DOI:10.19651/j.cnki.emt.2209011

基于 GPU 的卫星信号信道化并行设计与实现

胡 杰 刘 凯
(上海大学通信与信息工程学院 上海 200444)

摘 要:针对军事信息侦察中卫星接收系统信号处理速度慢的问题,设计了一种基于 GPU 的卫星信号信道化方案。利用数字下变频级联多相信道化来提高子信道带宽位置的灵活性;设计二维 Block 复用输入数据以并行处理多路下变频过程;基于共享内存设计多相滤波内核完成 72 路信道化中的加权叠加运算。CUDA 环境下使用 NVIDIA RTX 2060 和 Intel(R) Core(TM) i5-6400 的测试结果表明,本设计并行处理 4 路 75M 实信号的 288 路信道化耗时 4.51 s,相较于单 CPU 处理的加速比高达 40.27 倍,极大地提高了侦察卫星接收端的处理速度。

关键词:数字信道化;GPU;并行设计;共享内存
中图分类号:TN914 **文献标识码**:A **国家标准学科分类代码**:510.50

Parallel design and implementation of satellite signal channelization based on GPU

Hu Jie Liu Kai
(School of Communication & Information Engineering, Shanghai University, Shanghai 200444, China)

Abstract: In order to solve the problem of slow signal processing speed of satellite receiving system in military information reconnaissance, a satellite signal channelization scheme based on GPU is designed. Using digital down conversion cascaded multiphase channelization to improve the flexibility of sub-channel bandwidth location. Two-dimensional Block multiplexing is designed to process the multichannel down conversion process in parallel. A multiphase filtering kernel based on shared memory is designed to perform the weighted stack operation in 72 channels channelization. The test results of NVIDIA RTX 2060 and Intel(R) Core(TM) i5-6400 in CUDA environment show that the 288 channel channelization of four 75 M real signals takes 4.51 s in parallel processing, which is 40.27 times faster than that of single CPU processing. It greatly improves the processing speed of the receiving end of reconnaissance satellite.

Keywords: digital channelization; GPU; parallel design; shared memory

0 引 言

近些年,卫星通信因其具有覆盖区域大、通信距离远、信道质量好、传输稳定等诸多优势特点[1],在军事和民用领域得到了广泛的应用。当前的卫星通信系统大多采用频分多址(FDMA)和多频时分多址(MF-TDMA)[2]等多址接入技术为不同用户分配不同的频段资源,从而实现多路频分复用。在卫星通信系统接收端,一般利用数字信道化技术来分离各频路信号,为了减轻射频前端的处理压力,将模数转换(ADC)尽可能地靠近天线部分[3],从而可在数字域中完成更多的信号处理功能,在此基础上,借助并行平台的强大算力,有望缩短处理时间,从而提高卫星通信侦察的时效性。

图形处理器(graphics processing unit, GPU)因其具有众多运算核心和多种内存区域[4],天然适合于大数据的并行处理,目前已有研究学者将 GPU 的多线程并行技术应用到雷达信号处理[5]、遥感卫星图像处理[6]等实时性要求较高的领域,并且取得了不错的加速效果。文献[7]基于 GPU 实现了一种 64 通道的均匀多相信道化器,相较于串行信道化模型,速度提升了 16 倍;文献[8]基于 GPU 对多相信道化算法效率进行了分析,给出了时域卷积和频域卷积的最佳适用条件;文献[9]给出一种基于 GPU 的快速数字下变频实现,利用数据重叠和缓存置换算法极大地提高了处理效率。

基于上述研究,本文提出一种基于 GPU 的卫星信号信道化设计方案。为了实现对子系统带宽的灵活处理,采用

收稿日期:2022-02-15

数字下变频级联多相信道化的结构，并在此基础上集成分数倍采样率变换；为了提高信道化处理的速度，将下变频和多相滤波过程的运算并行化映射到 GPU 的流多处理器上；为了进一步优化整体时间开销，调整数据的分块大小，并利用共享内存缓存显存中非连续分布的数据，使得块内线程满足对齐访问原则。

1　信道化原理分析

本文所述设计用于一卫星通信系统的接收端，该系统采用频分复用方式，系统带宽为 B，其中非均匀分布着 N 个频宽相同的子信道，每一个子信道可用于传输 M 路带宽相同的窄带信号。我们可以通过 $M\times N$ 次数字下变频（digital down converter，DDC）得到每一路目标信号，但这种方式引入了大量的重复计算，且考虑到每一子信道处的窄带信号呈现均匀分布特征，因此可以采用两级结构来实现：第 1 级采用 DDC，将 N 个子信道变频到零频附近，由于待处理信号采样率 f_s 一般是百兆级，因此可以采用多相滤波来降低计算压力；第 2 级采用多相滤波加离散傅里叶变换的均匀信道化，由此，输出最终的 $M\times N$ 路窄带信号。

1.1　基于多相滤波的数字下变频

设接收端的输入信号为 $x(n)$，采样率为 f_s，第 1 个子信道中心频点为 ω_d，经下变频、低通滤波、变采样后得到采样率为 f_d 的输出信号 $y(m)$，$m=n\times D$，$D=f_s/f_d$，该过程如式(1)、(2)所示，其中 $h(n)$ 为低通滤波器。

$$y'(n)=\sum_{k=-\infty}^{+\infty}h(k)x(n-k)\mathrm{e}^{\mathrm{j}\omega_d(n-k)} \tag{1}$$

$$y(m)=y'(nD) \tag{2}$$

当 D 不是正整数的时候，不能直接对 $y-(n)$ 进行抽取，而应该对其采取先插值、后抽取的方式得到 DDC 的最终结果。多相滤波器为这类操作提供了实现框架[10]，在满足采样定理的前提下，使用内插/抽取并配合滤波器，可以实现数据的分数倍采样率变换。

设滤波器 $h(n)$ 的长度为 L，$x_D(n)$ 为 $x(n)$ 下变频后的结果，当 $D=Q/P$ 时，将 $x_D(n)$ 经 P 阶内插，Q 阶抽取后可以得到采样率为 f_d 的 $y(m)$，其实现过程如式(3)所示，其中 $<.>_P$ 表示对 P 取余，$[.]$ 表示向下取整。则每计算一点 $y(m)$ 需要 L/P 次乘法，$L/P-1$ 次加法，相较于采用多相滤波结构之前运算效率提升了 P 倍。

$$y(m)=\sum_{k=0}^{L/P}h(kP+<Qm>_P)x_D\left([\frac{Qm}{P}]-k\right) \tag{3}$$

1.2　信道化方案设计

经过第 1 步 DDC 之后，将中心频点位于 ω_d 的信号搬移到了零频，该信号内包含 M 路频点均匀分布的窄带信号，且每路信号的采样率相同，因此可以通过一个均匀信道化器来完成最终的分离。于是，可以得到最终的信道化方案设计如图 1 所示。

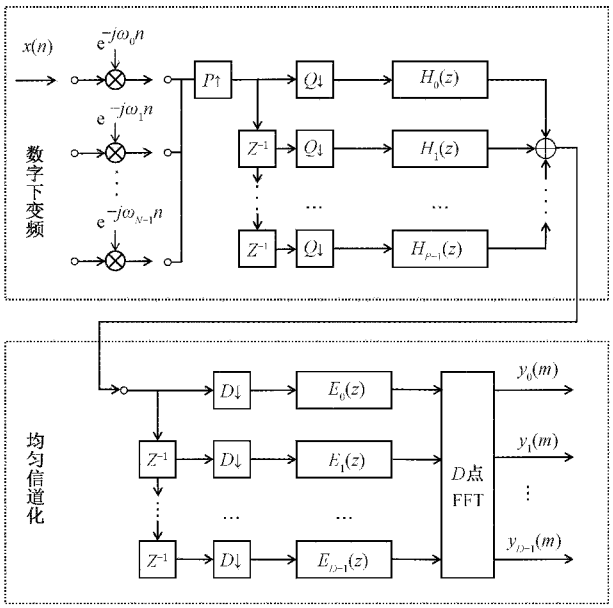

图 1　信道化方案图

2　信道化并行设计

2.1　变采样器的多线程设计

CUDA（compute unified device architecture）是 NVIDIA 公司提出的一种通用 GPU 计算框架，主要由运行时库和 C/C++ 的扩展版本组成[11]，该框架采用单指令多线程体系结构，可用于解决复杂的计算问题，因此，本文设计在 CUDA 环境下进行实现。

在本设计的目标通信系统中，$f_s=75$ MHz，第 1 级输出采样率 $f_d=7.56$ MHz，则输入信号 $x(n)$ 经 $P=63$ 倍内插，$Q=625$ 倍抽取可实现采样率的正确变换，利用 Kaiserord 窗函数得到插值抽取滤波器 $h(n)$ 的阶数 $L=18\ 207$ 阶，记一次乘法运算为两次加法运算，则计算得到 $y(m)$ 需要的运算量 $T_1(n)$ 如式(4)所示，如果采用传统思路下的 CPU 串行计算，需要的时间开销将十分巨大，如何设计 GPU 内核来精确、快速的计算出相同的结果是本部分设计的研究重点。

$$T_1(n)=f_d\times(2L+L-1)\approx 413G\ FLOPs \tag{4}$$

变采样的运算过程如图 2 所示，其中 $x_P(n)$ 为 $x(n)$ 内插 P 倍后的结果，通过分析可知每得到一点 $y(m)$ 需要的有效运算量为 289 点数据与 289 点系数的乘累加，而其余部分的运算结果可提前预知为零（数据值为 0），因此如何准确的找出非零值在内存中的位置成为内核设计的关键。设数据的起始下标为 i，系数的起始下标为 j，则由式(3)可知，i 和 j 可由式(5)、(6)唯一确定。

$$i=floor(k*Q/P) \tag{5}$$

$$j=mod(k*Q,P) \tag{6}$$

于是，可考虑设计核函数 D_SRC1，在其一个 Block 中开启 289 个 Thread 完成 289 点的乘法和加法，每个 Thread

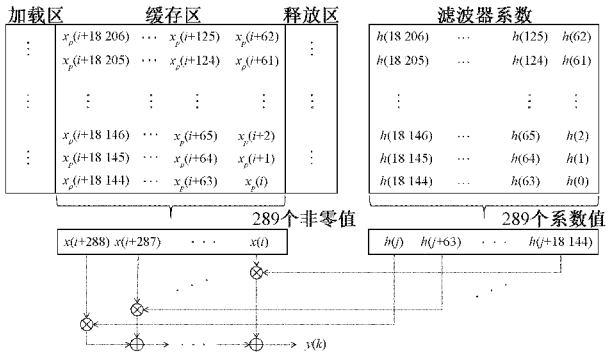

图 2　插值抽取滤波过程

的指令流可简化表述为:

1)取滤波器系数、信号数据;

2)计算点乘;

3)等待同步;

4)点乘相加,写数据。

然而,在 GPU 实际执行的过程中,基本的调度单位是 32 个 Thread 组成的线程束(Warp),通常而言,同一 Warp 中的所有 Thread 在任意时钟周期都执行相同的指令流,除非遇到逻辑分支而分化[12]。考虑 289 不是 32 的整数倍,因此 GPU 在实际的调度过程中一定存在 Warp 内分化严重的情况,结合 Nsight Compute 工具得知 D_SRC1 的 SM Busy 率只有 27.41%、SM 利用率只有 67.04%,于是重新设计 D_SRC1 的 Block 如图 3 所示,在 D_SRC1 的一个 Block 中开启 512 个 Thread,在每一 Thread 内计算 289 点乘法和加法,使得在满足 Warp 调度规律的同时,增加单个处理核心的计算任务,此时 SM Busy 率上升为 31.39%、SM 利用率上升为 92.93%。

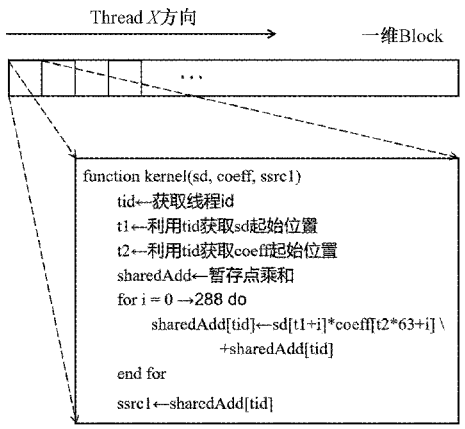

图 3　D_SRC1 的 Block 设计图

由式(5)可知,计算 $y(k)$ 与 $y(k+1)$ 需要的起始点数据 $x(i)$ 与 $x(i+1)$ 在内存中的距离约为 $floor(Q/P)=10$ 点,则 $y(k)$ 与 $y(k+1)$ 需要的总数据重叠约 279 点,从而 Block 内的相邻 Thread 将在不同时隙访问相同位置的数据,且该数据存在于全局内存中。GPU 中片上 Thread 访问全局内存的指令延时十分巨大,随着访问次数的增加,其时间开销足以抵消计算部分缩减的时间[13],设法优化这部分的内存访问结构成为提高内核并行度的关键。

共享内存作为 GPU 的片上高速缓存区,不仅提供远低于全局内存的访问延时,还可作为块内线程的通信方式[14]。于是可考虑使用共享内存暂存参与计算的数据,从而可缩短取数据的时间开销。

在此基础上,探索多路并行的可能性。由于每一路 DDC 的输入数据相同,只是在变频位置有所差异,因此,可增加 Block 的维度如图 4 所示,其中在 Block 的 x 方向开启 256 个 Thread,y 方向开启 4 个 Thread,利用共享内存暂存 289+255×10=2 839 点输入的 short 类型实数据,总共占用,在 y 方向的 4 个 Thread 中以内置索引 threadIdx.y 唯一确定对应的变频位置,则核函数的一个 Block 可实现同时处理 4 路 DDC,由此可提高核函数的执行效率。

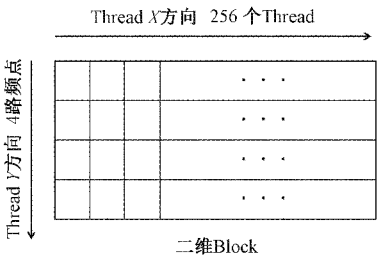

图 4　D_SRC1 的二维 Block 设计图

2.2　多相滤波的多线程设计

在第 1 级 DDC 的输出信号 $y(m)$ 内包含 $M=72$ 路带宽相等的窄带信号,每一路目标采样率为 $f_{out}=f_d/M=105$ kHz,则由 72 路均匀信道化可实现采样率的正确变换。此处用到的滤波器阶数为 $L_2=288$ 阶,经多相分解将其映射为 72 行 4 列的二维结构,同样将多相输入数据映射为相同的二维结构[15],则信道化过程可表示为图 5 所示,参与 FFT 运算的 72 点数据由每一行输入数据与对应行滤波器系数卷积得到,72 点数据的 FFT 结果分别为 72 路输出信号 $z(q)$ 中的一点,$q=0,1,\cdots,M-1$。因此每得到 72 路 $z(q)$ 中的一点,需要 4×72=288 次乘法和 3×72=216 次加法,以及一次 72 点的 FFT 运算,CUDA 的官方库 CUFFT 提供了 GPU 上的高效 FFT 实现,因此本部分只考虑图中左半部分的多相滤波运算,该运算量 $T_2(n)$ 如式(7)所示。

$$T_2(n)=f_{out}\times(2L_2+L_2-1)\approx 91G\ FLOPs \quad (7)$$

由 2.1 节知,为了提高内核的并行度,可在一个 Thread 内完成一组数据与子滤波器系数的乘加运算。首先考虑设计内核 D_SRC2,其 Block 维度为 72,在每一 Thread 内完成 4 点乘法和 3 点加法,Block 执行一次正好得到 72 组子滤波结果;但此时 Block 内的线程数过小,指令并行度仍可增加,且 72 无法被 32 整除,从而存在 Warp 分化的情况,因此考虑增加 Block 的维度为(72,12),如图 6 所示,其 y 方向表示数据移动的次数,此时 Block 执行一

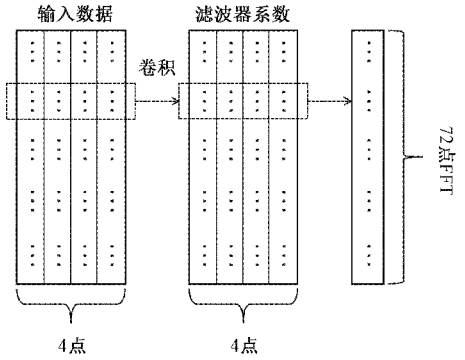

图 5　M 路均匀信道化过程

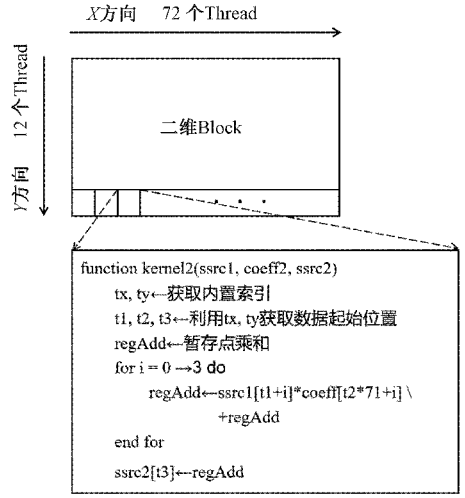

图 6　D_SRC2 的 Block 维度图

次，每路可得到 12 点输出结果。

但 Block 维度的增加同样带来对全局内存访问次数的增加。通过分析相邻两次滤波过程可知，数据寄存器每更新 72 点数据，72 路输出就吐出一点数据，因此相邻滤波可最多共用 288－72＝216 点数据。考虑将 Block 内滤波需要的一段数据全部缓存到共享内存中，以消除片上到全局的冗余访问，并提高获取数据的速度，此时一个 Block 内共享内存占用(288＋72×11)×2×4＝8.64 kB。

3　实验与分析

本次实验使用的 GPU 型号是 Turing 架构的 GeForce RTX 2060，计算能力为 7.5，显存容量为 6 GB；该型号 GPU 包含 30 个 SM，每个 SM 含有 64 个 CUDA 核心以及 48 kB 的共享内存，每个 SM 最大可并发执行 1 024 个线程、最多可贮存 32 个 Block；CPU 型号是 Intel(R) Core (TM) i5-6400，时钟频率 2.7 GHz，内存为 8 GB。在 Windows10 系统下采用 Visual Studio 2015 配合 CUDA10.2 作为开发环境。

取单次处理的数据长度为 10 M，利用上述优化后的设计完成信道化处理，将其运行时间与 CPU 端进行对比，变采样器的多线程优化结果如表 1 所示，其中 SRC1 对应 CPU 端变采样器的实现；D_SRC1 对应在核函数的一个 Block 中开启 289 个 Thread，相较于 SRC1 取得了 5.76 倍的加速效果；多线程优化对应图 4 的二维 Block 设计，相较于 D_SRC1 性能提升了 77%，相较于 SRC1 取得了 24.78 倍的加速效果。

表 1　变采样器的多线程优化时间记录　　ms

算法	SRC1	D_SRC1	多线程优化
时间	471	83	19
加速比	1	5.67	24.78

第 2 级多相滤波的多线程优化结果如表 2 所示，其处理的数据长度为 4.032 M，其中 SRC2 对应 CPU 端的实现；D_SRC2 对应图 6 的 Block 设计，其相较于 SRC2 取得了 3.56 倍的加速效果；内存优化对应使用共享内存暂存输入数据的情况，其相较于 D_SRC2 性能提升了 55.2%，相较于 SRC2 取得了 7.95 倍的加速效果。

表 2　多相滤波的多线程优化时间记录　　ms

算法	SRC2	D_SRC2	内存优化
时间	37.49	10.53	4.71
加速比	1	3.56	7.95

接下来分别利用 CPU、CPU-GPU 异构系统对 5 组不同大小的数据进行完整信道化处理，实验结果如表 3 所示，为了去除测量误差，每组数据取 5 次的平均值，可以看出，相较于 CPU，本次并行优化可以将处理速度提升 40 倍左右。

表 3　CPU 与 CPU-GPU 运行时间对比　　s

数据大小	CPU	CPU-GPU	加速比
20 MB	48.31	1.208	39.99
75 MB	181.63	4.51	40.27
200 MB	495.64	12.09	41.00
500 MB	1 246.94	30.17	41.33
1 GB	2 647.69	67.57	39.18

4　结　　论

本文针对卫星接收系统信号处理速度慢的问题，基于 GPU 提出一种卫星信号信道化方案。本文的核心思路是利用数字下变频级联多相信道化结构以实现兼具速度和灵活性的信道化处理。本文的核心技术为通过 GPU 单指令多线程的结构并行完成信道化过程的运算、通过 GPU 的片上内存结构加速信道化过程的运算。本次实验采用的 GPU 是 Turing 架构的 GeForce RTX 2 060，其 CUDA 核心数量只有 1 920 个，因此，在未来的研究中，可以针对目

前的高性能 GPU,例如 Volta 架构的 V100、Ampere 架构的 A100 对本文设计中的 Block 维度进行调整,使核函数单次处理任务量提升,从而达到速度的进一步优化。

参考文献

[1] 朱立东,张勇,贾高一. 卫星互联网路由技术现状及展望[J]. 通信学报,2021,42(8):33-42.

[2] 许楠. MF-TDMA 和 FDMA 传输自适应体制下的资源分配[J]. 无线电工程,2019,49(3):195-198.

[3] HARRIS F J, DICK C, RICE M. Digital receiv-ers and transmitters using polyphase filter banks for wireless communications[J]. IEEE Transactions on Microwave Theory and Techniques, 2003, 51(4): 1395-1412.

[4] 陈博伦. 基于嵌入式 GPU 的图像处理算法加速技术研究[D]. 上海:上海交通大学,2020.

[5] FANG H, LEI J, ZHANG J, et al. Modelling ground-penetrating radar wave propagation using graphics processor unit parallel implementation of the symplectic Euler method[J]. Near Surface Geophysics, 2019, 17(4):417-425.

[6] GAO J, SUN Y, ZHANG B, et al. Multi-GPU based parallel design of the ant colony optimization algorithm for endmember extraction from hyperspectral images[J]. Sensors, 2019, 19(3):598.

[7] AL-SAFI A, BAZUIN B. GPU based imple-mentation of a 64-channel polyphase channa-lizer[C]. Circuits & Systems Conference, IEEE, 2015:1-4.

[8] 陈永强,马宏,党宏杰,等. 基于 GPU 的多相信道化算法效率分析与应用[J]. 无线电工程,2021,51(3):189-198.

[9] KANDAUROV N A, LIPATKIN V I, VARLAMOV V O. Implementing digital downconversion on a GPU[J]. 2021 Systems of Signal Synchr-onization, Generating and Processing in Telecommunications(SYNCHROINFO), 2021: 1-8.

[10] KIM S C, BHATTACHARYYA S S. A wideband front-end receiver implementation on GPUs[J]. IEEE Transactions on Signal Processing, 2016, 64:2602-2612.

[11] 李涛,董前琨,张帅,等. 基于线程池的 GPU 任务并行计算模式研究[J]. 计算机学报,2018,41(10):2175-2192.

[12] 曹洁,胡文东,王进花,等. 基于 GPU 并行优化的 BBPSO-PF 算法[J]. 华中科技大学学报(自然科学版),2021,49(3):12-17.

[13] 林玉哲,张为华. GPU 事务性内存技术研究[J]. 大数据,2020,6(4):3-17.

[14] 徐坤浩,聂铁铮,申德荣,等. 基于 CPU-GPU 异构体系结构的并行字符串相似性连接方法[J]. 计算机研究与发展,2021,58(3):598-608.

[15] 韩学涛,梁宫,孙洪波,等. 基于多相滤波的通用数字信道化技术[J]. 现代雷达,2018,40(10):62-66.

作者简介

胡杰,工学硕士,主要研究方向为 GPU 并行计算、通信信号处理。

E-mail:hujiehujie@shu.edu.cn

刘凯,博士,副教授,主要研究方向为雷达信号处理、通信信号处理、免携带室内定位技术。

E-mail:liukai@shu.edu.cn