CUDA-Based Polyphase Filter Bank Design for Radio Astronomy: Postprint

Authors: Tohtinur, Zhang Hailong, Wang Jie

Date: 2017-09-26T00:00:00+00:00

Abstract

A radio astronomy polyphase filter bank was designed utilizing graphics processing units and the latest general-purpose parallel computing architecture, with its performance metrics subsequently tested and analyzed. By exploiting the powerful floating-point computation and efficient parallel execution capabilities of graphics processing units, acceleration of both polyphase filter and Fast Fourier Transform algorithms was realized, thereby improving the execution efficiency of the polyphase filter bank algorithm. Experimental results demonstrate that the designed polyphase filter bank possesses flexibility and scalability, enabling high-speed filtering and channelization of radio signals, and can effectively enhance the parallel data processing capability and computational efficiency of digital backend algorithms for radio telescopes.

Full Text

Design of a Polyphase Filter Bank for Radio Astronomy Based on CUDA

Tohtinur[1], Zhang Hailong[1,2], Wang Jie[1]

[1]Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi, Xinjiang 830011, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]Key Laboratory of Radio Astronomy, Chinese Academy of Sciences, Nanjing, Jiangsu 210008, China

Abstract: This paper presents the design of a polyphase filter bank for radio astronomy using Graphics Processing Units (GPUs) and the latest NVIDIA CUDA parallel computing architecture, with comprehensive testing and analysis of its performance metrics. By leveraging the GPU' s powerful floating-point computation capabilities and highly efficient parallel execution, we achieve significant

acceleration of both the polyphase filter and Fast Fourier Transform (FFT) algorithms, substantially improving the execution efficiency of the polyphase filter bank. Experimental results demonstrate that the designed polyphase filter bank offers considerable flexibility and scalability, enabling high-speed filtering and channelization of radio signals while effectively enhancing the parallel data processing capabilities and computational efficiency of digital backend algorithms for radio telescopes.

Keywords: CUDA; GPU; Polyphase Filter Banks; Parallel Data Processing

1. Introduction

Traditional CPU-based polyphase filter bank designs suffer from long simulation times due to algorithmic complexity and computational demands. The Compute Unified Device Architecture (CUDA) provides an effective solution to the low efficiency of polyphase filter bank algorithms in high-speed data processing by harnessing the GPU' s efficient parallel execution capabilities. CUDA is a parallel programming model developed by NVIDIA that accelerates high-performance parallel processing on GPUs, offering an effective approach to solving massive-scale computations with simple logic. GPU computational capabilities have advanced at a rate exceeding Moore' s Law, encompassing instruction set architecture and parallel computing engines within the GPU. Modern GPUs demonstrate clear advantages over CPUs in both single-precision floating-point processing power and external memory bandwidth. CUDA has achieved significant improvements in programming and optimization, substantially enhancing the general-purpose computing capabilities of GPUs. With the development of parallel processing technology, GPUs have become the preferred platform for real-time astronomical signal processing.

Polyphase filter banks play a crucial role in radio telescope digital backends, performing channelization through polyphase filtering while effectively controlling spectral leakage generated by FFT. As an efficient implementation form of digital filter banks, polyphase filtering technology has been widely adopted in radio telescope backend development, including pulsar backends, dedispersion systems, correlators, and digital spectrometers. The combination of GPUs and CUDA architecture provides a high-speed implementation pathway for multi-channel radio astronomy polyphase filter bank design.

2. Principles of Polyphase Filter Banks

Radio astronomy polyphase filter banks consist of multiple decomposed Finite Impulse Response (FIR) filters and FFT components. The processing involves spectral shifting, filtering, and FFT operations. Decomposed FIR filters offer

numerous advantages compared to other digital filter types, including the ability to achieve strict linear phase and arbitrary amplitude characteristics. The fundamental principle of polyphase filter banks involves decomposing a digital filter' s impulse response function into multiple phase components for processing.

The transfer function of an FIR filter can be expressed as:

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$$

where N is the filter length. This can be decomposed into D groups:

$$H(z) = \sum_{k=0}^{D-1} z^{-k} E_k(z^D)$$

with $E_k(z^D) = \sum_{n=0}^{Q-1} h(nD+k) z^{-nD}$, where $Q = N/D$ is an integer and $k = 0, 1, 2, ..., D-1$.

Using the Discrete Fourier Transform (DFT), the decomposed FIR filter is processed to obtain the spectral response of the input signal. The DFT of the input signal $x[n]$ is:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

The original signal can be reconstructed from the spectral coefficients. When performing FFT processing on signals, both the time and frequency domains are discrete and finite-length, requiring sampling of analog signals and truncation in time. Since digital backend processing operates on finite time segments while FFT theoretically requires integration from negative to positive infinity, practical signal processing involves periodic extension of the signal for FFT analysis.

---

3. Design and Performance Testing

3.1 Implementation Based on CUDA

The implementation flow of the CUDA-based radio astronomy polyphase filter bank is illustrated in [Figure 2: see original paper]. The CPU and GPU coordinate to complete data processing tasks, with the CPU handling logic control and serial-related operations while the GPU manages highly parallel computations. Initially, the CPU completes initialization and prepares data for processing. The `cudaMalloc()` function creates GPU memory space, and `cudaMemcpy()`

transfers data from CPU memory to GPU memory. The kernel function then performs parallel algorithm processing.

To accelerate the algorithm, polyphase filter bank coefficients based on window functions are pre-generated on the CPU and transferred to GPU shared memory. The data is multiplied with polyphase filter coefficients to implement FIR filtering, followed by FFT processing. Finally, results are copied from GPU memory back to CPU memory for display.

The design utilizes the cuFFT library, which provides a simple programming interface for flexible data layouts and supports both single and double-precision transforms for complex and real-valued one-dimensional, two-dimensional, and three-dimensional transforms. The `cufftExecC2C()` function enables high-speed parallel FFT implementation, with speed improvements of up to 128× for one-dimensional transforms.

To reduce spectral leakage during FFT processing, window functions are applied. [Figure 3: see original paper] shows the impulse response and frequency response of an FIR filter based on the Hamming window.

3.2 Experimental Setup

The experimental and testing environment is detailed in . The hardware and software configuration includes an Intel Xeon E5 CPU, NVIDIA Quadro K620 GPU, Ubuntu 14.04 operating system, and Nsight Eclipse development environment.

** GPU and CUDA Parameters**

Parameter	Specification
Device	Quadro K620
CUDA Driver Version / Runtime Version	7.5 / 7.5
Total amount of global memory	2,047 Mbytes (2,146,752,752 bytes)
CUDA Cores	384 CUDA Cores
Memory Bus Width	128-bit
Warp size	32
Maximum number of threads per block	1,024
Maximum dimension size of a thread block (x, y, z)	(1,024, 1,024, 64)
Maximum dimension size of a grid size (x, y, z)	(2,147,483,647, 65,535, 65,535)
Total amount of shared memory per block	49,152 bytes

To validate the performance of the CUDA-based radio astronomy polyphase

filter bank, throughput and data processing time were tested and analyzed for various channel configurations. The test input consisted of 32 MB of 8-bit polarized complex noise signals at 128 MHz sampling rate. Data was transferred from host memory to GPU memory, converted to single-precision floating-point format, and processed through the polyphase filter bank and FFT operations. The output data underwent square-root operations to obtain power spectra, which were then transferred back from GPU to host memory.

---

4. Experimental Results

The experimental results are presented in [Figure 6: see original paper] through [Figure 9: see original paper]. [Figure 6: see original paper] and [Figure 7: see original paper] show the spectral outputs for polyphase filter banks with 1,024 channels and 1,048,576 channels respectively. While noise interference affects the output, the energy spectrum resolution improves with increasing channel count, enhancing detection capability.

[Figure 8: see original paper] illustrates the relationship between channel number and throughput. The throughput increases with channel count, reaching a maximum at 16,384 channels before declining.

[Figure 9: see original paper] depicts data processing time versus channel number. The primary time consumption occurs in three aspects: data transfer, polyphase filtering, and FFT operations. For 1,024 channels, the processing times are approximately 308 ms, 7 ms, and 13 ms respectively. As channel count increases from 1K to 16K, the average data processing time shows a decreasing trend, with the rate of reduction slowing as channel numbers grow further. The data transfer from GPU memory to CPU memory constitutes a significant portion of the processing time.

---

5. Conclusion

Experimental and testing results demonstrate that the CUDA-based radio astronomy polyphase filter bank effectively meets the channelization and rapid processing requirements of digital backends. CUDA technology successfully accelerates the algorithm, with the polyphase filtering and FFT operations achieving high parallelization. The design significantly improves real-time processing performance by leveraging GPU multi-threading capabilities, enhancing parallel processing speed for millions of channels. The implemented polyphase filter bank is easily extensible and upgradeable, with CUDA acceleration substantially improving computational efficiency for filter and FFT algorithms. Throughput, processing time, and power spectra across different channel configurations have been thoroughly tested and optimized.

References

[1] Garland M, et al. Scalable parallel programming with CUDA. Queue, 2008: 40-53.
[2] Nickolls J, Buck I, Ryoo S, et al. Optimization principles and application evaluation of a multithreaded GPU using CUDA. Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2008: 73-82.
[3] Zhao X, Jin C, Zhu Y, et al. Design of multi-channel digital backend based on polyphase filter. Astronomical Research & Technology, 495-502.
[4] Zhu K, Gan H, Zhu Y, et al. Performance discussion of polyphase filter bank spectral analysis method and its application in radio astronomy. Astronomical Research & Technology—Publications of National Astronomical Observatories of China, 81-90.
[5] Chen Y, Guo P. Matlab-aided DSP approach to FIR digital filter. Light Industry Science and Technology, 57-58.
[6] Zhao R, Yan L, Guo J. Design of real-time frequency spectrum analyzer for HF signals. Electronic Measurement Technology, 9-10.

Note: Figure translations are in progress. See original paper for figures.