Development of Real-Time High-Density Pulsar Data Transmission and Processing for Grid Synchronization

Wei Qiu, *Member, IEEE*, Zhangqing Chen, Cheng Liu, Yao Zheng, Sihao Tang, *Student Member, IEEE*, He Yin, *Senior Member, IEEE*, Wenxuan Yao, *Senior Member, IEEE*, and Yilu Liu, *Fellow, IEEE*

***Abstract*—Taking advantage of the extreme stability of the pulsar period, it can serve as the timing source for grid synchronization to compensate for the timing drift instigated by the loss of GPS signal. Nevertheless, the real-time transmission and processing of the pulsar data suffer from its high-frequency data rate, varying from megahertz to gigahertz, resulting in reduced computing speed and increased time delay. To mitigate this issue, the hardware and software frameworks are implemented for the high-density pulsar data transmission and processing for grid synchronization in this research. Initially, the high-density pulsar data is transferred using open-source software. The complementary duty cycle timing module is designed to coordinate the operation of the dual-channel high-speed interface and software. Subsequently, the multiple-threading is applied to the receiving, parsing, and splicing pulsar data. Next, the pulsar signal extraction method is implemented based on the polyphase filterbank and time of arrival estimation. Ultimately, real-time performance verification experiments are carried out for different components under two hardware platforms. The results demonstrate that only 0.482 s is required for processing 4 Gigabyte data through multiple-threading, which is 3.8 times faster than the single thread. The pulsar signal extraction can also be executed within 707 ms for 4.8 seconds of data, thereby indicating that real-time requirements can be met.**

***Index Terms*—Grid synchronization, high-density pulsar data, timing module, data transmission and processing, multiple-threading.**

I. Introduction

RECENTLY, the development of a pulsar-based timing source for grid synchronized sampling has gained traction due to its exceptional period stability. It is reported that the frequency stability of millisecond pulsar PSR 1937 + 21 exceeds 6×10^{-14} over a span of four months [1], [2]. The primary advantages of pulsar-based timing are that it can serve as a backup non-satellite clock source and defense against GPS signal loss and spoofing. For instance, the tolerance for the timing source is 1 μs, and the maximum timing error can not be exceeded ± 26 μs and ± 31 μs for the 60 Hz grid and 50 Hz power system, respectively in the area of grid synchronized [3]. However, the local system can only withdrawn for several minutes because the timing drift will easily exceed tens of microseconds based on the actual testing in [4]. Concurrently, pulsars have great potential for achieving high timing precision given the stability of the pulsars [5]. The signals emitted by pulsars are electromagnetic waves, with the most distinct profile of most pulsars being extractable from signals around 1440 MHz [6]. To facilitate the sampling of the pulsars, the frequency range of the radio telescope typically spans from 0.3 to 116 GHz [7]. To drastically alleviate the computational burden, the mixer can be used to down-convert the high-frequency input signal to hundreds of megabytes [8]. However, such a high sampling rate still challenges the real-time performance of establishing the pulsar-based timing source, especially for the continuous observation of pulsars [9].

To enable pulsar signals to be used for grid synchronization, a series of critical stages must be meticulously executed [10]. These stages encompass the collection and transmission of pulsar signals, the extraction of pulse profiles, and time of arrival estimation (TOA)-based grid synchronization signal generation. The efficacy of the initial stage is contingent upon the capabilities of the embedded hardware and the velocity of sampling. The rest two stages can be implemented on the data server or data center, which shifts the focus towards the intricacies and computational efficiency of the algorithmic processes involved.

In the initial stage, to expedite fast pulsar signal collection and transmission, several hardware platforms have been deployed. Notably, the second generation of the Reconfigurable Open Architecture Computing Hardware (ROACH) [11] platform is developed by merging the characteristics from the IBOB and BEE2 platforms [12], where the IBOB and BEE2 both are the early boards with high-speed and high memory bandwidth [13]. Additionally, a total of 14 Collaboration for Astronomy Signal

Manuscript received 23 December 2023; revised 1 April 2024; accepted 29 May 2024. Date of publication 10 July 2024; date of current version 23 September 2024. Paper 2023-PSEC-1586.R1, presented at the 2023 IEEE Industry Applications Society Annual Meeting, Nashville, TN, USA, Oct. 29–Nov. 02, an approved for publication in the IEEE Transactions on Industry Applications by the Power Systems Engineering Committee of the IEEE Industry Applications Society [DOI: 10.1109/IAS54024.2023.10406690]. This work was supported in part by the National Natural Science Foundation of China under Grant 52177078 and Grant 52307093 and in part by Hunan Provincial Natural Science Foundation of China under Grant 2023JJ40151. *(Corresponding author: Wenxuan Yao.)*

Wei Qiu, Zhangqing Chen, Cheng Liu, Yao Zheng, Sihao Tang, He Yin, and Wenxuan Yao are with the College of Electrical and Information Engineering, Hunan University, Changsha 410082, China (e-mail: qiuwei@hnu.edu.cn; liu@utk.edu; tangsihao@hnu.edu.cn; wenxuanyao@hnu.edu.cn).

Yilu Liu is with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996 USA, and also with the Oak Ridge National Laboratory, Oak Ridge, TN 37831 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIA.2024.3425807.

Digital Object Identifier 10.1109/TIA.2024.3425807

Processing and Electronics Research (CASPER) [14], [15] Smart Network ADC Processor (SNAP) [16] field-programmable gate array (FPGA) boards are built for the CSIRO Parkes Telescope in New South Wales, Australia [17]. A total of 27 compute nodes are installed for this telescope with a 1.125 GHz bandwidth. Importantly, the output of the pulsar data is quantized into 8 bits to reduce the computation. Another example is the design of China re-configurable for a Five-hundred-meter Aperture Spherical radio Telescope (FAST) in [18]. By integrating multiple single-FPGA hardware platforms, an extremely high speed and huge memory configuration can be applied for long-life and high-performance operation. This hardware configuration is usually utilized in some well-known projects such as the Square Kilometre Array (SKA), and Karoo Array Telescope (KAT). Typically, aiming at the 4GS/s to 6GS/s sampling rate, the sampling data is routed through the 10 Gigabit Ethernet (GbE) module, which is capable of transmitting Ethernet frames at a bandwidth of 10 gigabits per second. The challenge of the 10 GbE lies in mitigating the risk of data frame loss problems during high-density data transmission.

Apart from improving the hardware configuration, software optimization is another way to help refine pulsar signal profiles and boost computational efficiency in the rest two stages. It is time-consuming to search for the objective pulse from the heavily noise-laden signal. For example, the Wavelet denoising algorithm is developed to increase the signal-to-noise in terms of the cross-correlation technology [19], and a near 0.9867 coefficient is achieved. Then, the epoch folding of the data with repetition periods is utilized using the jumping average window [20]. Results demonstrated that the detected time decreased while keeping a high signal to noise. Additionally, some research develops software packages to process the pulsar data. For example, in [21], a Python/C++ framework named Bifrost is developed to process the high-throughput data stream. It leverages the high performance of Graphics Processing Units (GPUs). Another technology based on the GPU parallel computing technology is developed for ultra-wide bandwidth pulsar processing [22]. Besides, the PSRCHIVE is an open-source and object-oriented data analysis software library that is suited for pulsar astronomy [23]. It integrates different data processing functions, such as modifying the metadata, and produces customized quality plots. The other software are TEMPO2 [24], PSRCAT [25], and PRESTO [26]. The advantages of these libraries are that data processing would become easier and there is no need to dig into the details of the pulsar signal processing. Conversely, the difficulty is that it would be challenging to process the pulsar signal if the data format is different from what the library stipulates. In instances such as debugging hardware and software with analog front-end data, where the format is often bespoke, this can pose a considerable challenge.

Before the pulsar signal can be deployed into the grid synchronization, the TOA-based technology is necessary to estimate the frequency of the crystal oscillator. Approaches in both the temporal and frequency domains are applicable. For instance, the TOA of the observation pulsar can be determined by the template-matching methodology by comparing the phase offset between the template and observation [27]. However, the results may not be reliable because it is sensitive to noise. To this end, the frequency method is proposed. In [28], the channelized discrete Fourier transform is implemented to improve the timing precision using the broad-band timing data. It improves the uncertainties of TOA measurement by approximately 20 percent in the median. The complexity of this frequency domain is approximately $O(Nlog(N))$, where N is the length of the data, and the real-time performance can be satisfied in most of the embedded systems.

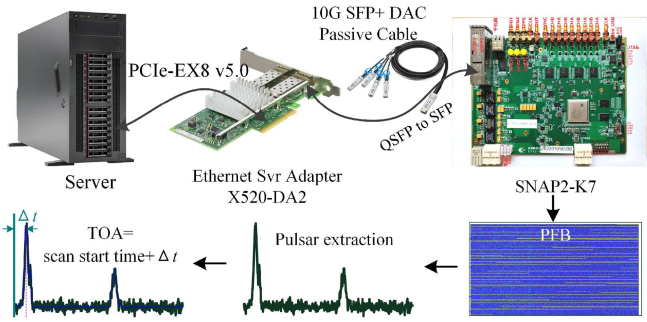


Fig. 1. The hardware framework of high-density pulsar data transmission and processing.

Taking into account the aforementioned stages, to facilitate real-time high-density pulsar data transmission and processing for grid synchronization, the speed of data processing is expected to surpass the speed of data transmission. Consequently, the pulses can be extracted from the pulsar data and applied to grid synchronization [29].

To address the aforementioned limitations, this paper introduces a real-time high-density pulsar data transmission and processing framework for grid synchronization. Building upon the analog front-end circuit and pulsar-based hardware framework established in [30], this research further enhances real-time performance as an extension of the IEEE IAS 2023 conference [31].

1) To prevent frame loss, a data transmission method is developed based on hardware-assisted software PF_Ring ZC. Additionally, a complementary duty cycle timing module is designed to coordinate the operation of the hardware and PF_Ring ZC, achieving a dual-channel 10 GbE/s speed.
2) To minimize operation time and attain real-time performance, a multi-threading approach is employed for pulsar data processing.
3) The pulsar signal extraction methods are introduced based on the polyphase filterbank (PFB) and fast folding. The efficacy of the extracted pulse profile is subsequently assessed through a meticulously designed cross-correlation evaluation.
4) A hardware platform for pulsar data sampling and transmission is designed. Comprehensive experiments under various hardware and test scenarios are conducted, with the results indicating that real-time high-density pulsar data transmission and processing can be achieved.

The structure of this paper is organized as below. The framework of the developed data transmission and processing is presented in Section II. The principle of the designed software

is introduced in Section III and IV. Then, the pulse signal extraction is introduced in Section V. Next, the experiments are conducted in Section VI. Finally, the conclusions are drawn in Section VII.

II. Framework of the Developed Data Transmission and Processing

To achieve real-time high-density pulsar data transmission and processing, the hardware and software scheme are developed, as demonstrated in Fig. 1, which can be segmented into three distinct stages

i) Data transmission based on PF_Ring ZC: High-density pulsar data, encompassing both the sampling data and polyphase filterbank data [32], are collected. Subsequently, this data is transferred via the PF_RingZC, utilizing the 10 GbE module.
ii) Implementation of multi-threading: Following this, the pulsar data transmitted from the 10 GbE module is stored in the data server. A multi-threaded program is implemented to parse the pulsar data, thereby accelerating the calculation process.
iii) Pulsar data processing: The pulsar data is processed and the pulse signal is extracted utilizing PFB, dedispersion, and folding technologies. Thereafter, the grid synchronized signal (or the pulse per second) can then be generated according to the estimated time of arrival based method, which involves comparing the standard profile with the extracted signal [33], [34].

Here, the second edition Smart Network ADC Processor (SNAP2) [35], [36] based on the Kintex-7 FPGA (K7) is selected, also called SNAP2-K7, as shown in Fig. 1. It contains three ADCs with a maximum sampling frequency of up to 500 MHz and two Quad Small Form-factor Pluggable (QSFP) connectors.

III. Data Transmission Based on PF_RingZC

To achieve high-density data transmission, a new type of network socket named PF_Ring technology is utilized [37]. It is available for Linux kernel with the advantage of improving the packet capture speed dramatically.

The commonly used high-speed packet capture libraries include the tcpdump, libpcap and PF_Ring. The tcpdump and libpcap are the command-line packet analysis and portable C/C++ libraries for network traffic capture, respectively [38]. The differences in the data packet transmission are demonstrated in Fig. 2. For the libpcap, it can be seen from Fig. 2 that the data packet will be saved in the data buffer, and then the user can capture the data. The data would be copied several times thus limiting its efficiency.

Compared to the libpcap, the data packet capture buffer is removed and therefore the data speed can be faster for the PF_Ring. Besides, the PF_Ring ZC is an extended version of PF_Ring, which can accelerate packet capture by means of dynamically loadable kernel plugins. As depicted in Fig. 2, the PF_Ring ZC can directly capture the data from the Direct Memory Access (DMA) without the need for an RX buffer.

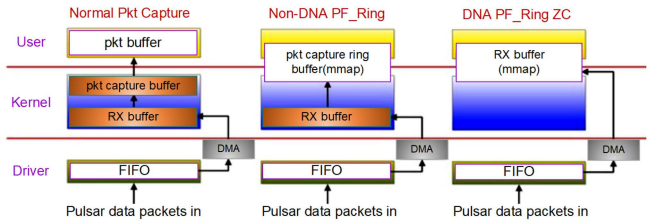


Fig. 2. Comparison for the libpcap, PF_Ring, and PF_Ring ZC, source: [39]. FIFO: First In, First Out, mmap: memory-mapped file I/O.

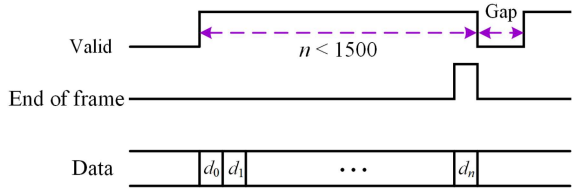


Fig. 3. The timing diagram, the $n < 1500$ denotes the number of the valid data frame does not exceed 1500.

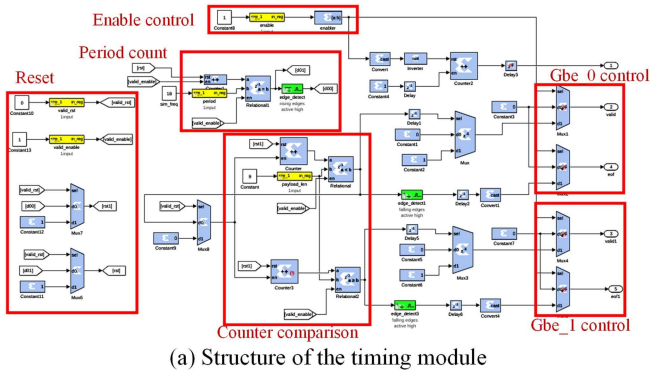
This technology is called direct network interface card access (DNA). The biggest advantage of DNA is that it can achieve zero-copy, which means that the CPU does not perform the data copy operation.

However, the data packet contains some parity bits, as shown in Fig. 3. It shows that there is a gap at the end of the frame, which indicates that the system can not guarantee the continuous transmission of data. The duty cycle can be calculated based on the length of the valid and gap, which can be expressed as valid/(valid + gap).

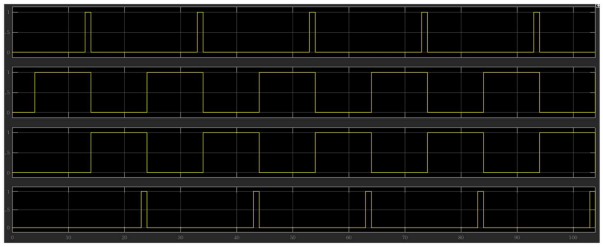
To this end, the dual-channel, hot-pluggable network interface, named the SFP+ ports is usually used in the hardware. To achieve the interfaces operating in coordination and continuous data transmission, a coordination mechanism is designed. The dual-channel ports are controlled to transfer data alternately. In this research, a complementary duty cycle timing module is developed, where the software and tested timing results are presented in Fig. 4.

In Fig. 4(a), the designed complementary duty cycle timing module consists of three components, including the period count, counter comparison, and reset. The software is implemented based on the Simulink design using both standard Xilinx system generator blockset, as well as library blocks specific to CASPER boards. As can be seen from the output of this timing module shown in Fig. 4(b), the output timing signals are completely complementary, indicating the effectiveness of the designed module.

The architecture of the programmed 10 GbE module is depicted in Fig. 5. The design encapsulates three distinct stages: gigabit network communication, complementary duty cycle timing module, and 10 gigabit Ethernet port. The Gigabit network communication is responsible for establishing communication with the SNAP2-K7 to realize real-time command control. Meanwhile, the 10 gigabit Ethernet port is utilized to encapsulate and transmit the data processed by the PFB.



(a) Structure of the timing module



(b) Timing change for Gbe_0 and Gbe_1

Fig. 4. The designed complementary duty cycle timing module and its tested timing results. In (a), the yellow part denotes the hardware configuration in the SNAP2-K7, the light blue denotes the basic operations such as the counter.

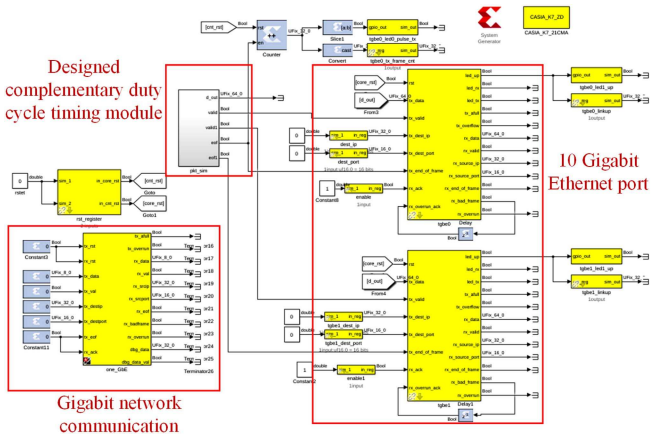


Fig. 5. The program of 10 GbE module based on Simulink.

Finally, the proceeding of the PF_RingZC can be summarized as

1) *Initialization:* The user receiving program is initialized first, a PF_RING socket is created, and waiting for the arrival of the data packet.
2) *Receiving:* The network card receives the data packet from SNAP2-K7. The hard interrupt is triggered, and the driver in the server copies the pulsar data into the ring_buffer allocated by the PF_RING socket through DMA technology.
3) *Saving:* The user program continuously writes the received data into the target address using $fwrite()$ in the $pfring\_loop()$ function, which is the C language function. Then the data will be saved on the hard disk.

Next, the data would be processed in the data server and the pulses can be extracted.

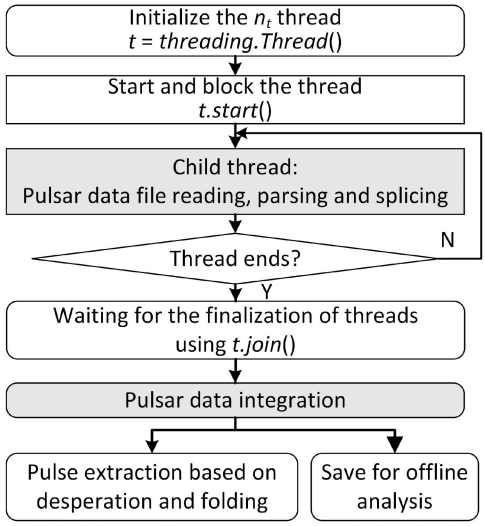


Fig. 6. The flowchart of the multi-threading for pulsar data processing.

IV. Real-Time Processing Based on Multi-Threading

If pulsar data streaming is not processed in a timely manner, it can result in a substantial accumulation of data. Traditionally, both multi-threading and multiprocessing are employed to expedite data processing. However, multiprocessing does not allow for data sharing between different process functions, which can slow down data access speed. To achieve real-time operation, multi-threading technology is designed in this section.

If pulsar data streaming is not processed in a timely manner, it can result in a substantial accumulation of data. Traditionally, both multi-threading and multiprocessing are employed to expedite data processing. The parallelism with multiple isolated processes is popular in Python but it may lead to inefficiency because the system needs to allocate separate memory space when a new process is created [40], which can slow down data access speed. For multi-threading, there is no need for additional memory allocation and environment copying, thus the overhead of creating a new thread is small. Importantly, it is worth mentioning that the pulsar data stream is a binary file requiring more I/O operations, where the global interpreter lock would be released during I/O operations and it also helps speed up the data parsing process. To achieve real-time operation, multi-threading technology is designed in this section.

The fundamental multi-threading process is depicted in Fig. 6. Utilizing the Python language, the number of multi-threading is set as n_t. Subsequently, the received data is processed in parallel before being integrated. The library named 'threading' is utilized in this research [41].

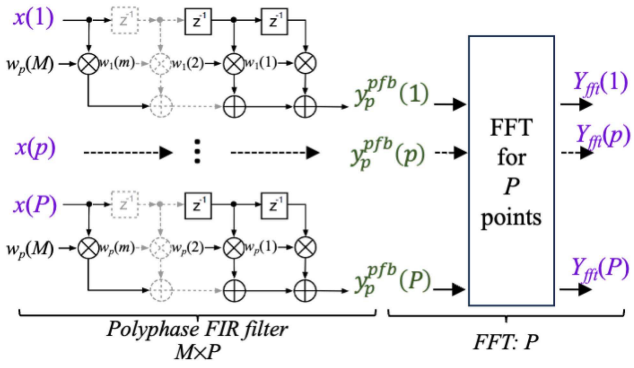


Fig. 7. Implementation of PFB for pulsar data streaming.

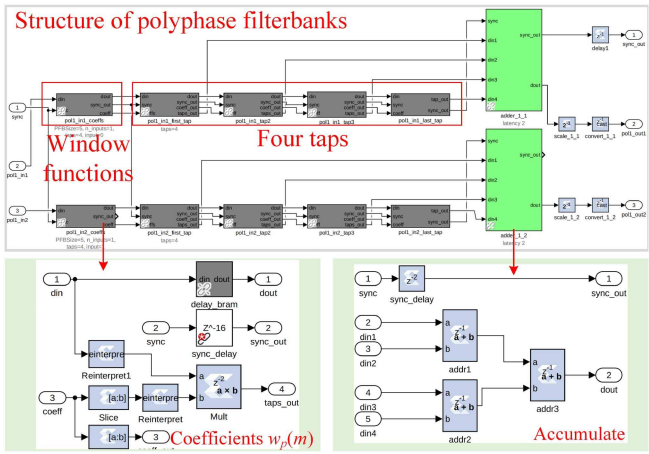


Fig. 8. Implementation of polyphase filterbanks based on Simulink.

V. Pulsar Signal Extraction and Synchronized Signal Generation

A. Pulsar Signal Extraction Based on PFB

The polyphase filterbanks have the advantage of low computational complexity in processing the high-sampling radiotelescope data.

Denoted received pulsar data as $x(n)$, the PFB consists of two steps to calculate its output, including the polyphase finite impulse response (FIR) filter and fast Fourier transform (FFT). The motivation for deploying a polyphase FIR filter is to enhance the power spectrum response as well as decrease the calculation time.

In the first step, the input data $x(n)$ will be decomposed into a set of P sub-sequences. This process is achieved through a decimating lowpass polyphase filter, which can be expressed as

$$y_p^{pfb}(n') = \sum_{m=0}^{M-1} w_p(m) x_p(n' - m) \tag{1}$$

where $w_p(m)$ is the coefficients from the prototype lowpass filter M denotes the number of polyphase taps in the filter, $x_p(n)$ denotes the Pth branches derived from $x(n)$, $p = 1, 2, \ldots, P$. The window function can be derived from the window function, where the Hanning window function is selected in this research.

Therefore, each branch of $y_p(n')$ will be fed into the FFT to calculate the spectrum of the pulsar signal, we can get the FFT results

$$\begin{aligned} Y_{fft}(k, n') &= \sum_{p=0}^{P-1} y_p^{pfb}(n') e^{-2\pi i k p/P} \\ &= \sum_{p=0}^{P-1} \sum_{m=0}^{M-1} \left[w_p(m) e^{-2\pi i k p/P} \right] x_p(n' - m). \end{aligned} \tag{2}$$

As demonstrated in Fig. 7, the PFB for processing the pulsar data streaming is divided into two parts, where $M \times P$ points are used in the first part. Based on the above two equations, the sampling rate of the input $x(n)$ is downsampled into the Mth dataset with a length of P. To accelerate the calculation, the FFT is programmed in the FPGA as a part of PFB because it has a lower time complexity.

The designed program of polyphase filterbanks with four taps is demonstrated in Fig. 8. It shows that the window coefficients $w_p(m)$ are saved in the memory and then the multiple with the data from taps. Therefore, the output of four taps is accumulated through 3 adders. Finally, the output of polyphase filterbanks will be fed into FFT.

B. Complexity Analysis of PFB

The complexity of PFB is analyzed in this subsection. For the first part of PFB, a buffer of size $M \times P$ is necessitated to store the filter coefficients. Considering the real-time characteristics, only $2P$ points deserve to be saved, one for data saving and another for sum calculation. The complexity of the polyphase FIR filter is $O(log(MN))$, which simplifies to approximately $O(log(N))$, where N is the length of the data. The FFT exhibits a complexity of $O(Nlog(N))$. Besides, the memory size and bit length are also changed with the multiplication and addition operations.

C. Folding and TOA Estimation for Grid Synchronization

Caused by the vacuum media interference, the delaying effect will be reflected in different frequency channels. The low-frequency areas will be delayed more than the high-frequency band. Therefore, the incoherent dedispersion is deployed to calibrate the results of different frequency channels of PFB.

The observed time delay in each channel can be expressed as

$$\Delta T(f) \equiv \frac{k_{\mathrm{DM}}}{\Delta\tau} \left(\frac{1}{(f_0 + f\Delta f)^2} - \frac{1}{f_0^2} \right) \tag{3}$$

where the $k_{\mathrm{DM}} = 4.148808 \times 10^3\ \mathrm{MHz}^2\, pc^{-1}\mathrm{cm}^3\,\mathrm{s}$ is the dispersion constant. $\Delta\tau$ is the sampling interval between each sampling point. f denotes the frequency of each channel, f_0 and Δf are the start frequency of the band and channel frequency differences with the MHz unit, respectively.

Actually, the value of $\Delta T(f)$ is a decimal. Thus, the round function is utilized to calculate the time delay as

$$\Delta t(f) \equiv \mathrm{round}(\mathrm{DM}\Delta T(f)) \tag{4}$$

TABLE I
CONFIGURATION OF THE DATA SERVER AND LAPTOP

Hardware	CPU configuration			
	Model	Main frequency	Total cores & threads	Memory
Laptop	i5-9400	2.9 GHz	6 cores/6 threads	16 GB
Server	i9-12900	5.1 GHz	16 cores/24 threads	64 GB

where the $DM \equiv \int n_e dL$ denotes the dispersion measure and it is determined by the distance L to the observed pulsars and the electron number density n_e.

After that, the time series data is channeled into the folding steps for pulsar extraction. At this juncture, the folding period is approximated and is expected to fall within the interval $[mt_{bin}, (m+1)t_{bin}]$, and t_{bin} denotes the equivalent time per point after incoherent dedispersion. To ensure precise folding, the accumulated residual error, denoted by $\sigma = t_{bin} - f_p$, is meticulously tracked for the next folding, where f_p denotes the pulsar period.

Additionally, the improved fast folding algorithm (IFFA) will also employed to estimate the period of puslars, where the IFFA was originally designed by Stealin and Song [42].

Next, after extracting the temporal pulse profile, the time of arrival of the observation will be determined. The fundamental concept involves comparing the observation profile with a template, which is typically provided by long-term observations conducted through large radio telescopes. By establishing an estimation method for TOA, its uncertainty can be calculated using Fourier phase gradient and Fourier domain technologies, in conjunction with Markov chain Monte Carlo methodologies [13].

Ultimately, the frequency of the local crystal oscillator will be estimated by establishing the relationship between the sampling data and the TOA value.

VI. EXPERIMENTS

To verify the real-time performance of the designed framework, a series of experiments have been designed and conducted. The programming development environment is based on MATLAB 2016b and Vivado 2016.04 under Ubuntu 16.04. For hardware interfacing, the 10 G SFP+ DAC passive cable is selected to connect the SNAP2-K7 and Ethernet adapter. Furthermore, considering that the quad SFP (QSFP) is configured in SNAP2-K7, the cable converter is selected to realize QSFP to SFP interface conversion so that the 40 G speed can be converted to 10 G, as illustrated in Fig. 1.

In the tests, the operating frequency of the 10GbE module is 156.25 MHz, and the main frequency of FPGA is set to 100 MHz. To compare the speed, a laptop is also used. The configuration of the data server and the laptop are listed in Table I.

A. Transmission Rate Under Different Duty Cycles

To verify the transmission rate under different duty cycles and module clocks, the results are carried out as listed in Table II. In this Table, the frequency refers to the clock of the Gigabit transceiver reference clock. And a relatively low frequency is to explore the progressive relationship of the transfer speed before reaching the maximum rate.

TABLE II
TRANSMISSION RATE UNDER DIFFERENT DUTY CYCLE FOR DATA SERVER

Frequency	Duty cycle			
	10%	25%	50%	75%
60 MHz	0.77 GB/s	1.92 GB/s	3.84 GB/s	5.76 GB/s
100 MHz	1.3 GB/s	3.2 GB/s	6.42 GB/s	9.6 GB/s

It reveals that the transmission rate increases with the increase of the duty cycle. Meanwhile, the module clock exerts an impact on the transmission rate, with higher frequencies yielding increased rates. Importantly, the transmission rate can reach 6.42 G/s which is sufficient to receive the data from the SNAP2-K7 when utilizing dual SFP ports. This also means that the pulsar data would not be dropped under this configuration.

TABLE III
THE RUNNING TIME USING MULTIPROCESSING FOR 4 GB PULSAR DATA

Tasks	Number of cores (second): data parsing/integration			
	2	4	8	16
32	0.0124/2.362	0.0137/1.463	0.0182/2.034	0.0338/4.651
64	0.0119/2.101	0.0178/1.534	0.0231/4.405	0.0367/6.197
128	0.0131/2.109	0.0215/1.742	0.0404/6.441	0.0581/8.138

B. Comparison for Multi-Threading and Multiprocessing

Next, to further verify the real-time performance, both the multi-threading and multiprocessing are tested. The duration of data parsing and integration under multiprocessing is listed in Table III.

The data presented in Table III demonstrates that the time increases when the number of cores increases. Besides, a greater number of tasks correlates with increased time demands. For example, only 2.362 s is consumed, and this value escalates to 8.138 s with a greater task load, even when additional cores are employed. The primary reason could be that when a new process is created, the operating system needs to allocate independent memory space for it and copy the execution environment of the parent process, which is relatively time-consuming. Overall, the results reveal that there are more obstacles to meeting real-time requirements for multiprocessing in processing the pulsar data.

To assess real-time performance further, the time consumption via multi-threading is tested, as demonstrated in Fig. 9. A 4 GB data is collected which corresponds to a sampling time of approximately 4.8 s. When the $n_t = 1$, it means that the multi-threading is not activated. It can be seen that as the number of threads increases, the time decreases to 1.493 s on the laptop. For the data server, the time increases slightly with more cores. The reason could be that the server is much faster than the laptop, and the server reaches an inflection point where additional threads no longer equate to performance gains. If the data is processed without multi-threading, 4 GB data will

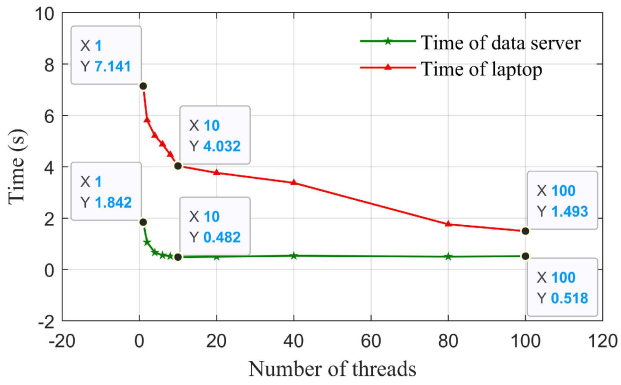


Fig. 9. The running time using multi-threading for 4 GB pulsar data.

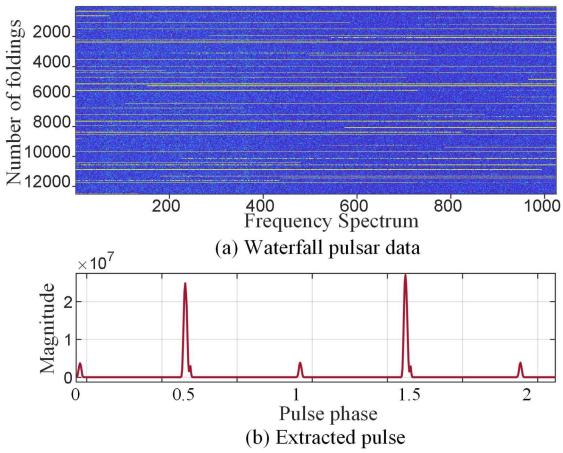


Fig. 10. The waterfall profile and the extracted pulses for pulsar J1939+2134.

consume 1.842 s for the data server. The lowest times are 0.482 s and 1.493 s for the data server and laptop when the number of cores is 100 and 10, respectively. The result demonstrates that the speed of processing data is increased by more than 3.8 times (1.842/0.482), indicating that the data is processed faster than the data is collected.

One illustrative case of pulse extraction is depicted in Fig. 10. It demonstrates that the pulses can be extracted successfully, where their profile is the same as the pulsar J1939+2134.[1] Utilizing this profile, the timing information can be connected with the Coordinated Universal Time. Then the pulse profile can be converted into the TOA that the grid synchronization measurement device can be refereed.

C. Verification of the PFB and Processing Time

To investigate the effectiveness of the PFB and the folding performance under different volumes of pulsar data, the extracted PFB and pulse profiles are carried out, as illustrated in Figs. 11 and 12, respectively. The tests are conducted based on the analog front-end circuit in the previous research [30], where the standard pulse profile suppressed full frequency range noise

[1]Source: http://www.epta.eu.org/epndb/

TABLE IV
THE RUNNING TIME USING PFB AND FOLDING PULSAR DATA

Volumes	Calculation time (ms)		
	PFB	Folding	IFFA Folding
1 GB	3334.41	54 ± 0.61	158± 3.42
2 GB	6671.02	111 ± 1.42	307±2.16
4 GB	13103.27	225 ± 1.23	610±2.76
8 GB	46745.75	442 ± 2.13	1221± 4.43
10 GB	83036.25	548 ± 3.74	1510± 7.86

is fed into the circuit. The pulse profile is referred to as the PSR B1937+21, which is a double pulsar system. Additionally, there is no need to perform dispersion as the analog signal in this instance does not contain any time delay.

As demonstrated in Fig. 11, the magnitude of the waterfall profile of pulsars undergoes random changes due to the superimposed heavy noise. As the sampling time increases, the folding times increase from Fig. 11(a) to (d). It is noteworthy that the magnitude ratio of the pulsar and noise is determined by the circuit voltage in this scenario. It is anticipated that the noise will neutralize itself over time.

Subsequently, the profiles of the pulsar are extracted and folded based on the PFB waterfall results, with all profiles being normalized. It is noteworthy that, despite the phase shift phenomena in Fig. 12, which is brought on by the observation time difference, the profile of the standard and observed pulses remains similar. It can be visualized from Fig. 12(a) that the observed pulsar signal contains a significant amount of noise in comparison to the standard profile. However, the pulse profile becomes more distinct as the number of folding times increases, as compared with Fig. 12(d). Importantly, the noise is eliminated with a longer time window, which aligns with the results presented in Fig. 12.

To quantitatively evaluate the performance of the extracted pulses, the normalized cross-correlation index is calculated, as demonstrated in Fig. 13. It reveals that the red line has the highest value, indicating that a long-time folding contributes to the improvement of the profile. Meanwhile, it can be seen from Fig. 13 that the noise will lead to increased cross-correlation when the phase is near 1000. This effect arises because the noise obscures the amplitude discrepancy between the pulse and noise. Consequently, a longer window size will produce a clearer profile with differential correlation indicators.

D. Performance With Different Volumes of Pulsar Data

To authenticate the real-time performance of the PFB and folding parts in Section IV, the time consumed for pulsar profile extraction is evaluated based on the server configuration delineated in Table I. The generated folding and improved folding method IFFA [42] are compared, with the results tabulated in Table IV. The uncertainty of folding is also presented.

It is evident that the time consumption increases with the volume of data. Despite the PFB calculation requiring nearly 3.3 seconds for 1 GB of data, this step can be integrated into the

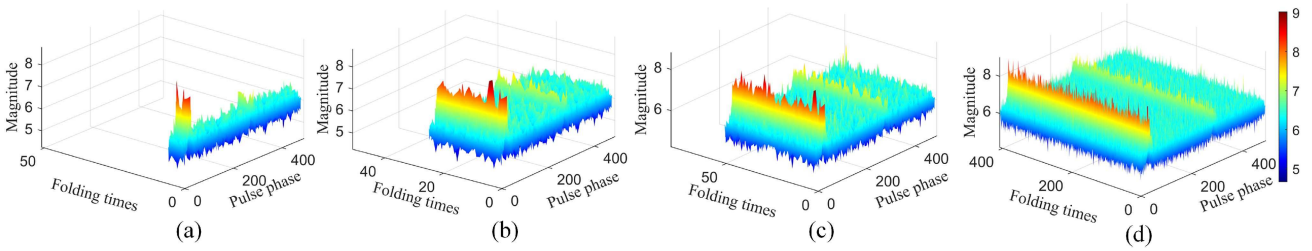


Fig. 11. The waterfall profile for pulsar PSR B1937+21 under different window sizes, (a) 0.003125 s window size, (b) 0.0125 s window size, (c) 0.05 s window size, (d) 2 s window size.

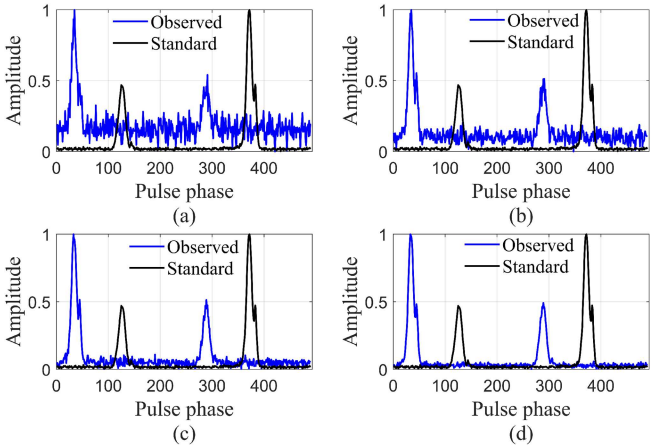


Fig. 12. Extracted pulses for PSR B1937+21 under different numbers of folding times. (a) 10 folding times for 0.003125 s data, (b) 30 folding times for 0.0125 s data, (c) 51 folding times for 0.05 s data, and (d) 408 folding times for 2 s data.

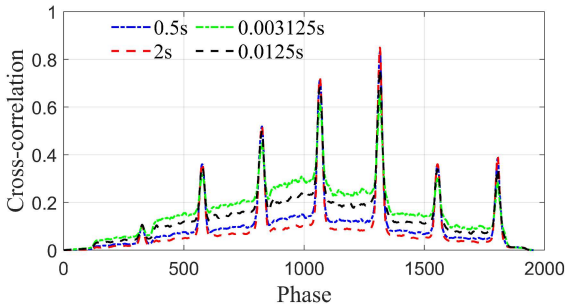


Fig. 13. The normalized cross-correlation under different numbers of folding times.

FPGA to facilitate real-time computation. Additionally, the folding time is linearly proportional to the data volume. Compared with the generated folding and IFFA, it is observed that IFFA consumes more time due to additional shifts and operations. The uncertainty results also indicate that larger data volumes result in higher uncertainties. Overall, it consumes about 482 ms for data receiving and 225 ± 1.23 ms to fold the pulse profile for 4 GB volume data, which corresponds to the 4.8 s original data, indicating the real-time performance can be satisfied.

VII. Conclusion

To facilitate real-time high-density pulsar data transmission and processing, a hardware and software framework is developed to promote the application of pulsars in grid synchronization. The data transmission technology based on PF Ring ZC is used. Besides, a complementary duty cycle timing module is designed to coordinate the dual 10GbE module ports. The results of transmission rate under different duty cycles demonstrate that 9.6 GB/s speed can be achieved, even at a 75% duty cycle. Then, the multi-threading is applied to the pulsar data processing, including reading, parsing, and splicing. The tested time based on the laptop and server reveals that it will consume 1.493 s and 0.482 s for 4 GB data, respectively. The developed framework enables real-time high-density pulsar data processing, significantly reducing the time required for pulsar signal extraction and TOA transformation. Moreover, the folding experiments are carried out, where the results demonstrate that approximately 707 ms will be consumed for 4 GB volume pulsar signal measurements when the PFB is performed in the hardware. This indicates that a low time delay can be achieved for grid synchronization.

Acknowledgment

The authors would like to thank the FNET team members, particularly Dr. Liang Zhang, Dr. Xiqian Luo, and Mr. Yuru Wu at the University of Tennessee, Knoxville for their early contribution to developing the hardware and software module for pulsar data processing.

References

[1] L. A. Rawley, J. H. Taylor, M. M. Davis, and D. W. Allan, "Millisecond pulsar psr 1937 21: A highly stable clock," *Science*, vol. 238, no. 4828, pp. 761–765, 1987. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.238.4828.761

[2] J. Singha et al., "Evidence for profile changes in PSR J1713 0747 using the uGMRT," *Monthly Notices Roy. Astronomical Soc.: Lett.*, vol. 507, no. 1, pp. L57–L61, Aug. 2021. [Online]. Available: https://doi.org/10.1093/mnrasl/slab098

[3] *IEEE Standard for Synchrophasor Measurements for Power Systems*,IEEE Standard C37.118.1, 2011.

[4] W. Yao et al., "Impact of GPS signal loss and its mitigation in power system synchronized measurement devices," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 1141–1149, Mar. 2018.

[5] W. Qiu et al., "Pulsar-calibrated timing source for synchronized sampling," *IEEE Trans. Smart Grid*, vol. 13, no. 2, pp. 1654–1657, Mar. 2022.

[6] D. R. Lorimer and M. Kramer, *Handbook of Pulsar Astronomy*, vol. 4. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[7] P. Bolli et al., "An international survey of front-end receivers and observing performance of telescopes for radio astronomy," *Pub. Astronomical Soc. Pacific*, vol. 131, no. 1002, Jul. 2019, Art. no. 085002.
[8] R. Finger, P. Mena, N. Reyes, R. Rodriguez, and L. Bronfman, "A calibrated digital sideband separating spectrometer for radio astronomy applications," *Pub. Astronomical Soc. Pacific*, vol. 125, no. 925, pp. 263–269, Feb. 2013.
[9] T. Dolch et al., "A 24 HR global campaign to assess precision timing of the millisecond pulsar j17130747," *Astrophysical J.*, vol. 794, no. 1, p. 21, Sep. 2014.
[10] T. Ergesh, J. Li, X.-F. Duan, X. Pei, and Z.-G. Wen, "Development of pulsar digital backend based on rfsoc," *Res. Astron. Astrophys.*, vol. 22, no. 2, Feb. 2022, Art.no. 025002. [Online]. Available: https://dx.doi.org/10.1088/1674-4527/ac3ad9
[11] CASPER, "ROACH," 2013. Accessed: Jul. 2024. [Online]. Available:https://casper.astro.berkeley.edu/wiki/ROACH
[12] J.-W. Mao et al., "Targeted search for fast radio bursts with nanshan 26m radio telescope," *Res. Astron. Astrophys.*, vol. 22, no. 6, May 2022, Art. no. 065006. [Online]. Available: https://dx.doi.org/10.1088/1674-4527/ac6797
[13] J. Wang et al., "A comparative analysis of pulse time-of-arrival creation methods," *Astron. Astrophys.*, vol. 658, 2022, Art. no.A181.
[14] CASPER Tutorials, [Online] available: https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/index.html
[15] J. Hickish et al., "Decade of developing radio-astronomy instrumentation using casper open-source technology," *J. Astronomical Instrum.*, 2016. [Online]. Available: https://www.worldscientific.com/doi/10.1142/S2251171716410014
[16] Hardware, 2018. [Online]. Available: https://casper.astro.berkeley.edu/wiki/Hardware
[17] D. C. Price et al., "The breakthrough listen search for intelligent life: Wide-bandwidth digital instrumentation for the CSIRO Parkes 64-m telescope," *Pub. Astronomical Soc. Aust.*, vol. 35, 2018, Art. no.e041.
[18] X. Zhang et al., "The design of China reconfigurable analog-digital backend for fast," *Res. Astron. Astrophys.*, vol. 20, no. 5, pp. 7301–7310, 2020.
[19] I. Garvanov, M. Garvanova, and H. Kabakchiev, "Pulsar signal detection and recognition," in *Proc. 8th Int. Conf. Telecommun. Remote Sens.*,2019, pp. 30–34.
[20] I. Garvanov, C. Kabakchiev, V. Behar, and M. Garvanova, "The experimental study of possibility for pulsar signal detection," in *Proc. IEEE Int. Conf. Eng. Telecommun.*, 2016, pp. 68–71.
[21] M. Cranmer et al., "Bifrost: A python/c framework for high-throughput stream processing in astronomy," *J. Astronomical Instrum.*, vol. 06, Sep. 2017, Art. no. 1750007.
[22] Z. Yazhou et al., "PSRDP: A parallel processing method for pulsar baseband data," *Res. Astron. Astrophys.*, vol. 24, 2023, Art. no. 015025. [Online]. Available: http://iopscience.iop.org/article/10.1088/1674-4527/ad0e99
[23] W. van Straten, P. Demorest, and S. Osłowski, "Pulsar data analysis with psrchive," 2012. [Online]. Available: https://arxiv.org/abs/1205.6276
[24] G. B. Hobbs, R. T. Edwards, and R. N. Manchester, "Tempo2, a new pulsar-timing package–I. An overview," *Monthly Notices Roy. Astronomical Soc.*, vol. 369, no. 2, pp. 655–672, May 2006. [Online]. Available: https://doi.org/10.1111/j.1365-2966.2006.10302.x
[25] H. Zhang, M. Demleitner, J. Wang, N. Wang, J. Nie, and J. Yuan., "The xinjiang astronomical observatory NSRT pulsar data archive," *Adv. Astron.*, vol. 2019, Jan., 2019, Art. no. 5712682.
[26] P. van Gelderen, J. Duyn, N. Ramsey, G. Liu, and C. Moonen, "The PRESTO technique for FMRI," *NeuroImage*, vol. 62, no. 2, pp. 676–681, 2012,. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053811912000341
[27] EPTA Collaboration, J. Antoniadis . et al., "The second data release from the European pulsar timing array," *Astron. Astrophys.*, vol. 678, pp. 1–29, 2023.
[28] K. Liu et al., "Measuring pulse times of arrival from broad-band pulsar observations," *Monthly Notices Roy. Astronomical Soc.*, vol. 443, no. 4, pp. 3752–3760, Aug. 2014. [Online]. Available: https://doi.org/10.1093/mnras/stu1420
[29] H. Yin et al., "Pulsar based alternative timing source for grid synchronization and operation," *IEEE Access*, vol. 8, pp. 147818–147826, 2020.
[30] W. Qiu et al., "Analog front-end: Circuit of pulsar-based timing synchronization for the WAMS," *IEEE Trans. Ind. Appl.*, vol. 58, no. 2, pp. 1622–1631, Mar./Apr., 2022.
[31] W. Qiu et al., "Development of real-time high-density pulsar data transmission and processing for grid synchronization," in *Proc. IEEE Ind. Appl. Soc. Annu. Meeting*, 2023, pp. 1–5.
[32] X. Luo, H. Yin, W. Qiu, L. Zhang, and Y. Liu, "Precise timing based on pulsar observation for grid synchronization," in *Proc. IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf.*, 2021, pp. 1–5.
[33] P.-T. Chen, J. L. Speyer, and W. A. Majid, "Experimental verification of a pulsarbased positioning system using L-band measurements," *J. Guid., Control, Dyn.*, vol. 43, no. 1, pp. 60–72, 2020. [Online]. Available: https://doi.org/10.2514/1.G003722
[34] L. Sheng-chang, Z. Xu, P. Rui, and L. Jiong-hui, "A method to calibrate the on-board timing based on pulsar observation," in *Proc. 6th IEEE Conf. Ind. Electron. Appl.*, 2011, pp. 2265–2269.
[35] CASPER, "SNAP2," 2016. Accessed: Jul. 2024. [Online]. Available: https://casper.astro.berkeley.edu/wiki/SNAP2
[36] J. Hao et al., "The design of snap2 system, nstitute of automation," *Chin. Acad. Sci.*, 2020. [Online]. Available:https://casper.astro.berkeley.edu/wiki/images/6/62/SNAP2_Doc.pdf
[37] "Pf_ring, high-speed packet capture, filtering and analysis," 2022. [online], available: https://www.ntop.org/products/packe-capture/pf_ring/
[38] V. Duarte and N. Farruca, "Using libPcap for monitoring distributed applications," in *Proc. Int. Conf. High Perform. Comput. Simul.*, 2010, pp. 92–97.
[39] whatday, "PF_ring introduction," 2022. [Online]. Available: https://blog.csdn.net/whatday/article/details/89242246
[40] A. Malakhov, "Composable multi-threading for python libraries," in *Proc. SciPy*, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:11858935
[41] Miss Islington, "Threading – thread-based parallelism," 2022. [Online], Available: https://docs.python.org/3/library/threading.html
[42] M. Song et al., "Fast period estimation of X-ray pulsar signals using an improved fast folding algorithm," *Chin. J. Aeronaut.*, vol. 36, no. 10, pp. 309–316, 2023.