第 41 卷第 5 期　　遥　测　遥　控　　Vol. 41, No. 5
2020 年 9 月　　Journal of Telemetry, Tracking and Command　　September 2020

Email:ycyk704@163.com　TEL:010-68382327　010-68382557

# 基于 GPU 的干涉测量 VDIF 格式数据编帧方法*

陈永强，马　宏，焦义文，刘燕都
（航天工程大学电子与光学工程系　北京　101416）

**摘要：**为提高深空测控网 VLBI 基带转换器灵活性和多种数据格式适应能力，设计一种基于 GPU 的 VDIF 格式数据编帧方法。基于 CUDA 并行运算架构开发符合 VDIF 规范的帧头计算模块、多通道单线程编帧模块和多通道多线程编帧模块。为了实现编帧过程效率的优化，设计基于纹理缓存查找表的数据量化方法和基于流式架构的多帧信号异步编帧方法。试验验证表明，设计的编帧方法结果正确，高效可靠，有望为我国深空测控干涉测量数据记录系统参与国际 VLBI 联合观测提供有效的支持。

**关键词：**VDIF；DBBC；GPU；干涉测量；深空探测

**中图分类号：**V556　　**文献标识码：**A　　**文章编号：**CN11-1780(2020)05-0042-10

## A new data encoding method for VDIF data based on GPU

CHEN Yongqiang,　MA Hong,　JIAO Yiwen,　LIU Yandu
（Department of Electronic and Optical Engineering, Space Engineering University, Beijing 101416, China）

**Abstract:** In order to improve the flexibility of the deep space station VLBI baseband converter and the adaptability of multiple data formats, a data encoding method which conforms to VDIF specification is developed based-on GPU, including frame header computing module, multi-channel single-datathread encoding module and multi-channel multi-datathread encoding module. To optimize the efficiency of encoding process, a data quantization method based on lookup table using texture memory and a multi-frame asynchronous encoding method based on streaming architecture are designed. Experimental verification shows that the encoding method designed in this paper is accurate, efficient and reliable, which is expected to provide effective support for VLBI data recording system of China's deep space station to participate in international VLBI joint observation.

**Key words:** VDIF; DBBC; GPU; Interferometry; Deep space exploration

## 引　言

甚长基线干涉测量 VLBI（Very Long Baseline Interferometry）技术利用孔径综合思想，将分布在各地的观测天线等效成为一个天线，具有极高的测角精度和空间分辨率，能够为空间目标的观测提供高精度横向信息。目前，我国已建成中国科学院 VLBI 网、国家授时中心 VGOS VLBI 全球观测系统（VLBI Global Observation System）多站观测系统，以及我国深空测控网等多个重要的 VLBI 观测网络，这些网络将在基于 VLBI 的时空框架的建立和维持、地球自传参数测量、深空导航以及大地测量等领域扮演越来越重要的角色[1-3]。完整的 VLBI 观测网络由观测站和相关处理中心两部分构成，VLBI 数字基带转换器 DBBC（Digital Baseband Converter）作为观测站数据采集、记录和传输的核心设备，在我国深空探测及国际 VLBI 联测中发挥了重要作用。

为了实现联合观测的数据交互，VLBI 研究机构在 21 世纪初为 VLBI 系带转换器数据记录系统设计了 VLBI 标准接口 VSI（VLBI Standard Interface）规范[4]，该标准是上一代干涉测量系统基带转换器实信号数据接口和软件控制的指定标准格式，并作为国际 VLBI 联合观测的主要数据交换格式一直沿用至今。然而，VSI 规范却未将 VLBI 数据传输格式纳入考虑范围[5]。为了解决数据传输问题，基于标

*基金项目：北京市科技重大专项（Z181100002918004）
收稿日期：2020-08-16　收修改稿日期：2020-09-18

准 RTP/RTCP 协议的 e-VLBI 标准被开发出来，即 VSI-E，该标准功能全面，但由于其协议过于复杂，导致难以得到广泛应用[6]。

近年来，随着通信技术和网络技术的快速发展，许多研究机构研制了新一代 VLBI 数据基带转换系统，由于各机构产品标准不一，导致国际 VLBI 联合观测数据交换需要进行大量的数据格式转换。为了解决此问题，2008 年，国际 VLBI 研讨会决定开发 VLBI 接口规范 VDIF（VLBI Data Interchange Format）[7]。与 VSI-E 格式相比，VDIF 格式更加简化，数据格式设计完全独立于传输协议，这使得该标准适合于各种网络传输协议和磁盘文件传输。近年来，随着 VDIF 标准的不断更新[5,8]，大量新一代基带转换系统，包括欧洲 DBBC3[9]、中国 CDAS3[10]、美国 R2DBE[11]、日本 OCTAVE[12]、俄罗斯 MDBE[13]等系统均支持该标准，同时，我国部分院校的研究人员对 VDIF 格式在 FPGA 平台的实现方式也进行了研究[14]。

我国深空测控网 VLBI 基带转换与记录系统是一套基于 FPGA 平台开发的 VLBI 数字后端 DBE（Digital Backend Equipment）系统，该系统从设计之初便同时支持 VSI 标准规范和 RDEF 规范。但随着系统功能的拓展，该系统需支持 VDIF 格式以应对 VLBI 联合观测需求的日益增加，而当前系统的体系结构给系统进一步升级换代带来了较大困难。为了提高系统的灵活性和扩展性，同时提高系统的可重构性，需要对系统结构进行改进。

本文通过研究，利用图形处理单元 GPU（Graphic Process Unit）的高灵活性和高效并行数据处理能力[15]，设计了一种基于 GPU 的干涉测量 VDIF 格式数据编帧方法，并通过试验验证了方法的正确性。

## 1　干涉测量 VDIF 数据格式规范

### 1.1　VDIF 数据格式规范简介

VDIF 数据帧（VDIF Data Frame）由数据帧头（Data Frame Header）和数据阵列（Data Array）两部分构成。帧头处于数据帧的头部，是一个带有自我识别标识的 32 字节字段，内含用于标识本帧数据的附加信息。帧头后面跟着一个或多个频率通道采样点组成的时间数据序列。

VDIF 规范的数据排列具有很高的灵活性，一个数据帧中可携带多个通道的数据，也可以仅携带一个通道的数据。在 VDIF 规范中，来自同一组子带的时间序列帧被称作一个数据线程（Data Thread），同一个数据线程中每一个数据帧用位于帧头内的线程序号（Thread ID）来标识。一组数据集内的数据线程被合并成单个串行的数据流（Data Stream），一个观测扫描周期内的所有的数据线程被称作一个数据段（Data Segment）。在 VDIF 规范中推荐了两种数据流组织方式[7]：①一个数据流仅由一个数据线程组成；②一个数据流由多个单通道数据线程组成。两种模式下数据流如图 1 所示。

VDIF 规范的设计同时适用于网络传输和硬盘存储。在网络传输模式下，每个数据包仅包含一个 VDIF 数据帧作为其数据载荷，在此模式下，数据帧长一般被限制在约 64 字节～9000 字节。而在硬盘存储模式下，

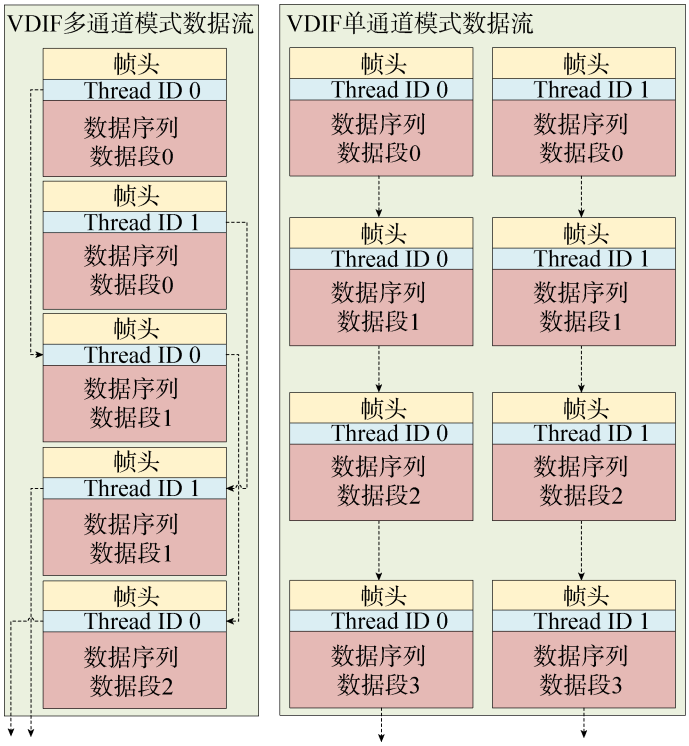


图 1　多通道模式和单通道模式 VDIF 数据流示意图

Fig. 1　Schematic diagram of VDIF data flow in multi-channel mode and single-channel mode

数据帧长受限于帧头中表示帧长的数据位数，即 $2^{27}$ 字节。

### 1.2　VDIF 数据帧头

标准的 VDIF 数据帧头大小为 32 字节，其内部结构如图 2 所示[7]。

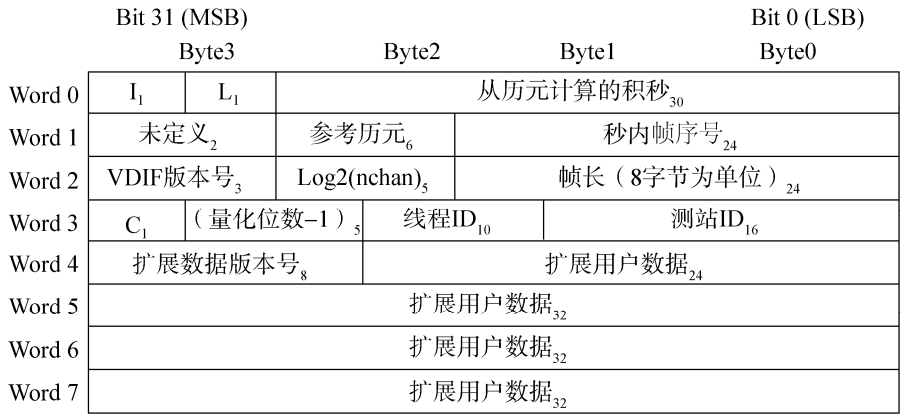


图 2　VDIF 规范数据帧头格式

Fig. 2　VDIF standard data frame header format

当前，VDIF 规范数据帧头格式已经更新到第四版[16]，该版是为了解决多路复用 VDIF 数据的有效性问题而设计。其核心功能主要解决两个方面的问题，一是大量数据处理时丢帧的问题，二是相关处理时数据重新排列（corner-turn）的问题。此外，NICT[17]、NRAO[18]、Haystack[19]和 ALMA[20]等机构均在 VDIF 规范的基础上利用用户扩展字节开发了自己的 VDIF 版本。

### 1.3　VDIF 数据序列的排列

标准的 VDIF 数据序列的排列方式只与帧头内的通道数和量化位数有关。数据格式必须遵守这些基本设定以便后续的数据处理设备能够快速解析数据。根据帧头中数据类型和通道数的定义，VDIF 规范的数据排列方式可分为单通道实数，单通道复数，多通道实数，多通道复数四种模式。在 VDIF 规范中，复数是以实部和虚部两部分单独处理的，其量化与实数相同，只是数据排列时将复数的 I、Q 两部分连续排列即可。

① 单通道 VDIF 数据排列格式

在单通道实数模式下，数据量化位数 1bit 到 32bit 任选，数据排列从低位开始，最先进入的数据占据 bit0，其后的数据依次占据后面的数据位，直到 32bit 字的剩余空间不足一个采样点量化存储时，从下一个 32 位字重新开始写入数据，即一个数据的量化结果不跨越 32 位字的边界。需要注意的是，第一个数据点的时刻必须与帧头中标识的时间严格对应。以 2bit 量化为例，单通道模式下实数数据排列方式如图 3（a）所示。

单通道复数模式下 VDIF 数据排列方式与实数排列方式类似，主要区别在于复数的一个采样点实际上由实部和虚部两个部分构成，且两个部分的量化位数相同。这两个部分在量化时是作为一对数据处理而且是相邻放置。放置时，实部位于低位，虚部位于高位。当量化位数大于 16bit 时，实部占据前两个 32 位字，虚部占据相邻的后两个 32 位字。单通道模式下量化位数 2bit 的复数数据排列方式如图 3（b）所示。

② 多通道 VDIF 数据排列格式

为了符合传统 VLBI 应用实践，同时简化数据格式，VDIF 规范多通道模式仅支持 2 的整数次幂数量的通道数和量化位数。为了定义多通道模式下数据格式，VDIF 规范提出了“complete sample”的概念，complete sample 表示所有通道一次采样所得量化位数。在实数模式下，若用 $2^n$ 表示通道数，$2^k$ 表示量化位数，则 complete sample=$2^n \times 2^k$。多通道模式下 complete sample 为 4bit 的实数数据排列方式如图 3（c）所示。需要注意，该模式下每个数据矩阵必须包含整数个 complete sample，而这一要求在通

道数较多时将面临困难。因此，多通道模式在网络传输时会遇到较大的障碍，而在文件记录和传输时不受影响。

在多通道复数模式下，通道数依然限制为 $2^n$，而复数的实部和虚部的量化位数均为 $2^k$，因此复数模式下，complete sample=$2\times2^n\times2^k$。复数数据各通道排列方式与实数相同，而实部和虚部的排列方式与单通道模式下相同，多通道模式下 complete sample 为 8bit 的复数数据排列方式如图 3（d）所示。

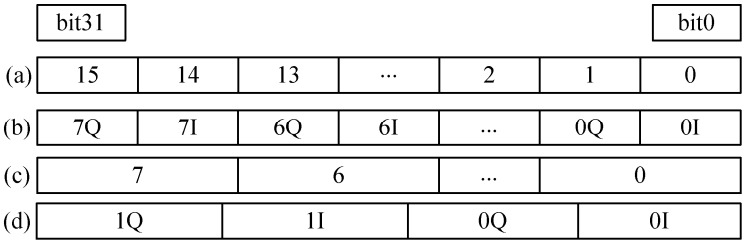


图 3　VDIF 规范数据排列方式

Fig. 3　VDIF specification data arrangement

## 2　基于 GPU 的 VDIF 格式数据编帧方法

### 2.1　基于 GPU 的 VDIF 格式数据编帧流程

典型的深空测控干涉测量系统基带转换器结构如图 4 所示[21]。基带转换器整体由数据预处理模块、并行信道化模块、数据格式化模块和 IO 模块组成。模拟输入信号经过数据预处理模块的幅度调节和并行信道化模块的分通道处理，最终按照要求输出固定带宽的多路并行子带信号。数据格式化模块按照接口文件的要求，将各通道信号量化编帧，输出符合标准格式规范的数据流。数据格式化模块是基带转换器和相关处理机沟通的桥梁，只有经过该模块严格格式化的数据才能最终通过 IO 模块送到数据处理中心进行进一步处理。

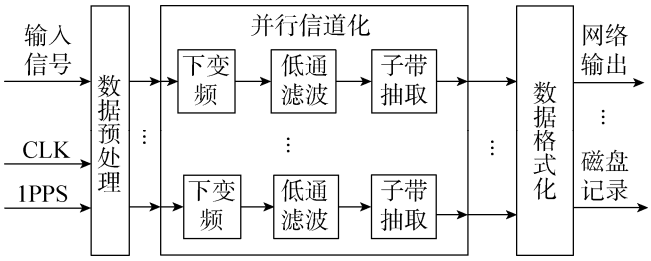


图 4　VLBI 基带转换器工作流程

Fig. 4　Workflow of VLBI baseband converter

经过并行信道化模块的处理，串行的高速数据变为并行的多路低速数据，在数据速率降低的同时，数据的并行性大幅提高，各通道之间数据完全独立，可充分利用 GPU 实现数据的并行化处理。

根据 VDIF 格式规范要求，多路并行的数据可选择多通道合并量化或者单通道单独量化，最终输出格式化的数据。由于各路数据之间互相独立，且同一通道内各数据之间也有独立性，因此可利用每个 GPU 线程完成一个数据点的量化，最终将量化结果合并为格式化文件。计算统一设备架构 CUDA（Compute Unified Device Architecture）是 NVIDIA 提出的 GPU 平台并行计算模型，基于 CUDA 的 VDIF 数据编帧流程如图 5 所示。

由图 5 可知，该流程主要由三个模块构成，即初始化和帧头参数计算模块、单通道单线程编帧模块以及多通道单线程编帧模块。该流程输入数据为信道化后输出的并行多通道数据，数据处理模块根据量化参数生成帧头，并按照量化模式分别对多通道数据进行量化处理。在单线程多通道模式下，系统仅开启一个数据线程，所有通道的数据均按通道顺序量化编帧，由于 CUDA 数组访存是行优先，为

了提高访存效率，首先对数据进行转置，然后按行对数据进行逐点量化并组成字节和 32 位字。在单线程单通道模式下，系统按照通道数启动多个数据线程，每个数据线程处理一路数据。而在 CUDA 环境中，可采用 CUDA Stream 异步启动多个流并行处理每一通道数据。在一帧数据处理完后，循环更新帧号并处理下一帧数据，实际上，由于数据的并行性，此流程也可以同时实现多帧数据的并行量化。

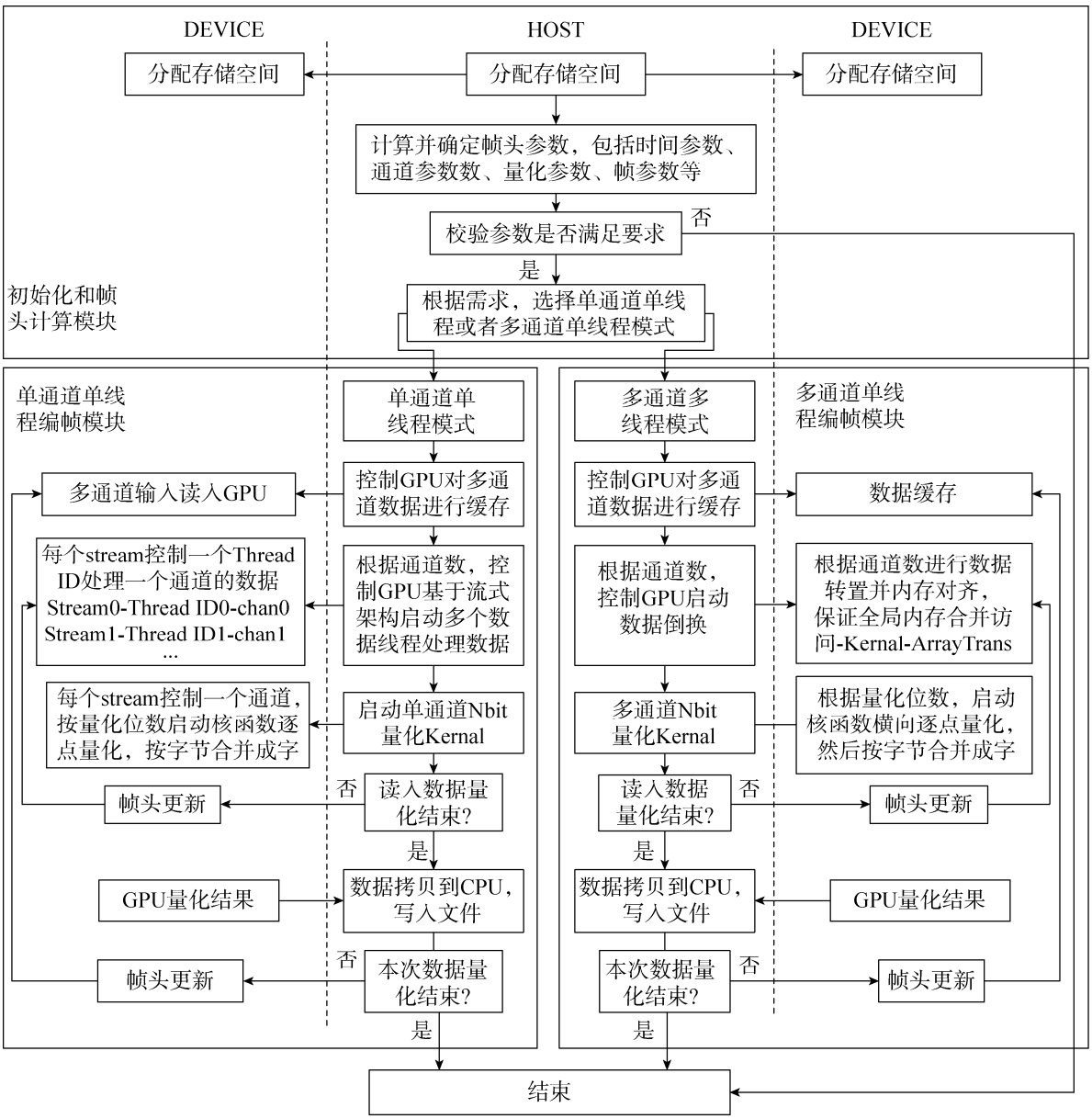


图 5　基于 GPU 的 VDIF 数据编帧流程

Fig. 5　GPU-based VDIF data encoding process

## 2.2　帧头参数计算模块

帧头参数计算模块主要完成计算单元的初始化和帧头参数的计算，而帧头参数的计算为其核心内容。在实际处理过程中，帧头参数的计算和更新流程如图 6 所示。

在图 6 中，帧头参数计算模块输入参数为量化位数、带宽、通道数、帧长、数据类型。该模块接收到上述参数后，首先启动时标计算，根据 VDIF 格式规范，首先根据当前时间计算当天的参考历元 epoch，然后根据 epoch 计算当前时刻的积秒。时标计算完成后，根据通道数、量化位数和数类型计算

complete sample，同时根据带宽、量化位数和数据类型计算数据速率。最后，根据数据速率和 complete sample 计算量化后的数据量，并根据输入帧长参数对帧长按照 8 字节取整。计算完成后，该模块进行参数合法性判定：一是 1s 内帧数是否为整数，二是量化后数据量是否为完成量化位数的整数倍。至此，帧头参数的计算和校验完成，加入线程 ID、测站 ID 和扩展数据后即创建初始帧头。

初始帧头创建完成后，数据量化流程同时启动，按照量化的数据帧信息，帧头内时标和帧序号实时更新并循环写入数据帧，直到量化编帧流程结束。

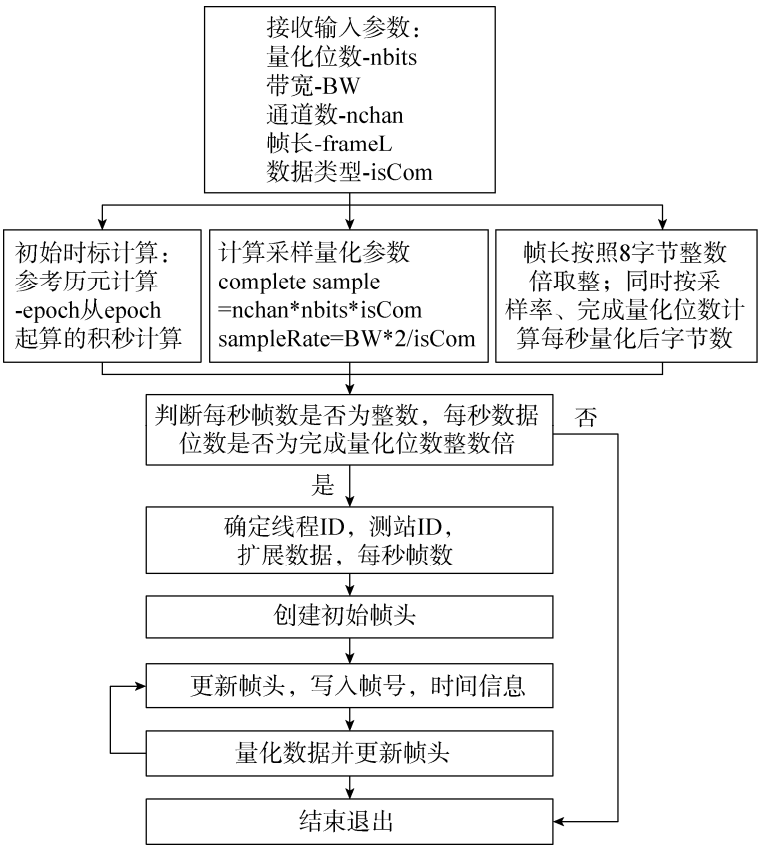


图 6　VDIF 规范数据帧头创建和更新流程

Fig. 6　VDIF specification data frame header creation and update

### 2.3　多通道单线程编帧模块

在多通道单线程模式下，为了实现数据存取操作的加速，需要将原来按通道行优先存储的数据变为按通道列优先存储，这样按行高效存取时就能直接获取一次采样的所有通道数据，使得存取效率大大提高。

在完成了数据转置后，并行的多路数据变为按列存储的二维数据矩阵。在此二维数据的基础上，针对性开辟二维 CUDA 线程块，每个线程负责一个数据点的量化，最终将量化结果拼接成字节存储。以 2bit 量化为例，线程块的分配、数据量化流程如图 7 所示。线程网格分配按照 512 为基数循环加载，保证处理完所有的数据点。线程网格内，线程块分配为二维，第一维度线程数大于通道数并以 32 为单位增加；第二维度代表需要处理的每通道数据点，设置线程数为 256，并随着线程网格循环加载直到处理完所有数据。在量化阶段，每个线程负责处理一个数据点，首先执行位置判断确定其所处字节位置，然后执行量化判断选择量化结果，最终按顺序将相邻四个数据点合并为字节并编入帧结构。

与传统基带转换器数据格式相比，VDIF 规范多通道模式下通道数和量化位数仍然服从 2 的整数次幂的约束，但完成量化位数不再受 32 位字空间的约束，可以将符合 complete sample=$2\times2^{n}\times2^{k}$

（$n$=0~31, $k$=0~5）的所有数据经行编帧处理。从图 7 可知，在多通道数据转置完成后，量化编帧过程仅与完成量化位数有关，据此便可得到不同完成量化位数条件下的量化核函数。

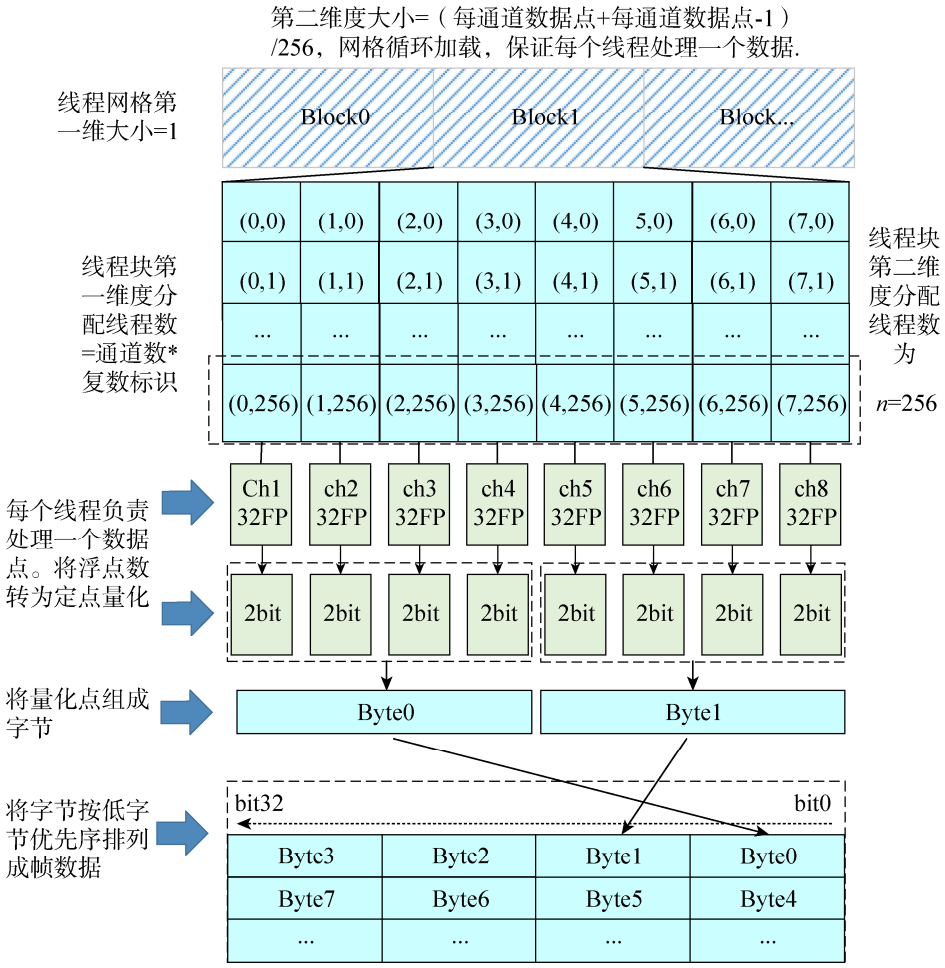


图 7　2bit 量化过程

Fig. 7　2bit quantization process

## 2.4　多通道多线程编帧模块

在多通道多线程模式下，并行的各通道数据由独立的数据线程进行处理，而由于各路数据相互独立，可采用流式异步并行架构对各路数据同时进行量化处理，数据处理流程如图 8 所示。在处理过程中，并行的各路数据流被按通道号分配给不同的 CUDA stream，各个 stream 同时拷贝数据到各自空间并执行异步量化和编帧，待各路编帧完成后，各 stream 读入下一段数据并重复上述过程。待处理完成后，符合 VDIF 格式规范的数据通过主机接口写入文件或写入网络数据包发送给用户。

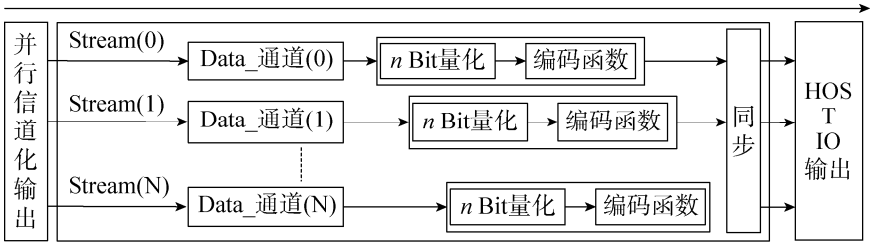


图 8　多通道多线程编帧流程图

Fig. 8　Flow chart of multi-channel and multi-threaded framing

在多通道多线程模式下，一个数据线程负责处理一个单独的数据通道，所以其数据编帧方式更加灵活，量化位数不再受到 2 的整数次幂的限制，但其整体数据帧结构仍然受到以 32 位字为单位的制约。另外，由于各通道数据本身就是按照行优先排列，不再需要数据转置。

由于量化过程的相似性，多通道多线程模式下数据量化编帧核函数依然分为数据量化映射与按位编帧两个步骤，因此可以认为单通道单线程是多通道单线程在通道数为 1 时的一个特例，在常用的 2 的整数次幂量化位数条件下，两种模式重要核函数可复用。

## 3　编帧模块的优化

数据编帧模块实现了基于 GPU 的并行多路数据的逐点量化和编帧。然而，在量化核函数中，采用了大量的分支判断结构，该结构逻辑清晰，但在 GPU 核函数中会给执行效率带来不利影响。另外，上述量化编码结构采用逐帧循环处理的方法处理数据，由于数据量大，循环结构在无法有效占用 GPU 资源的条件下，将会降低数处理效率。为了解决以上问题，本文设计了基于纹理查找表的编码方法和基于流架构的多帧数据异步并行编帧方法。

### 3.1　基于纹理缓存查找表的数据量化方法

为了解决分支结构给量化过程带来的效率损失，本文采用基于纹理缓存的量化查找表代替分支结构，实现数据量化过程。纹理缓存是 GPU 全局内存上一块特殊的区域，该区域经过特别的硬件加速，能够按照输入索引输出对应的列表值，适用于实现高效的查找表。

基于纹理缓存的量化查找表实现步骤如下：

① 根据输入参数计算完成量化位数（complete sample=通道数×量化位数×复数标识），其中当数据类型为复数（实数）时，复数标识为 2（1）；

② 根据完成量化位数分配缓存空间，并加载量化数据表；

③ 将量化数据表绑定纹理内存；

④ 在量化编帧核函数内，按照输入数据调用纹理拾取核函数，寻址方式为钳位寻址（CUDA Address Mode Clamp），纹理拾取滤波模式选择取整量化（CUDA Filter Mode Point）；

⑤ 将量化结果按位置写入数据帧。

### 3.2　基于流式架构的多帧信号异步编帧方法

在上文所述的编帧方法中，无论是多通道模式还是单通道模式，数据帧的写入均遵循帧内数据点并行量化，每帧数据串行循环处理的思路。虽然帧内数据的并行处理有效提高了数据并行性，但数据帧的串行处理并没有充分利用读入数据段的并行性。根据帧头参数设置，每一次所要处理的帧数已经确定，且各帧之间数据完全独立。因此，可利用异步并行结构，在查找表量化环节之后，直接对量化后的数据分段异步并行编帧，这样可以进一步提高数据的并行性。然而，并行编帧之前，需要预先计算好各个数据帧头的参数。流式架构异步并行编帧方法流程如图 9 所示。

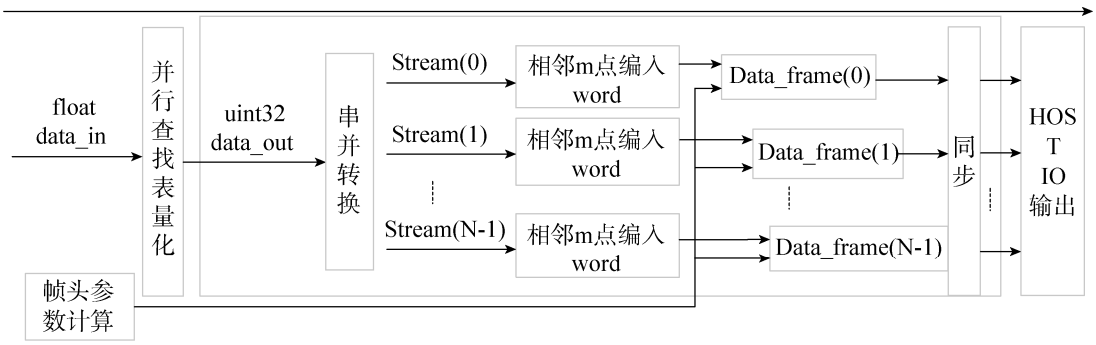


图 9　流式架构异步并行编帧算法流程

Fig. 9　Algorithm flow of asynchronous and parallel framing of streaming architecture

图 9 中，系统输入数据为信道化后的浮点数，该浮点数经过并行量化变为无符号的二进制量化数据。输出数据根据帧数 $N$ 分为并行的 $N$ 段，使得每段数据刚好组成一帧。启动 $N$ 个 CUDA stream，并保证每一个 stream 负责处理一帧数据。在每一个 stream 内，数据根据量化位数（多通道模式下为 complete sample）将相邻点的数据写入 32 位 word，并组成数据帧。

## 4　仿真验证与结果分析

为了验证所提编帧方法的有效性，本文利用仿真数据对编帧过程全流程进行了正确性测试。测试过程分为帧头测试和数据阵列测试两个部分。测试硬件平台为 HP ZBook-15，计算用 GPU 为 NVIDIA Quadro P2000 Mobile，计算能力 6.1。软件开发环境采用 Microsoft Visual Studio Community 2015 和 CUDA 10.2。

### 4.1　VDIF 格式帧头测试

帧头测试主要流程为，首先利用本文所提方法得到 VDIF 规范数据帧，然后利用测试软件测试帧头数据，验证帧头数据的正确性。测试时间为 UTC 时间 2020 年 6 月 21 日 1 时 46 分 11.5630 秒，带宽 2MHz。设置量化位数为 1，则实际量化位数 2bit。模式为单线程多通道模式，设置线程 ID 为 0，通道数对数为 1，则实际通道数为 2。设置不含帧头的 8bit 单位帧长为 1000，则实际帧长 8032 字节，数据为实数。另外设置测站 ID 为“HR”，VDIF 版本设置为 0，扩展数据 2 设置为帧同步字“0xACABFEED”，其余扩展数据位设置为零。这里为了测试方便，暂未对数据有效性信息相关的扩展字节定义。

根据以上信息生成 VDIF 格式数据帧，然后利用 VLBI 开源数据处理软件 DiFX[22]内用于 VDIF 数据测试的函数 printVDIFHeader 测试数据帧帧头数据，得测试结果如图 10 所示。从测试结果可知，本文设计的帧头生成结构可实现 VDIF 格式数据帧头正确生成。

### 4.2　VDIF 格式数据编帧结果测试

编帧结果测试主要流程为：首先，利用本文所提方法分别采用 CUDA 和 MATLAB 对输入数据进行量化，得到 VDIF 规范数据帧；然后，利用 MATLAB 分别提取两种方法生成的数据，并比较数据的正确性。输入数据选用均值为 0 方差为 1 的高斯白噪声。由此得到分别采用两种方法量化后一帧数据逐字节比对结果如图 11 所示，从图中可知，两种方法量化结果完全吻合，量化结果与理论值偏差优于 $10^{-10}$。测试结果证明了量化方法的正确性。

```
Current UTC time: 2020/6/21 1:46:11.563000
 weekday:0
VDIF Header
  epoch = 0x28 = 40
  seconds = 0xE2DAE2 = 14867170
  frame = 0x0 = 0
  threadid = 0x0 = 0
  framelength8 = 0x3EC -> frame length = 8032
  ln2 nchan = 0x1 -> nchan = 2
  nbits-1 = 0x1 -> nbits = 2
  legacymode = 0
  invalid = 0
  version = 0
  stationid = 0x4872 = 18546 = 'Hr'
  iscomplex = 0
  eversion = 0x0 = 0
  extended1 = 000000
  extended2 = ACABFEED
  extended3 = 00000000
  extended4 = 00000000
```

图 10　VDIF 规范数据帧帧头测试结果

Fig. 10　VDIF standard data frame header test results

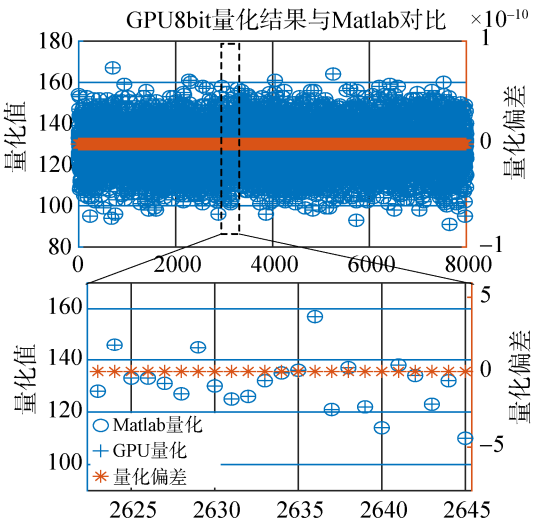


图 11　VDIF 规范数据 8bit 量化结果对比

Fig. 11　Comparison of 8bit quantization results of VDIF standard data

## 5 结束语

本文首先分析了干涉测量数据记录系统 VDIF 规范支持的必要性和基于 GPU 的编帧方法的可行性；然后分析了 VDIF 格式规范规定的数据编帧方法，介绍了帧头数据的填写规范，重点研究了不同数据类型、不同通道数和不同数据线程数条件下的数据排列方法；最后，基于 GPU 开发了符合 VDIF 规范的数据编帧方法，并介绍了帧头计算模块、多通道单线程编帧模块和多通道多线程编帧模块的实现方法。为了实现编帧过程效率的优化，设计了基于纹理缓存查找表的数据量化方法和基于流式架构的多帧信号异步编帧方法。通过实验验证，证明了本文编帧方法的正确性。本文所设计的基于 GPU 的 VDIF 规范数据编帧方法将能够有效提高我国深空测控干涉测量数据记录系统的灵活性和兼容性，为该系统参与联合观测提供有效的支持。

## 参考文献


[1] 李政, 杨旭海, 弓剑军, 等. VGOS 观测网初步观测及其讨论[J]. 时间频率学报, 2018, 41(1): 46–56.
LI Zheng, YANG Xuhai, GONG Jianjun, et al. Preliminary observation and discussion of VGOS network[J]. Journal of Time and Frequency, 2018, 41(1): 46–56.

[2] 陈中, 郑为民, 陈肖. 中国 e-VLBI 网的建立及应用[J]. 中国科学院上海天文台年刊, 2015(1): 136–147.
CHEN Zhong, ZHENG Weimin, CHEN Xiao, et al. E-VLBI applications in Chinese VLBI network[J]. Annals of Shanghai Astronomical Observatory Chinese Academy of Sciences, 2015(1): 136–147.

[3] 朱新颖, 李春来, 张洪波. 深空探测 VLBI 技术综述及我国的现状和发展[J]. 宇航学报, 2010, 31(8): 1893–1899.
ZHU Xinyin, LI Chunlai, ZHANG Hongbo. A survey of VLBI technique for deep space exploration and trend in china current situation and development[J]. Journal of Astronautics, 2010, 31(8): 1893–1899.

[4] NAKAJIMA J, KOYAMA Y, KONDO T, et al. VSI interface implementation, performance enhancement of Gbps VLBI instruments[C]. International VLBI Service for Geodesy and Astronometry: General Meeting Proceedings, 2002.

[5] WHITNEY A, KETTENIS M, PHILLIPS C, et al. VLBI data interchange format (VDIF)[J]. European VLBI for Geodesy & Astrometry Working Meeting, 2010, 194(2-5): 156–160.

[6] RUSZCZYK C. Real-time data transfer with VSI-E and EGAE[C]. 2020.

[7] WHITNEY A, KETTENIS M, PHILLIPS C, et al. VLBI data interchange format (VDIF) (invited)[C]//The 8th International e-VLBI Workshop. 2009.

[8] VLBI data interchange format (VDIF) specification[EB/OL]. 2014. https://vlbi.org/wp-content/uploads/2019/03/VDIF_specification_Release_1.1.1.pdf.

[9] TUCCARI G. Functional description of the DBBC3[C]. MIT Haystack Observatory in Westford, Massachusetts, USA. Approximately 50km NW of Boston: 9th IVS Technical Operations Workshop, 2017.

[10] ZHU R, WU Y, LI J, et al. The development of VLBI digital backend in SHAO[J]. 2018.

[11] VERTATSCHITSCH L, PRIMIANI R, YOUNG A, et al. R2DBE: a wideband digital backend for the event horizon telescope[J]. Publications of the Astronomical Society of the Pacific, 2015, 127(958): 1226–1239.

[12] OYAMA T, KONO Y, SUZUKI S, et al. A progress report on the development and performance of OCTAVE-DAS for VERA, JVN and Japanese e-VLBI (OCTAVE)[C]// URSI General Assembly and Scientific Symposium, 2014.

[13] IPATOV A, IPATOVA I, MARDYSHKIN V. MDBE-BRAS-Russia-IAA technology development center report 2014[R]. 2014.

[14] 赖见涛. VDIF 数据编帧模块和数据预处理模块的设计及 FPGA 实现[D]. 上海: 上海应用技术大学, 2017.
LAI Jiantao. Design of VDIF data encoding module and data preprocessing module and corresponding implementation by FPGA[D]. Shanghai: Shanghai Institute of Technology, 2017.

[15] NVIDIA. Cuda C++ Programming Guide[EB/OL], 2019. https://docs.nvidia.com/cuda/pdf/CUDA_C_programming_guide.pdf.

[16] BRISKEN W. Description of the VDIF extended data version 4: multiplexed VDIF data validity[EB/OL]. 2016. https://vlbi.org/wp-content/uploads/2019/03/vdif_extension4.pdf.

[17] VDIF Extension Version 0x01[EB/OL]. 2014, https://vlbi.org/wp-content/uploads/2019/03/vdif_extension_0x01.pdf.


(下转第 70 页)