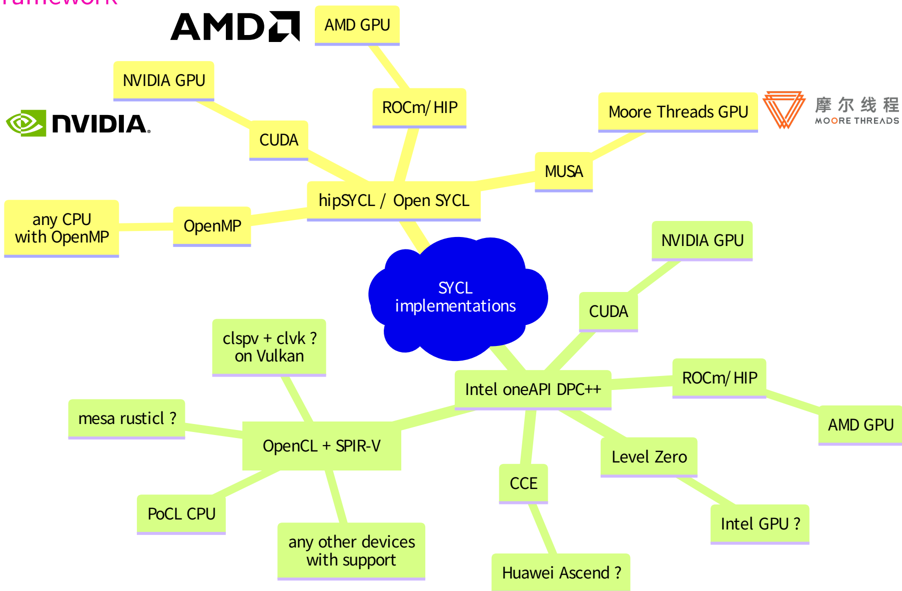


Figure 7. Some implementations and backends of SYCL. Devices with "?" have not been tested. [3]

- No code change required! (except for vendor specific things...)

[3] hipSYCL / Open SYCL: Alpay and Heuveline (2020), Alpay, Soproni, et al. (2022), Alpay and Heuveline (2023), J. M. Meyer (2021), and J. Meyer et al. (2023). hipSYCL MUSA backend is ported by the author based on SSCP backend. Huawei Ascend CCE: Feng, Maghareh, and Wang (2021).

Method

Pipeline Structure

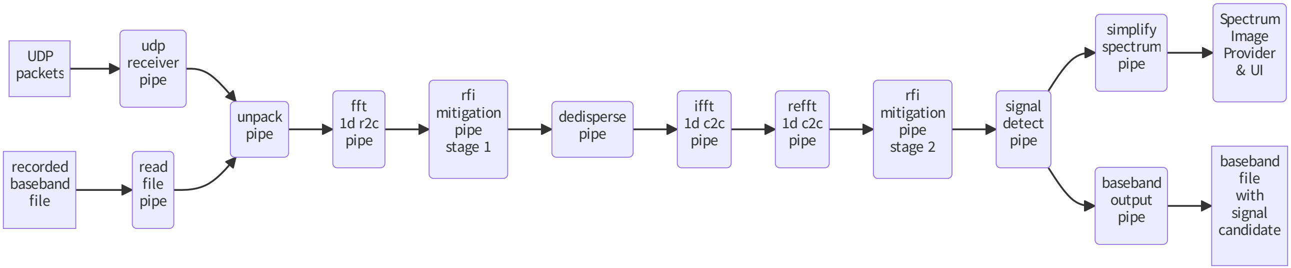


Figure 8. Pipeline structure, inspired by Jiang et al. (2021) and Jiang (2022)

- Coherent dedispersion has been mentioned before.
- RFI mitigation stage 1 is based on average intensity.
- RFI mitigation stage 2 is based on spectral kurtosis method.
  (Vrabie, Granjon, and Serviere (2003), Nita (2016), Taylor et al. (2018), and Jiang (2022))

Result

1 TL; DR . . . 3
2 Introduction . . . 4
3 Method . . . 11
3.1 Programming Framework . . . 12
3.2 Pipeline Structure . . . 14
4 Result . . . 15
4.1 Simulated data . . . 16
4.2 Example baseband data from dspsr . . . 17
4.3 Observation of Crab Pulsar . . . 19
5 Compatibility & Benchmark . . . 27
5.1 GPUs . . . 28
5.2 Optimizations for restricted FP64 capability . . . 29
5.3 GPUs vs. CPUs . . . 31
6 Conclusion . . . 32
7 Reference . . . 34

Result

Simulated data

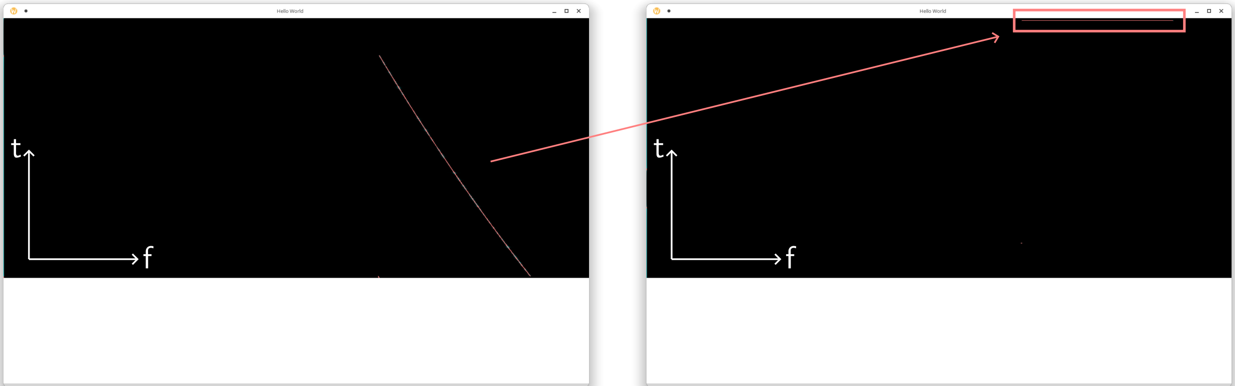


Figure 9. Spectrum (waterfall) of simulated data before and after dedispersion. Left: DM = 0 (original), Right: DM = 563.9.

- credit: [4]

[4] simulation code `generatebuffer.py` and reference dedispersion code `de_dispersion.py` by Shiling Yu

Result

Example baseband data from dspsr

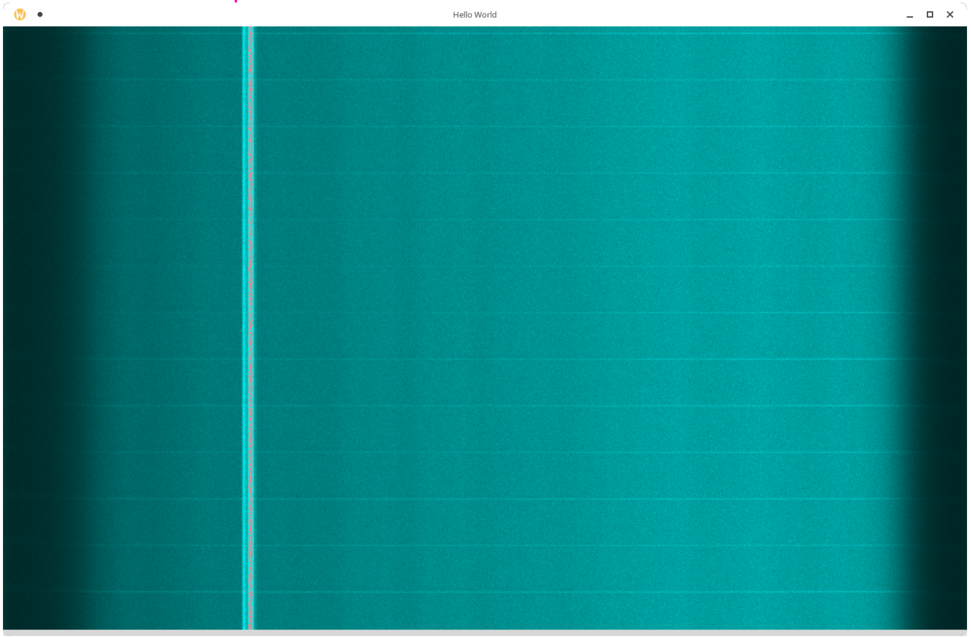


Figure 10. dedispersed spectrum of PSR J1644-4509, DM = 478.80

Result

Example baseband data from dspsr

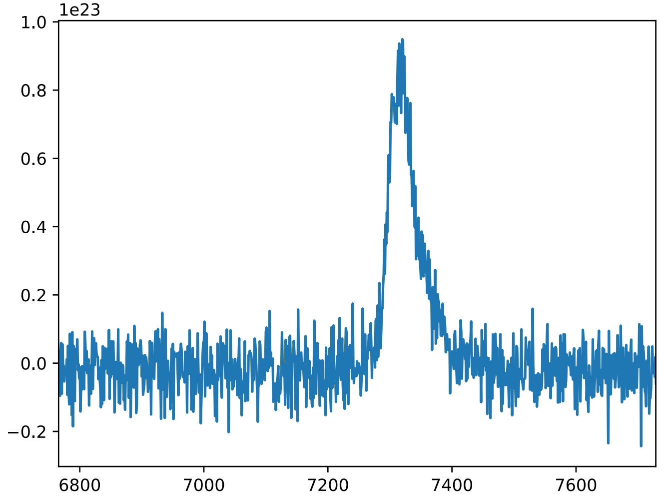


Figure 11. time series of one pulse summed from dedispersed spectrum on last page, DM = 478.80

- Data from DSPSR example [5]

[5] `https://dspsr.sourceforge.net/manuals/dspsr/example.shtml`

Result

Observation of Crab Pulsar

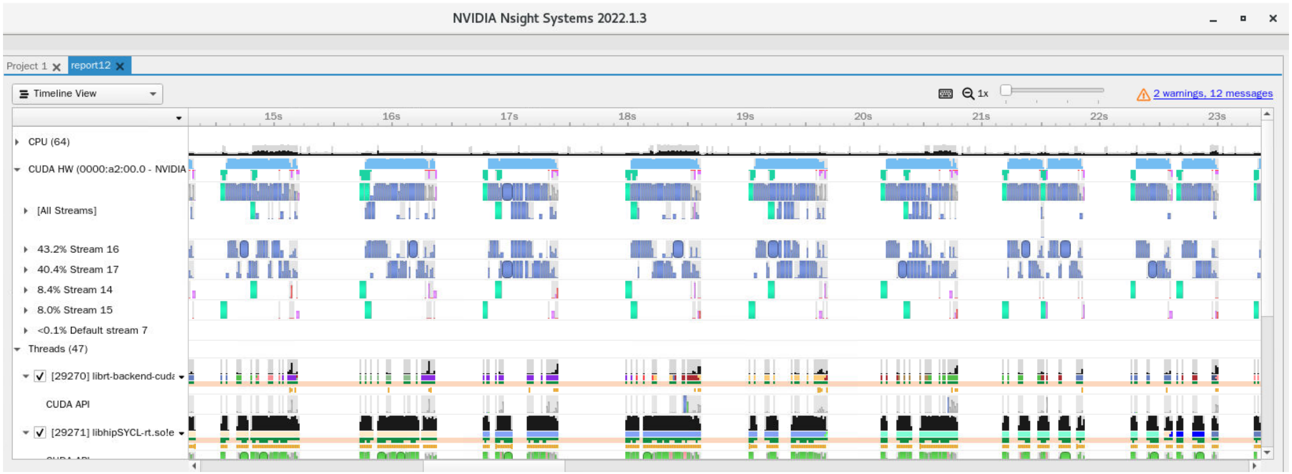


Figure 12. Timeline when receiving UDP packets from ROACH 2 + 4.5 m antenna

- can handle $2 \times (1 \times 10^9)$ samples within 1 second

Result

Observation of Crab Pulsar

Figure 13. the 4.5 m antenna in Guizhou Normal University

Result

Observation of Crab Pulsar

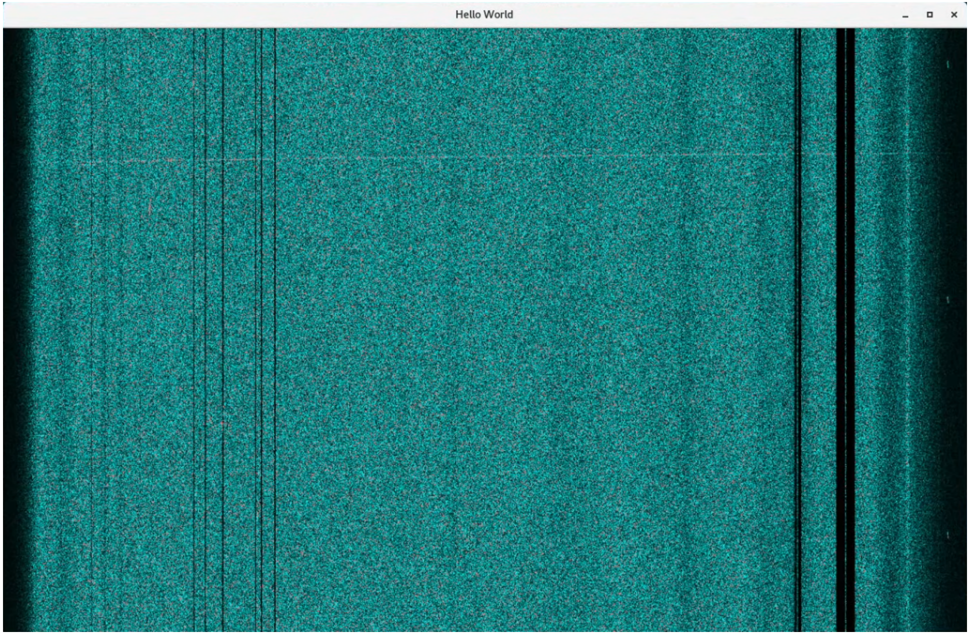
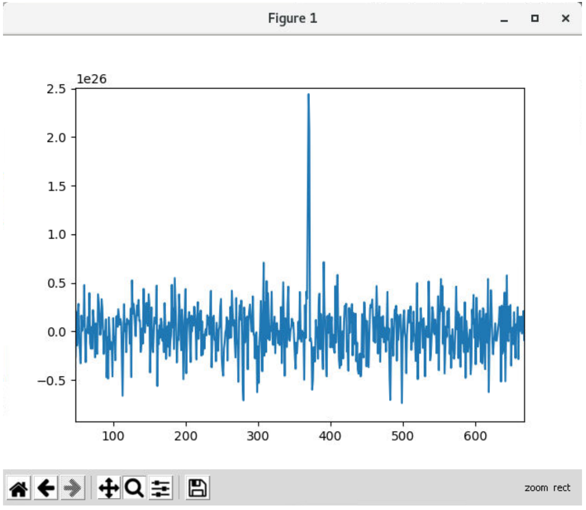
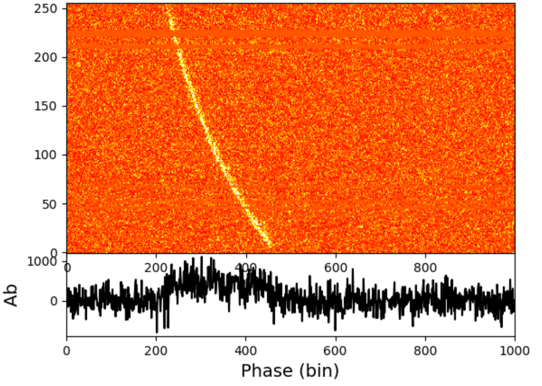


Figure 14. spectrum of first captured giant pulse of Crab on 2023.01.18, target DM = 56.778

Result
Observation of Crab Pulsar



(a) time series, 1 bin = 32.768 μs



(b) spectrum processed using dspsr, dedispersed with inaccurate DM, 1 bin = 512 ns

Figure 15. info of first captured giant pulse of PSR J0534+2200, target DM = 56.778

- captured using a 4.5 m antenna on 1000 MHz - 1500 MHz

Result

Observation of Crab Pulsar

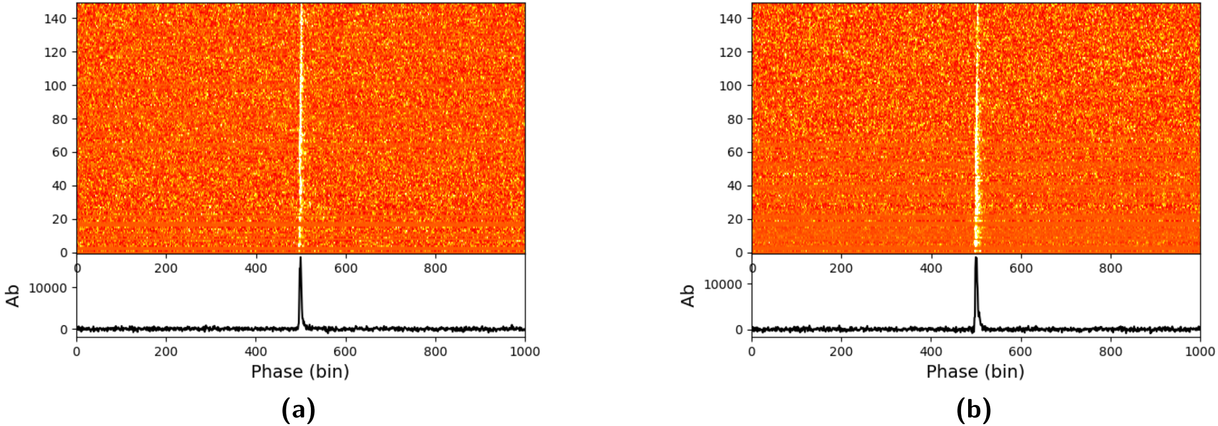


Figure 16. some captured and dedispersed giant pulses (20230304, 2nd), 1 bin = 512 ns

Result

Observation of Crab Pulsar

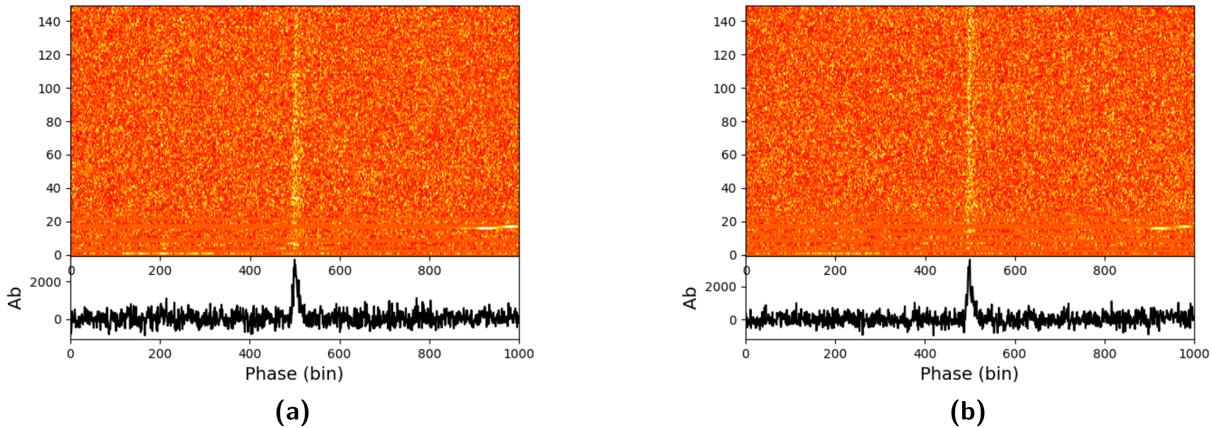
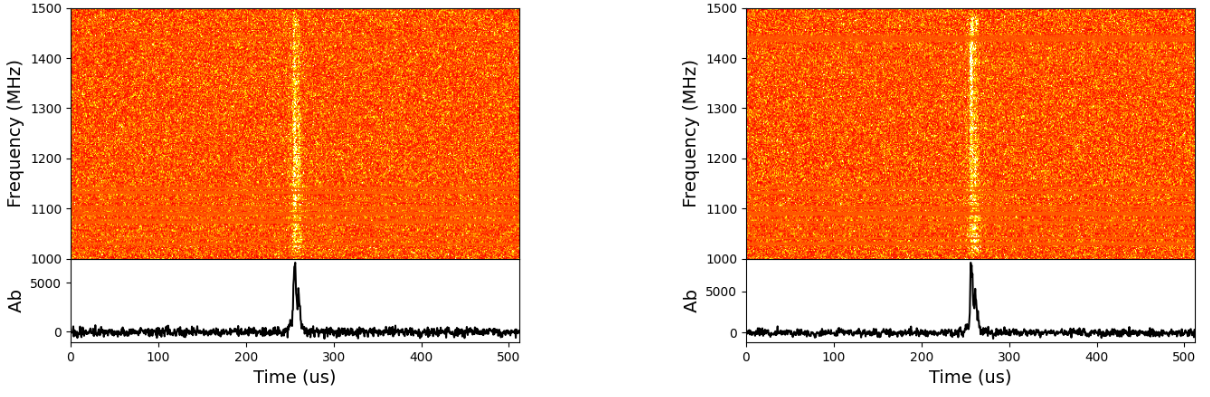


Figure 17. some captured and dedispersed giant pulses (20230305, 14th), 1 bin = 512 ns

Result

Observation of Crab Pulsar



(a) **(b)**

Figure 18. some captured and dedispersed giant pulses (20230320, 9th), 1 bin = 512 ns

Result

Observation of Crab Pulsar

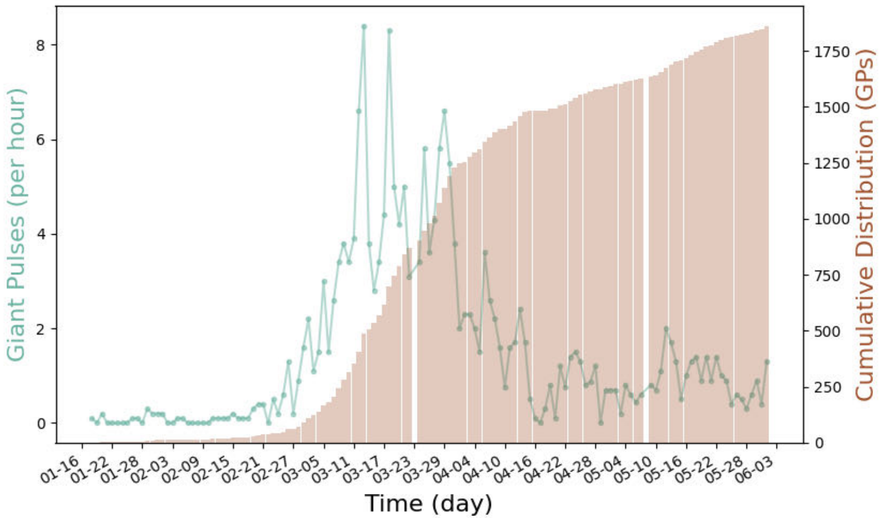


Figure 19. Giant pulse rate and count since the pipeline works, using one 4.5 m antenna

Compatibility & Benchmark

1 TL; DR . . . 3
2 Introduction . . . 4
3 Method . . . 11
3.1 Programming Framework . . . 12
3.2 Pipeline Structure . . . 14
4 Result . . . 15
4.1 Simulated data . . . 16
4.2 Example baseband data from dspsr . . . 17
4.3 Observation of Crab Pulsar . . . 19
5 Compatibility & Benchmark . . . 27
5.1 GPUs . . . 28
5.2 Optimizations for restricted FP64 capability . . . 29
5.3 GPUs vs. CPUs . . . 31
6 Conclusion . . . 32
7 Reference . . . 34

Compatibility & Benchmark

GPUs

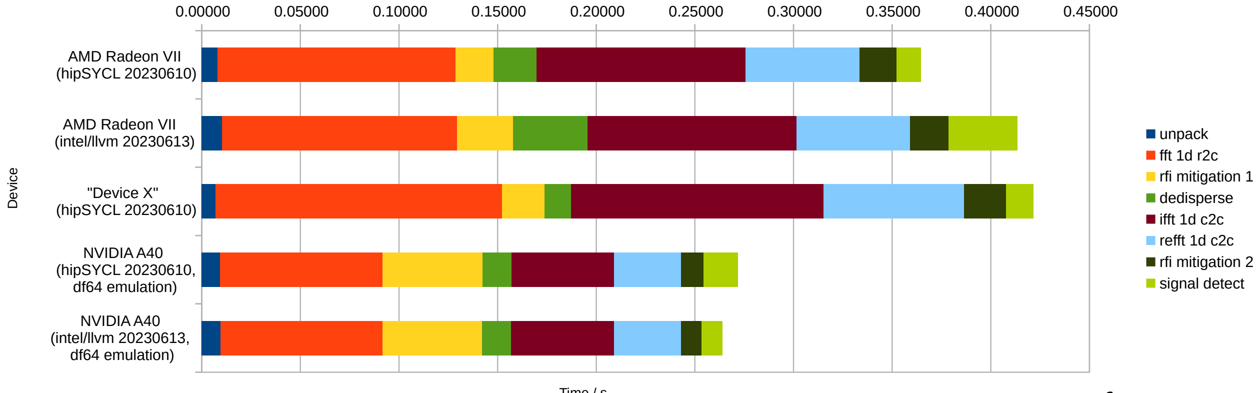


Figure 20. Benchmark result of AMD Radeon VII, "Device X" & NVIDIA A40. [6]

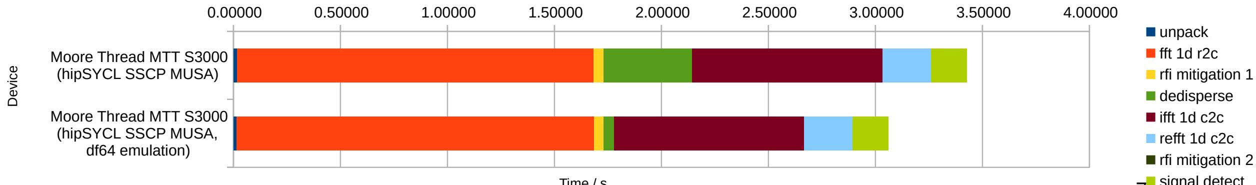


Figure 21. Benchmark result of Moore Threads MTT S3000, a very immature GPU. [7]

[6] Do not compare performance directly, as they are not of the same generation. For reference, theoretical performance (FP32): AMD Radeon VII: 13.44 TFLOPS, NVIDIA A40: 37.42 TFLOPS, Moore Threads MTT S3000: 15.2 TFLOPS. Detailed information of "Device X" is not available due to policy of the vendor.

[7] Results incomplete and may inaccurate, waiting for profiling tools from vendor.

Compatibility & Benchmark

Optimizations for restricted FP64 capability

- for a typical L band antenna on $1000 - 1500\,\mathrm{MHz}$, and target $\mathrm{DM} = 1000\,\mathrm{pc} \cdot \mathrm{cm}^{-3}$,

$$\phi_{\max} \approx 2.90 \times 10^9 \approx 2^{31}$$

- Only the value modded 2π has effect; "significand bits" of `float32` and `float64` are 24 and 53 respectively ("IEEE Standard for Floating-Point Arithmetic" (2019)),
- `float32` is not sufficient, `float64` seems a must.
- But some vendor restricted `float64` performance of most of their devices.

Compatibility & Benchmark

Optimizations for restricted FP64 capability

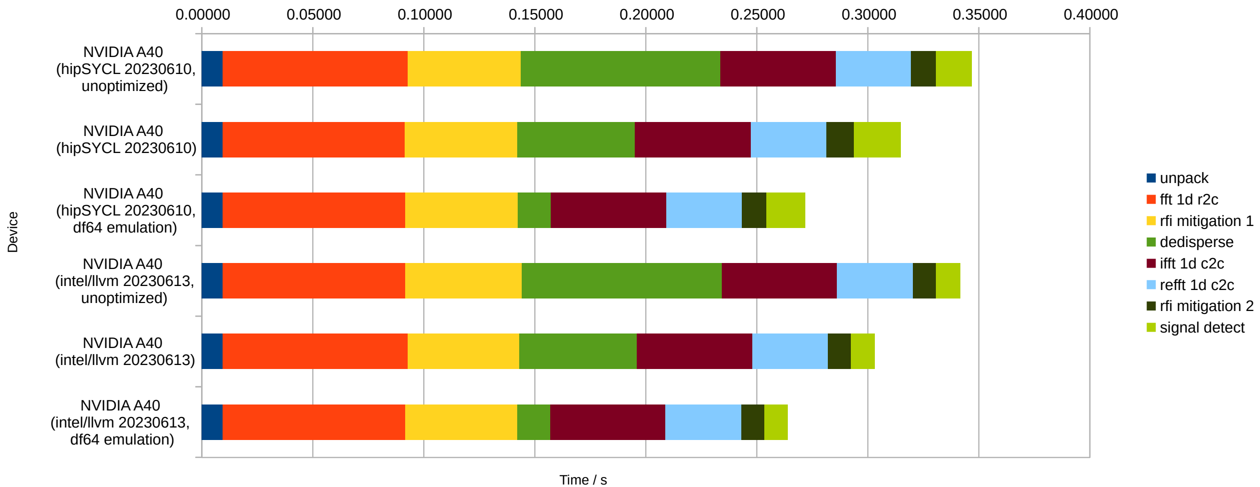


Figure 22. Benchmark result of NVIDIA A40 (FP64 : FP32 = 1:64), with different manual optimizations.

- Emulate `float64` with unevaluated sum of 2 `float32`s (`df64`), a common technique used at the early stages of development of GPU.
- Not required for AMD Radeon VII! [8]

[8] For comparison, FP64 : FP32 FLOPS ratio: AMD Radeon VII 1 : 4, its professional counterpart AMD Instinct MI50 1 : 2.

Compatibility & Benchmark

GPUs vs. CPUs

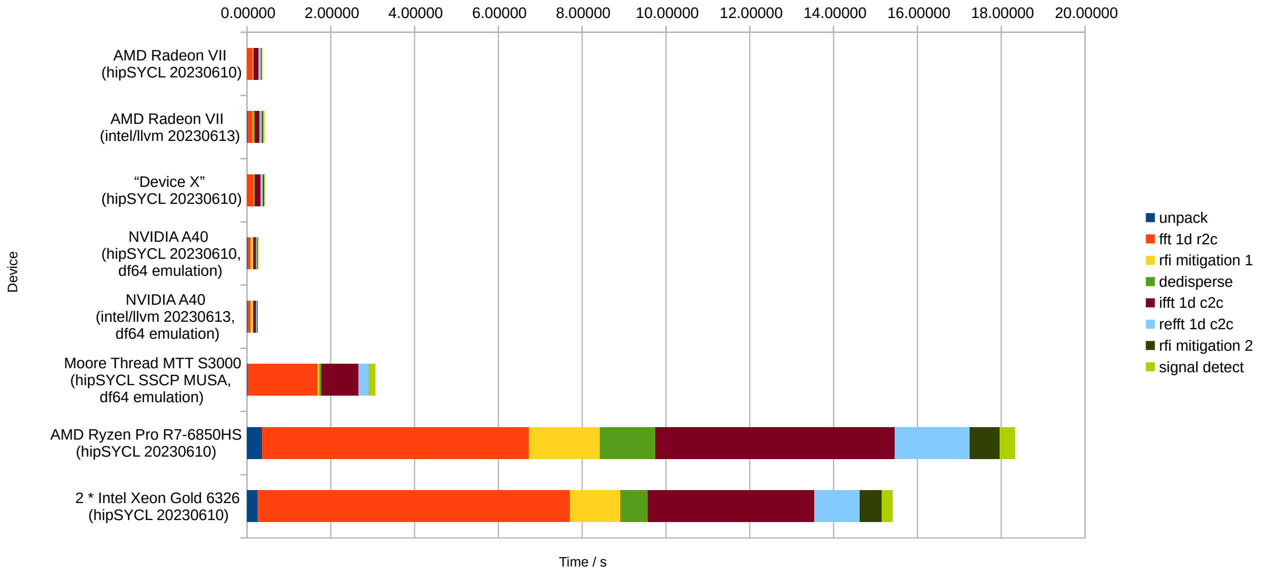


Figure 23. Comprehensive benchmark result of AMD Radeon VII, "Device X", NVIDIA A40, Moore Threads MTT S3000, AMD Ryzen Pro R7-6850HS, and 2 $\times$ Intel Xeon Gold 6326 with different SYCL implementations available.

Conclusion

- presented a simple radio telescope backend software
- implemented real-time coherent dedispersion
- introduced open standard SYCL to this field, instead of vendor specific ones like CUDA
- detected giant pulses from Crab
- tested and compared performance of AMD GPUs, NVIDIA GPUs, "Device X", Moore Threads GPUs, AMD CPUs and Intel CPUs

Thank you for listening!

code available at `https://github.com/fxzjshm/simple-radio-telescope-backend`

Reference I

Alpay, Aksel and Vincent Heuveline (2020). "SYCL beyond OpenCL: The Architecture, Current State and Future Direction of hipSYCL". In: Proceedings of the International Workshop on OpenCL. IWOCL '20. Munich, Germany: Association for Computing Machinery. ISBN: 9781450375313. DOI: 10.1145/3388333.3388658. URL: https://doi.org/10.1145/3388333.3388658.

Alpay, Aksel and Vincent Heuveline (2023). "One Pass to Bind Them: The First Single-Pass SYCL Compiler with Unified Code Representation Across Backends". In: Proceedings of the 2023 International Workshop on OpenCL. IWOCL '23. Cambridge, United Kingdom: Association for Computing Machinery. ISBN: 9798400707452. DOI: 10.1145/3585341.3585351. URL: https://doi.org/10.1145/3585341.3585351.

Alpay, Aksel, Bálint Soproni, et al. (2022). "Exploring the Possibility of a HipSYCL-Based Implementation of OneAPI". In: International Workshop on OpenCL. IWOCL'22. Bristol, United Kingdom, United Kingdom: Association for Computing Machinery. ISBN: 9781450396585. DOI: 10.1145/3529538.3530005. URL: https://doi.org/10.1145/3529538.3530005.

Reference II

Feng, Wilson, Rasool Maghareh, and Kai-Ting Amy Wang (2021). "Extending DPC++ with Support for Huawei Ascend AI Chipset". In: International Workshop on OpenCL. IWOCL'21. Munich, Germany: Association for Computing Machinery. ISBN: 9781450390330. DOI: 10.1145/3456669.3456684. URL: https://doi.org/10.1145/3456669.3456684.

Frigo, M. and S.G. Johnson (2005). "The Design and Implementation of FFTW3". In: Proceedings of the IEEE 93.2, pp. 216–231. DOI: 10.1109/JPROC.2004.840301.

Hankins, T. H. et al. (Mar. 2003). "Nanosecond radio bursts from strong plasma turbulence in the Crab pulsar". In: Nature 422.6928, pp. 141–143. ISSN: 1476-4687. DOI: 10.1038/nature01477. URL: https://doi.org/10.1038/nature01477.

"IEEE Standard for Floating-Point Arithmetic" (2019). In: IEEE Std 754-2019 (Revision of IEEE 754-2008), pp. 1–84. DOI: 10.1109/IEEESTD.2019.8766229.

Jiang, Jinchen (2022). "Pulsar Single-Pulse Studies and Polarization Measurement with FAST". PhD thesis. Peking University.

Jiang, Jinchen et al. (2021). "Baseband polarimetry of millisecond pulsars using FAST". in prep.

Reference III

- Kuzmin, A. D. (Apr. 2007). "Giant pulses of pulsar radio emission". In: Astrophysics and Space Science 308.1, pp. 563–567.
- Lorimer, D. R. and M. Kramer (2004). Handbook of Pulsar Astronomy. Vol. 4.
- Lyne, Andrew and Francis Graham-Smith (2012). Pulsar Astronomy. 4th ed. Cambridge Astrophysics. Cambridge University Press. DOI: 10.1017/CBO9780511844584.
- Meyer, Joachim et al. (2023). "Implementation Techniques for SPMD Kernels on CPUs". In: Proceedings of the 2023 International Workshop on OpenCL. IWOCL '23. Cambridge, United Kingdom: Association for Computing Machinery. ISBN: 9798400707452. DOI: 10.1145/3585341.3585342. URL: https://doi.org/10.1145/3585341.3585342.
- Meyer, Joachim Mathias (2021). "Compiler-assisted optimizations for data-parallel paradigms in hipSYCL". MA thesis. Heidelberg University. URL: https://joameyer.de/hipsycl/Thesis_JoachimMeyer.pdf.
- Nita, Gelu (Mar. 2016). "Spectral Kurtosis Statistics of Transient Signals". In: Monthly Notices of the Royal Astronomical Society 458, stw550. DOI: 10.1093/mnras/stw550.

Reference IV

- Petroff, E., J. W. T. Hessels, and D. R. Lorimer (2019). "Fast radio bursts". English. In: The Astronomy and astrophysics review 27.1, pp. 1–75.
- Soglasnov, Vladimir A. et al. (2004). "Giant Pulses from PSR B1937+21 with Widths $\leq$ 15 Nanoseconds and $T_b \geq 5 \times 10^{39}\mathrm{K}$, the Highest Brightness Temperature Observed in the Universe". In: The Astrophysical Journal 616, pp. 439–451.
- Song, Huichao (2022). Electrodynamics. Electrodynamics, 2022 Spring, Peking University.
- Taylor, Jacob et al. (2018). "Spectral Kurtosis-Based RFI Mitigation for CHIME". English. In: Journal of Astronomical Instrumentation 8.1.
- Vrabie, Valeriu, Pierre Granjon, and Christine Serviere (2003). "Spectral kurtosis: from definition to application". In: 6th IEEE International Workshop on Nonlinear Signal and Image Processing (NSIP 2003). Grado-Trieste, Italy. URL: https://hal.science/hal-00021302.