doi: 10.7690/bgzdh.2022.08.012

基于 GPU 的 PCM/FM 信号非相干鉴频解调算法

孙宽飞，杨文革，焦义文，滕　飞，高泽夫
（航天工程大学电子与光学工程系，北京 101416）

摘要：面向未来任务日益多样化，试验环境日趋复杂化，传统基于 FPGA 的脉冲编码调制/调频(pulse code modulation/frequency modulation，PCM/FM)遥测系统存在软硬件紧耦合、研发周期长、升级维护困难等缺点，无法满足未来遥测系统大规模部署对弹性、灵活和敏捷的需求。针对以上问题，设计基于 GPU 的 PCM/FM 信号非相干鉴频解调实现方法。根据算法原理和 GPU 并行方法设计出基于 GPU 的并行解调实现方法，分别从并行数字下变频、并行 FIR 滤波和并行差分鉴频 3 部分分析具体模块的并行化实现方法。实验结果表明：在码速率为 10 Mbps、每处理 1 s 数据的条件下，基于 GPU 的非相干鉴频过程消耗的时间约为 426.867 ms，满足解调的实时性要求；解调增益稍优于传统基于 FPGA 的 PCM/FM 遥测基带，有约 0.1 dB 的解调增益。

关键词：调频遥测；GPU；PCM/FM 解调；并行计算
中图分类号：TP873　　**文献标志码：**A

Non-coherent Demodulation Algorithm for PCM/FM Signal Based on GPU

Sun Kuanfei, Yang Wen’ge, Jiao Yiwen, Teng Fei, Gao Zefu
(*Department of Electronic and Optical Engineering*, *Space Engineering University*, *Beijing* 101416, *China*)

Abstract: With the increasing diversification of future-oriented tasks and the increasing complexity of test environment, the traditional pulse code modulation/frequency modulation (PCM/FM) telemetry system based on FPGA has some shortcomings, such as the tight coupling of hardware and software, the long development cycle, and the difficulty of upgrade and maintenance. It can not meet the needs of large-scale deployment of future telemetry systems for elasticity, flexibility and agility. In order to solve the above problems, a GPU-based non-coherent demodulation method for PCM/FM signals is designed. The parallel demodulation method based on GPU is designed according to the algorithm principle and GPU parallel method and the parallel implementation method of specific modules is analyzed from the 3 parts of parallel digital down conversion, parallel FIR filtering and parallel differential frequency discrimination. The experimental results show that the non-coherent frequency discrimination process based on GPU consumes about 426. 867 ms when the code rate is 10 Mbps and the data is processed every second, which meets the real-time requirements of demodulation, the demodulation gain is slightly better than the traditional PCM/FM telemetry baseband based on FPGA, with about 0.1 dB demodulation gain.

Keywords: FM telemetry; GPU; PCM/FM demodulation; parallel computing

0　引言

飞行器遥测是导弹、火箭、卫星等航天器试验和运行过程中必不可少的重要支持系统，能够实时监测航天器内部工作状态、电气性能、环境参数等重要信息，为航天器性能检测、效能评估及故障分析提供依据[1]。随着航天事业的深入发展，全球各航天大国对宇宙的探索和宇宙资源的争夺愈加强烈[2]，这使遥测技术突显出更为重要的作用。

脉冲编码调制/调频(PCM/FM)技术具有较强的抗尾焰效应能力、抗噪声性能强、抗多径衰落、抗相位和干扰发射机功率高等特点[3]，已成为国内外航空航天遥测领域长期采用的一种主流体制[4-5]。传统调频遥测均采用 FPGA 实现系统功能，存在硬件设计周期较长、硬件平台通用性低、升级难度大等问题，距离软件无线电设计理念差距较大[6]。随着高性能计算技术的发展，GPU 从专用于图像领域的处理器逐渐向着通用并行计算平台转变[7]。GPU 具有的运算核心数远多于 CPU，比较适合用于数据密集型计算的并行加速处理[8]。2007 年 NVIDIA 提出的计算统一设备架构(compute unified device architecture，CUDA)允许开发者使用 C 语言进行编程，简化了 GPU 系统的开发流程，降低了使用 GPU 并行编程的难度，使得 GPU 通用计算技术在信号处理领域得到更为广泛的应用[9]。基于 CPU+GPU 的信号处理系统成为众多领域的研究热点，如雷达[7,10]、射电天文[11-12]、无线电通信[13-14]等。

收稿日期：2022-04-20；**修回日期：**2022-05-28
基金项目：北京市重大科技专项资助项目(Z181100002918004)
作者简介：孙宽飞(1995—)，男，河南人，从事航天测控通信系统研究。E-mail:1319188235@qq.com。

胡蕊[15]以 CUDA 并行编程为核心，设计了一种 CPU+GPU 异构计算的实时频谱分析仪；孙坚武[14]利用 GPU 模块化实现了遥控遥测信号的分析识别，并在伪码宽度估计模块达到 13.201 倍加速比。当前，基于 GPU 的信号处理系统具有强大的并行处理能力，可在极短时间内完成规模化的运算处理，满足信号处理实时性的要求。在以上研究成果的基础上，笔者针对调频遥测信号的数据具有依赖性的特点，提出基于 GPU 的并行非相干鉴频解调实现方法。

1　并行非相干鉴频解调算法设计

在遥测解调中，不仅要满足解调结果正确的需求，而且要满足实时解调的要求；因此，需要设计基于 GPU 的并行处理算法，才可以有效解决遥测解调计算复杂、耗时过长的问题，进一步满足解调实时性的要求。并行性可大体分为 2 种：一种是数据并行性，另一种是任务并行性[16]。

1) 数据并行性是把接收到的一段时间内的数据分段，然后交给多个 CUDA 流进行并行处理。如图 1 所示，每一个数据段分别在不同的 CUDA 流上完成遥测解调运算。调频遥测信号是连续的数据，数据之间具有高度依赖性，将数据简单地分段处理会破坏数据之间的依赖性，进而造成解调数据错误，所以数据并行方案对于遥测解调过程并不适用。

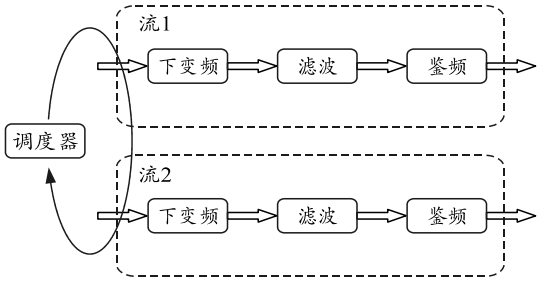


图 1　数据并行方案

2) 任务并行性不是通过数据分段让不同 CUDA 流同时做相同的任务，而是从另一个角度入手解决问题，即对任务进行分解。流水线技术是最为常用的任务并行技术，即把一个大的解调任务分解为多个小任务，不同任务模块交由不同的核函数进行并行处理。如图 2 所示，先把数据交给第 1 个任务模块处理，当第 1 个模块输出的数据量满足第 2 个任务模块的输入数据量时，将调度执行第 2 个任务模块，依次类推。该方案解决了数据连续性问题，但是连续数据的多普勒误差累积会大大影响解调结果的正确率，所以任务并行方案对于遥测解调过程并不完全适用。

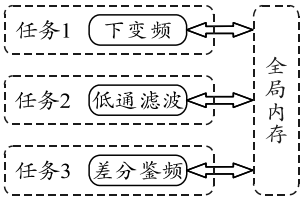


图 2　任务并行方案

考虑将数据并行和任务并行结合起来，首先根据数据特性将数据分解成小片段，并且分解的时候利用类似于重叠保留的方法，将每段数据的头部和尾部作少量重叠以保持数据的连续性和依赖性。然后对每段数据进行流水线式的任务分解，每个核函数完成一个任务模块的并行处理。该方案既解决了数据连续性问题，又解决了并行处理问题，可以用在基于高性能计算的遥测解调中。

由以上分析设计基于 GPU 的并行非相干鉴频算法实现结构如图 3 所示，数据解调流程为：1) 在数据预处理阶段，根据数据特性和计算资源数量将数据划分为 M 段，并将数据变为浮点数。2) 在每一个数据段内，根据频率生成 NCO 并计算多普勒频偏，然后执行并行下变频，将中心频率搬移到基带。3) 对下变频后的数据执行并行滤波。4) 最后进行并行差分鉴频并最终实现输出。

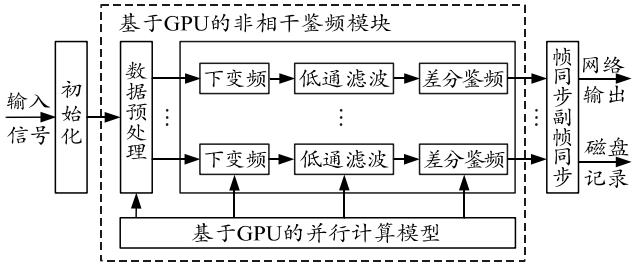


图 3　并行非相干鉴频算法流程

图 3 给出了基于 CUDA 架构的并行非相干鉴频算法实现结构，根据系统功能和算法情况，数字下变频模块、滤波模块和差分鉴频模块适合并行化，所以在 GPU 上的运算过程可分为下变频、FIR 低通滤波和差分鉴频 3 个模块。3 个模块调度以及运算中间数据的传输由 CPU 控制和协调，另外一些不适合并行运算的部分如帧同步等模块也需在 CPU 上运行。

2　基于 GPU 的非相干鉴频解调算法研究

2.1　基于 GPU 的数字下变频算法设计

2.1.1　并行数字下变频模块设计

在遥测解调中，经过 ADC 采样后的中频信号的数据速率太快，直接进行中频解调处理的难度比

较大、解调性能也不理想；所以，需要经过数字下变频(digital down converter，DDC)将中频信号转换为基带信号。典型的 DDC 模块包括数控振荡器(numerical control oscillator，NCO)、下变频、滤波和采样抽取[17]，在下变频之后使用低通滤波器选取所需的基频信号来滤除高频信号。滤波之后进行抽取来降低数据速率，之后在基带进行解调处理。本节主要介绍 GPU 并行下变频算法的设计与实现。输入的中频信号经过预处理后转化为浮点数，并且分成了 M 段，需要对每段信号进行 DDC，将信号搬移到基带。DDC 运算流程如图 4 所示。

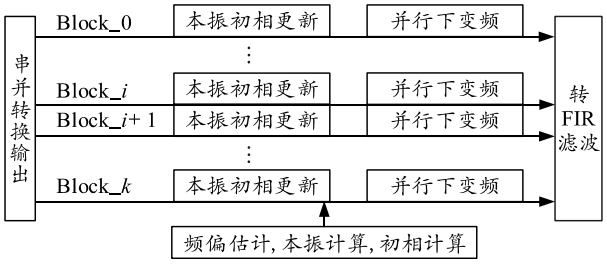


图 4　数字下变频运算流程

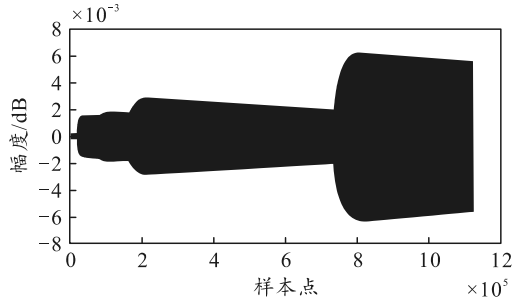
由上图可知，DDC 模块运算流程如下：

1) 根据数据预处理结果，启动 k 个 Block(线程块)，确保 k 个 Block 可以处理一段数据；

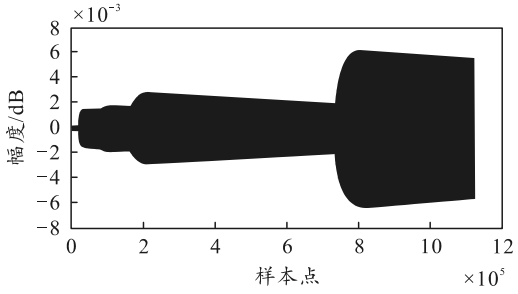
2) 根据接收信号的载波中心频率以及频偏估计的结果，生成并行数字本振，并在下变频之前对本振的初始相位进行更新；

3) 在步骤 2)的基础上，在每一个 Block 内对信号并行数字下变频处理，并将输出的基带信号交给 FIR 低通滤波模块处理。

为验证数字下变频模块设计的可行性，对中频 PCM/FM 遥测信号进行并行下变频仿真。中频信号的中心频率为 70 MHz，采样频率为 56 MHz，码速率为 2 Mbps。分析 20 ms 数据，长度为 1.12×10^6。利用数字 NCO 产生本振信号，将本振信号与中频信号混频以实现下变频。将 GPU 下变频后的信号幅度和 Matlab 下变频后的信号幅度进行对比，结果如图 5 所示。



(a) I 路变频误差



(b) Q 路变频误差

图 5　GPU 和 Matlab 下变频后幅度误差

从仿真结果可以看出：利用 GPU 浮点数进行下变频的结果和 Matlab 的运算结果相比有误差，误差量级在 10^{-3} 左右，并且有慢慢增大的趋势。造成这种误差的主要因素是 NCO 相位浮点数累加运算误差的累积效应。为了解决这一相位累加误差的累积问题，将深入研究数字 NCO 在 GPU 平台上的实现方法。

2.1.2　并行数字 NCO 实现算法设计

为减小舍入误差的累积，笔者在数据分段的前提下设计了相位循环消整周的算法。该方法灵活高效、结构简单。该算法的实现前提是数据要分段，每段数据内的数据点具有单独的索引值直接计算相位，而不进行相位累加，避免了舍入误差的累积。每段数据间进行相位累加运算时采用相位循环消整周的方法，把相位值限制在 2π 的范围内，这样进行累加的相位浮点数相近，减小了舍入误差。算法流程如下：

1) 对转换为浮点数的调频遥测数据进行分段，设每段数据长度为 N，n=1, 2, 3, …, N，共分为 nSeg 段；

2) 为避免误差累计不进行相位累加，而直接使用公式 $\varphi(n)=2\pi f_0 n/f_s$ 逐点计算第一段数据相位值；

3) 完成第一段数据的计算后，将最后一个点的相位值做消整周处理，也就是

$$\Delta\varphi(N)=\varphi(N)-\lfloor\varphi(N)\rfloor=2\pi f_0 N/f_s-\lfloor 2\pi f_0 N/f_s\rfloor。\quad(1)$$

式中 $\lfloor\bullet\rfloor$ 为向下取整。

4) 利用第 1 段消整周后的末尾相位值，计算第 2 段数据的初始相位值。则初始相位值为：

$$\varphi(N+1)=\Delta\varphi(N)+2\pi f_0/f_s。\quad(2)$$

在此基础上，类比第 1 段数据相位值的运算，逐点计算第 2 段数据的相位值。

$$\varphi(N+n)=\Delta\varphi(N)+2\pi f_0 n/f_s。\quad(3)$$

5) 由此可以推算出第 i, (i=1, 2, 3, …, nSeg) 段

数据的相位值为：

$$\varphi((i-1)N+n)=\Delta\varphi((i-1)N)+2\pi f_0 n/f_s \text{。} \quad (4)$$

由以上方法流程可以看出，本方法在每段数据最后一个点进行相位值的消整周。这样保证了相位值大小相差不大，减小了浮点数相位运算的舍入误差；在每段数据内，相位值有单独的索引直接计算而不进行相位累加，减小了误差积累。在相位循环消整周算法优化的基础上，基于 GPU 的数字 NCO 下变频实现流程如图 6 所示。

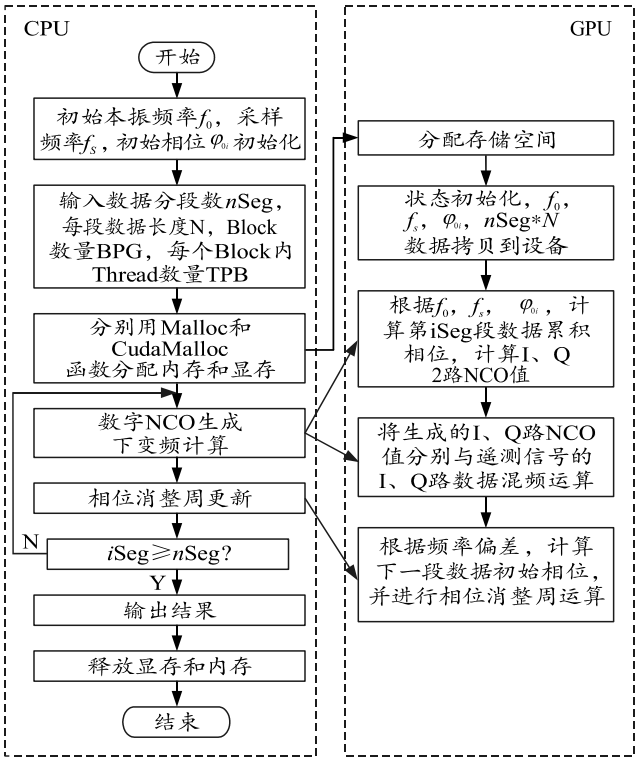


图 6　基于 GPU 的数字 NCO 下变频流程

上图中：f_0 为本振频率；f_s 为数据采样频率；遥测数据被分为 nSeg 段，每段长度为 N；φ_{0i} 为本振初相。在初始化阶段，完成数据初始化，分别用 CudaMalloc 和 Malloc 函数分配显存和内存，将调频遥测数据从 CPU 拷贝到 GPU。GPU 在 CPU 的调度控制下，调用核函数计算累加相位值、实时计算并输出 I、Q 2 路本振信号、实现下变频运算。每一段数据的下变频计算完成后，根据频率偏差 f_d 和相位消整周算法计算下一段数据的初始相位。在 nSeg 个数据片段计算完成后，将下变频结果从 GPU 显存拷贝到 CPU 内存中，然后释放内、显存，结束计算。

在运算过程中，设置 CUDA 线程块和线程为 1 维，线程块大小为 TPB，网格大小为 BPG = $\lceil (N+\text{TPB}-1)/\text{TPB} \rceil$，保证每个线程处理一个数据点的 NCO 本振信号的产生和下变频的实现。CUDA 分配的线程块以及每个线程的计算如图 7 所示。每段数据的相位 $\varphi((i-1)N+n)$ 利用单独的索引值直接计算，并且加上 $\Delta\varphi((i-1)N)$，$\Delta\varphi((i-1)N)$ 为经过消整周处理的上段末位数据的相位值。在本段数据计算完毕，输出经过消整周处理的本段末位数据的相位值 $\Delta\varphi(iN)$。

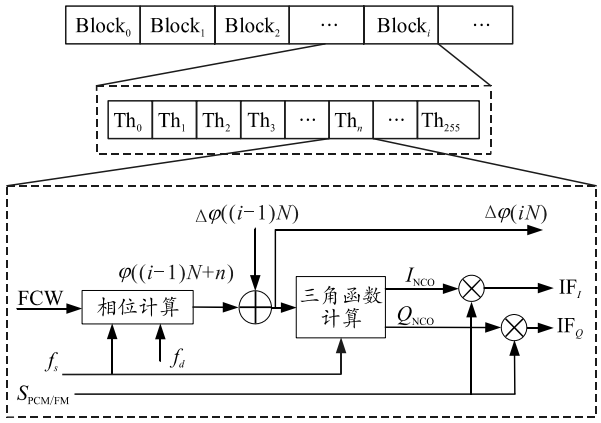


图 7　基于 CUDA 的 NCO 下变频实现

2.2　基于 GPU 的并行滤波方法研究

2.2.1　时域滤波和频域滤波

FIR 滤波有时域滤波和频域滤波 2 种实现方式，利用重叠保留法进行时域滤波和频域滤波时，算法的复杂度在滤波阶数和数据长度不同时有很大差别。不管是时域卷积还是频域 FFT 运算，都可将其看作多次乘加运算。在乘加运算中，乘法运算比较耗时，所以就以 2 种滤波方法中乘法运算的次数为基准，分析在滤波阶数和数据长度不同时算法的复杂度情况。

时域滤波过程可表示为输入数据和滤波系数的线性卷积关系。设输入数据长度为 L，FIR 滤波器系数长度为 M，则时域卷积运算时乘法的次数为 $2ML$。但每段数据参与计算前会与 M−1 位前一段数据拼接起来，这样每次计算就多了 $M(M-1)$次乘法计算。则采用重叠保留法进行时域滤波运算时总共需要 S 次乘法运算，S 为：

$$S=2ML+M(M-1)=M(2L+M-1) \text{。} \quad (5)$$

由上式可以看出，当滤波器长度 M 远小于数据长度 L 时，乘法运算次数 $S\approx 2ML$ 与数据长度 L 成线性关系。当滤波器长度 M 与数据长度 L 大小相近时，乘法运算次数 $S\approx 3L^2$ 与数据长度 L 的平方成线性关系。所以，时域滤波算法在滤波器阶数较小的情况下更有优势。

根据傅里叶变换可知，2 个信号时域的卷积等于这 2 个信号频域的乘积，即可写出 FIR 滤波器的频域计算表达式：

$$Y(k)=X(k)H(k),\ k=0,1,2,\cdots,N-1。\tag{6}$$

式中：$N\geqslant L+M-1$；$X(k)$是输入数据 $x(n)$的离散傅里叶变换(DFT)；$H(k)$是滤波器系数 $h(n)$的离散傅里叶变换；$Y(k)$是滤波结果的频域表达式，其值与 $y(n)$的离散傅里叶变换相同，所以对 $Y(k)$求离散傅里叶反变换(IDFT)可得卷积变换结果 $y(n)$：

$$y(n)=x(n)*h(n)=\mathrm{IDFT}[Y(k)]=\mathrm{IDFT}[X(k)H(k)],$$
$$n=0,1,2,\cdots,N-1;\ \ k=0,1,2,\cdots,N-1。\tag{7}$$

用重叠保留法将输入数据 $x(n)$进行分段，则每段长度为 $L+M-1$。对输入数据 $x(n)$做 FFT 运算，$X(k)=\mathrm{DFT}[x(n)]$。同时也对滤波器系数 $h(n)$做同样点数的 FFT 运算，$H(k)=\mathrm{DFT}[h(n)]$，则 2 次 FFT 运算需要的乘法运算的次数为：

$$S=4\times\begin{bmatrix}(L+M-1)/2\times\log_2(L+M-1)+\\(L+M-1)/2\times\log_2(L+M-1)\end{bmatrix}=$$
$$4\times\left[(L+M-1)\log_2(L+M-1)\right]。\tag{8}$$

信号和滤波器系数经过 FFT 运算变到频域之后，需要做乘积运算，乘法运算的次数为 $L+M-1$，得到乘积结果的个数为 $L+M-1$。然后对乘积结果做 IFFT 运算，需要的乘法次数为 $2\times[(L+M-1)\log_2(L+M-1)]$。则采用重叠保留法进行频域滤波运算时总共需要的乘法运算次数为：

$$S=6\times\left[(L+M-1)\log_2(L+M-1)\right]+(L+M-1)=$$
$$(L+M-1)\left[6\log_2(L+M-1)+1\right]。\tag{9}$$

由上式可看出，当滤波器长度 M 远小于数据长度 L 时，乘法运算次数 $S\approx L[6\log_2 L+1]$。当 M 与 L 大小相近时，乘法运算次数 $S\approx 2L[6\log_2(2L)+1]$。所以，频域滤波算法在滤波器阶数较大的情况下比时域滤波算法更有优势。

将时域滤波算法转换到频域滤波是为了回避计算复杂度较高的卷积运算，在频域滤波过程中利用快速 FFT 运算的优势，可在滤波器阶数较高和输入数据点数较大的运算中取得明显的加速效果。频域滤波算法将原本只有简单乘加运算的时域卷积滤波算法转变成数据扩充、2 次 FFT 运算、一次点乘运算和一次逆傅里叶变换。这无疑把原本简单的运算流程变得更加复杂。在滤波器阶数比较低时，这种流程上的复杂性将会大大降低算法的性能。与之相反的是，时域滤波算法在滤波器阶数比较低时，有更低的运算复杂度，且运算流程较为简单；在滤波器阶数比较高时，运算的复杂度大大增加。因此，在滤波器阶数比较高时，频域滤波算法具有更大的优势；在滤波器阶数比较低时，时域滤波算法具有更高的效率。

2.2.2 基于 GPU 的时域 FIR 滤波算法

下变频后的 FIR 滤波器阶数较低，所以选择时域滤波算法。在利用重叠保留法进行数据分段后，将进行时域滤波运算，而时域 FIR 滤波运算的本质就是将输入信号与滤波器系数做卷积运算。在 GPU 中对输入信号进行滤波实际上就是将滤波器系数作为加权值，对输入数据进行加权求和运算。基于 GPU 的并行时域 FIR 滤波实现流程如图 8 所示。

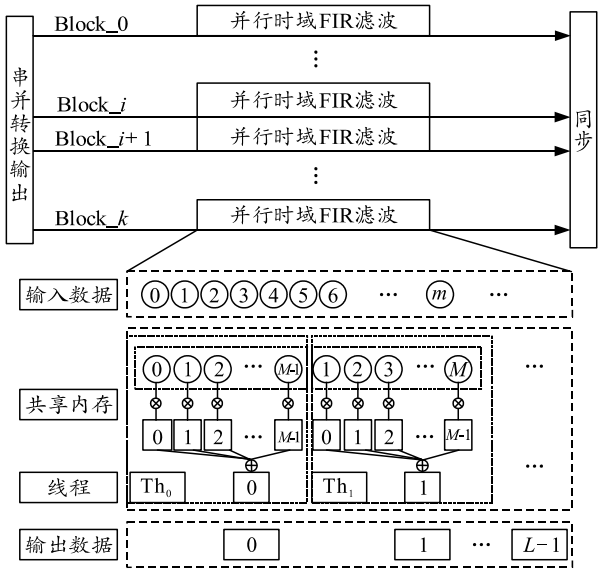


图 8 基于 GPU 的并行时域 FIR 滤波实现流程

由上图可知，基于 GPU 的并行时域 FIR 滤波实现流程如下：

1) 按照重叠保留法对数据进行分段处理，每段数据的长度为 L，滤波器系数长度为 M；

2) 根据数据分段结果，启动 k 个 Block，确保 k 个 Block 刚好可以处理一段数据；

3) 将滤波器的系数和扩展后长度为 $L+M-1$ 的数据存入共享内存，以加快内存访问速率；

4) 用滤波器系数将输入数据进行加权求和，每个线程输出一个滤波结果；

5) 将滤波结果存入显存，待计算完成将会输出本段数据的滤波结果 $y_i(n)$，长度为 L。

2.3 基于 GPU 的并行差分算法

解出符号间的相位差是调频遥测信号非相干解调的重要步骤。常用的运算方法有一阶差分鉴频解调和叉积鉴频解调 2 种[4]。叉积鉴频最终的求解公

式实际是三角函数的近似，要提高鉴频精度，就需要减小采样间隔也即提高采样频率。这样的求解思路又和降低解调速率相矛盾，所以鉴频的精度被限制在有限的相位范围内[18]。为提高鉴相的精度，通常采用 VOLDER[19]在 1959 年提出的 CORDIC 算法进行求解反三角函数，但是 CORDIC 算法求解反正切的运算过程需要多次对向量旋转累加，实现起来相对复杂。GPU 有专门计算超越函数的特殊函数单元(special function unit，SFU)[20]，可以直接快速地计算反正切函数；因此，首先求解反正切得到相位信息，然后把求得的相位信息进行差分运算就会得到鉴频结果。

直接利用反正切函数 atan()来求解相位信息简单高效，但受到反正切函数值域$(-\pi/2, \pi/2)$的限制，求出结果需要做相位扩展才能得到真正的相位信息。CUDA C 语言中有个函数 atan2()，原型为 double atan2(double y, double x)，返回值为弧度，表示的是 y/x 的反正切。atan2()是 atan()的增强版，值域为$(-\pi, \pi]$，利用 atan2()可直接求得真正的相位信息。根据差分鉴频原理可知，经过鉴相求得相位信息后，对求得的相位进行差分就可求得调制信号。差分算法的指令简单、复杂度低，而且每 2 个相位数据的差分结果与其他数据没有依赖性，因此适用于 GPU 多线程进行并行处理。

差分鉴频算法在 GPU 上实现的流程如图 9 所示。

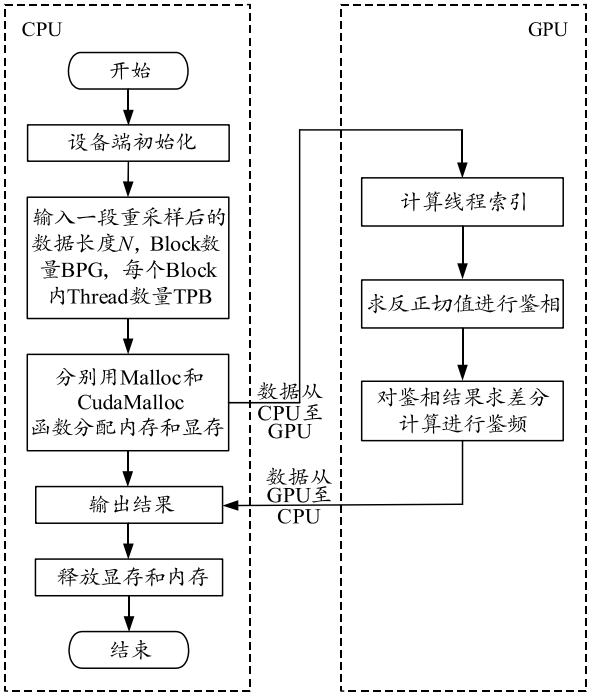


图 9　差分鉴频优化后的 CUDA 实现流程

在介绍并行数字 NCO 时，进行了较为详细的流程描述，设备的初始化、分配以及释放显存和内存都是相同的。接下来，只重点介绍核函数在 GPU 上的实现方式。在 GPU 上进行优化后差分鉴频运算的核函数实现方式如图 10 所示。

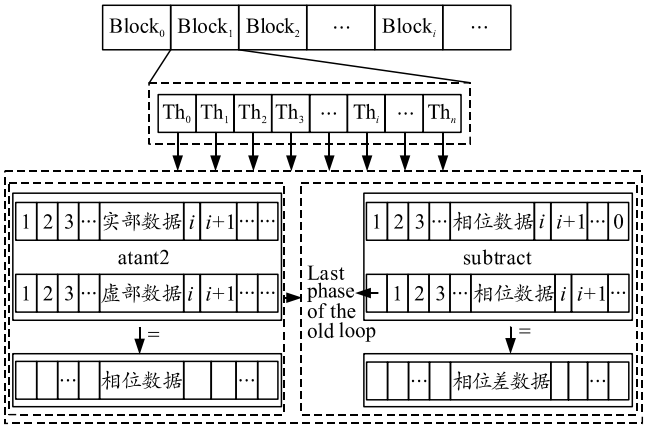


图 10　差分鉴频优化后的核函数实现

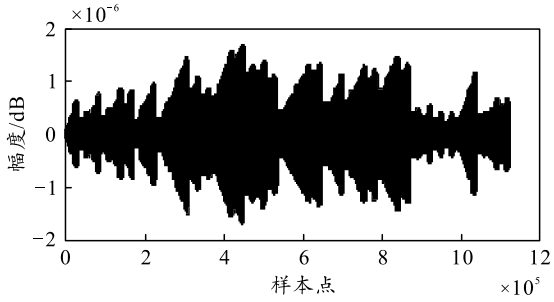
长度为 N 的重采样后的数据传输至 GPU 后，在 GPU 上发射一个核函数用于反正切和差分运算实现鉴频。GPU 上的线程以及线程块均为 1 维，假设每个线程块中线程个数为 TPB，则线程块个数根据数据长度进行计算。由于每段数据长度为 N，线程块个数计算公式为$[(N+\text{block}.x-1)/\text{block}.x]$。通过开启的线程同时并行的进行反正切和差分运算，每个线程拥有一个唯一的索引值，完成一位数据的反正切之后再进行差分运算。

3　实验验证

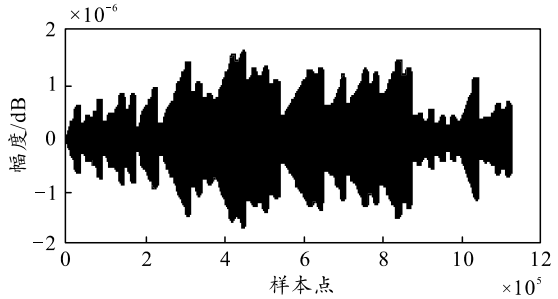
3.1　各模块实验验证

3.1.1　基于 GPU 的数字下变频算法实验验证

为验证在数据分段的条件下相位循环消整周优化算法的正确性，对中频 PCM/FM 遥测信号进行并行下变频实验验证。中频信号的参数与 2.1 节相同，分析 20 ms 数据，长度为 1.12×10^6。将 GPU 下变频后的信号幅度和 Matlab 下变频后的信号幅度进行对比，结果如图 11 所示。



(a) I 路变频误差



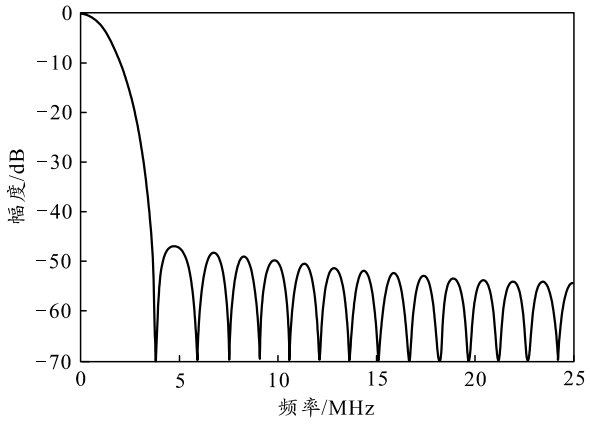
(b) Q 路变频误差

图 11 消整周后 GPU 和 Matlab 下变频的幅度误差

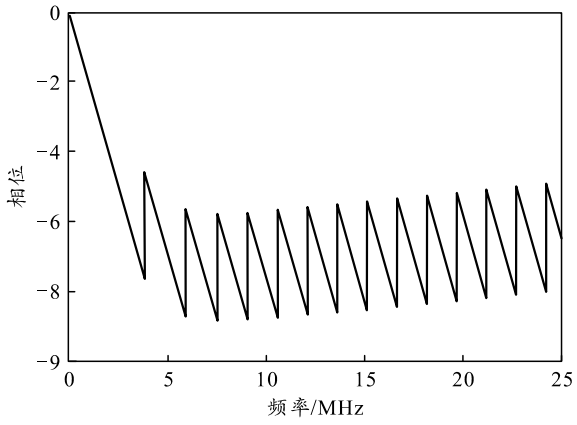
从上图可以看出，在相位循环消整周算法的优化下，下变频的计算误差大大减小，被压缩在 10^{-6} 量级，小于 10^{-5}，满足计算精度的要求。这也验证了消整周优化的有效性和正确性。

3.1.2 基于 GPU 的 FIR 时域滤波算法实验验证

按照图 8 时域 FIR 滤波流程进行仿真验证，在 Linux+CUDA11.1 开发环境下，使用 Tesla V100 GPU。下变频之后的滤波参数为：数据的采样速率为 56 MHz，码速率为 2 Mbps，滤波器阶数为 36。首先，利用 Matlab 的 fdatool 工具设计出 FIR 滤波器，FIR 滤波器的特性如图 12 所示。



(a) 幅频响应



(b) 相频响应

图 12 FIR 滤波器幅频响应和相频响应特性

按照重叠保留法进行数据分段，一秒钟数据被分为 2 000 段，每段数据为 0.5 ms，长度为 2.8×10^4。按照图 8 所示的流程在 GPU 上进行时域滤波处理，分别统计滤波运算用时和带数据拷贝的滤波运算用时，结果如表 1 所示。

表 1 时域 FIR 滤波用时统计对比 ms

次数	Matlab time	Kernel time	Kernel+ memcpy time	Kernel+ memcpy+36 HtDmemcpy time	Kernel+ memcpy+36 DtDmemcpy time
1	78.128	0.008 16	0.038 910	0.052 220	0.048 160
2	78.125	0.008 19	0.038 910	0.051 200	0.050 180
3	83.475	0.008 19	0.046 080	0.053 180	0.060 420
4	78.125	0.009 22	0.048 130	0.053 180	0.048 130
5	78.125	0.008 19	0.039 940	0.053 250	0.048 130
6	78.125	0.009 22	0.038 940	0.052 140	0.049 150
7	78.125	0.010 24	0.039 900	0.061 200	0.048 130
8	78.125	0.008 19	0.043 010	0.052 180	0.048 130
9	78.125	0.008 19	0.038 940	0.052 040	0.049 150
10	78.125	0.008 16	0.038 910	0.052 120	0.048 130
平均值	78.660	0.008 59	0.0411 67	0.0532 71	0.049 771
加速比	—	9 151.87	1 910.76	1 476.61	1 580.440

上表中：Matlab time 表示用 Matlab 进行时域滤波运算所用的时间；Kernel time 表示用 GPU 进行时域滤波运算所用的时间；Kernel+memcpy time 表示在滤波运算所用时间的基础上加上输入数据从内存传输到显存的时间；最后 2 列为在 Kernel+memcpy time 的基础上又加上了重叠保留法所扩展的 36 位数据的传输时间。

可以看出，若只计算滤波运算所用的时间，GPU 相比于 Matlab 取得了将近一万倍的加速比。由于 GPU 平台的运算不得不把数据从 CPU 内存传输到 GPU 显存的时间考虑在内，数据传输又是极其费时的，加上数据传输之后取得将近 2 000 倍的加速比。为了保证滤波运算的正确性，笔者采用重叠保留法进行数据分段，即每次又增加了 36 个数据点的传输。数据从 CPU 内存传输到 GPU 显存效率是极低的，且传输函数 cudaMemcpyHostToDevice 存在调用时延；因此，笔者采用将本段显存中的末尾 36 位数据拷贝到下一段运算显存的前 36 位的方法进行加速，将加速比从 1 476.61 倍提高到 1 580.44 倍。以上分析可以说明，对于时域 FIR 滤波算法，GPU 并行处理过程可以取得较为明显的加速效果，大大减小了算法运行时间。

3.1.3 基于 GPU 的差分鉴频算法实验验证

按照图 9 的算法实现流程进行反正切鉴相和差分鉴频验证，每段数据为 0.5 ms，重采样之后长度为 8×10^3。表 2 给出了算法优化后在 Matlab 和 GPU 上的运算时间对比。

表 2　差分鉴频算法优化后用时统计对比　　ms

次数	Matlab time	Kernel time	Kernel+ memcpy time
1	0.502 0	0.007 170	0.031 330
2	0.525 0	0.009 250	0.033 180
3	0.494 0	0.006 140	0.032 740
4	0.503 0	0.007 170	0.031 420
5	0.540 0	0.007 170	0.031 200
6	0.498 0	0.007 170	0.031 550
7	0.540 0	0.008 190	0.032 220
8	0.494 0	0.007 170	0.033 440
9	0.496 0	0.007 170	0.032 290
10	0.501 0	0.007 170	0.032 540
平均值	0.509 3	0.007 377	0.032 191
加速比	1.000 0	69.040 000	15.820 000

上表中：Matlab time 表示用 Matlab 进行反正切和差分运算所用的时间；Kernel time 表示用 GPU 进行反正切和差分运算所用的时间；Kernel+ memcpy time 表示在反正切和差分运算所用时间的基础上加上输入数据从内存传输到显存的时间。可以看出，若只计算差分运算所用的时间，GPU 相比于 Matlab 取得了将近 70 倍的加速比，加上数据传输之后也取得 15.82 倍的加速比。

3.2　总体性能实验验证

为对基于 GPU 的非相干鉴频解调算法加速性能进行实验验证，利用 PCM/FM 遥测基带设备产生多种速率的调频遥测信号。为降低实验过程中，时间统计数据极端值对实验结论的影响，笔者将使用截尾平均数法对多次实验数据求平均值作为最终结果。即直接将明显不正确的极值结果剔除，本实验取 20 次实验结果剔除 3 次。将数据分为 0.5 ms 每段，每段数据长度 2.8×10^4，采用真实的调频遥测基带产生实时 PCM/FM 遥测信号。信号中频 70 MHz，采样频率 56 MHz。

在并行 PCM/FM 遥测信号解调算法中，核心的运算环节采用了多线程并行方法对运算过程进行了加速，并针对 GPU 的多级存储结构的应用进行了优化，运算过程使用共享内存和锁页内存使得运算效率得到了较大的提升。为了实现调频遥测信号的实时解调，1 s 的信号要求解调过程的运算时间不超过 1 s。为测试系统的实时性表现，对系统在不同码速率以及不同线程块大小下的运算时间进行了测试。系统运行时间结果如图 13 所示。

图 13 显示了线程并行模式下，不同码速率和不同线程块大小的 PCM/FM 遥测信号非相干鉴频解调全流程的运算时间，以及 Matlab 串行解调程序在不同码速率下的运算时间。从图中可以看出，随着码速率的增大，解调的运算量逐渐增大，Matlab 串行程序的运行时间逐渐增加，并且增速比较快，从 1 Mbps 时的 242.219 s 增加到 10 Mbps 时的 417.548 s；基于 GPU 的并行解调算法的运行时间也在增加，但增加的幅度比较小，从 1 Mbps 时的 351.512 ms 增加到 10 Mbps 时的 392.543 ms。

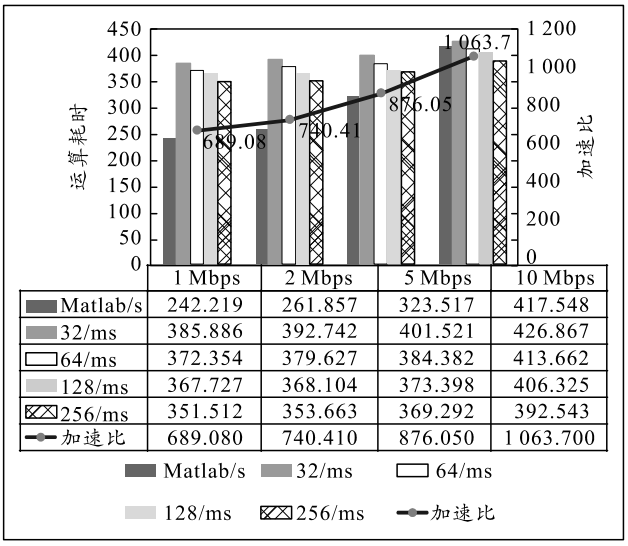


图 13　并行鉴频解调运算时间对比

本实验测试了线程块大小为 32、64、128、256、512 时，并行解调程序在不同码速率下的用时。因为参数和数据过多，在线程块大小为 512 时线程块内的内存会溢出，所以未统计线程块大小为 512 时的时间。可以看出，随着线程块大小的增加，并行解调算法用时逐渐减小，在线程块为 256 时达到最优值。在本实验条件下，不同速率的并行算法用时最长时间为 426.867 ms，因此，笔者设计的并行解调算法完全满足信号实时解调的要求。

误码率测试结果如图 14 所示。

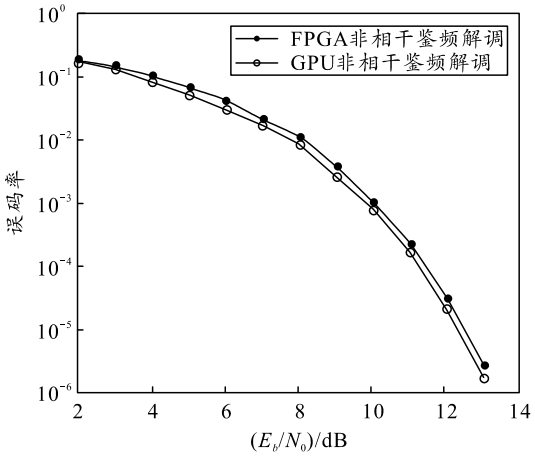


图 14　误码率测试结果

由上图可以看出，在误码率为10^{-4}的情况下，基于 FPGA 实现的非相干鉴频算法的 E_b/N_0 为

11.4 dB，基于 GPU 的并行非相干鉴频解调算法的 E_b/N_0 为 11.25 dB，比 FPGA 有 0.15 dB 的增益。可见，基于 GPU 的并行解调算法的性能与 FPGA 的误码率曲线近似，并且 GPU 解调增益稍优于 FPGA。

4 结束语

笔者提出将数据并行和任务并行结合起来的并行方法，设计了基于 GPU 实现解调的流程。分别从并行数字下变频、并行 FIR 滤波和并行差分鉴频 3 部分研究了具体模块的并行化实现方法。

针对 NCO 相位累加运算使得误差出现累积效应的问题，基于循环消整周的原理设计出数字 NCO 在 GPU 平台上的实现方法，并进行了实验验证，成功将运算误差由 10^{-3} 量级降低至 10^{-6} 级。分析了基于重叠保留法的时域滤波算法和频域滤波算法，分别对 2 种滤波方法在不同条件下的性能进行仿真，并给出滤波阶数较低的最佳 FIR 时域滤波方法，最终对 36 阶 FIR 时域滤波方法进行了验证，取得了 1 580.44 倍加速比。提出了基于 GPU 的差分鉴频算法并行处理方法并给出了优化策略，最终将加速比提高至 69.04 倍。

通过对算法进行实验验证，不同速率的并行算法用时最长时间为 426.867 ms，证明该算法完全满足信号实时解调的要求。基于 GPU 的并行解调算法与基于 FPGA 算法的误码率曲线近似，并且 GPU 解调增益稍优于 FPGA。

参考文献：

[1] 赵妍婷. PCM/FM 遥测遥控接收系统[D]. 秦皇岛: 燕山大学, 2012.

[2] 高山. 一种高动态环境下 PCM-FM 的低信噪比检测方法[J]. 遥测遥控, 2020, 41(2): 27-31.

[3] ZHOU Y, DUAN R, JIANG B. A Low-Complexity Implementation Scheme for PCM/FM Based on MLSD[C]// 2019 15th International Wireless Communications and Mobile Computing Conference (IWCMC). 2019.

[4] 杨毅. PCM/FM 遥测接收机基带信号处理技术研究与实现[D]. 重庆: 重庆大学, 2017.

[5] RICE M. PCM/FM aeronautical telemetry in frequency selective multipath interference[J]. IEEE Transactions on Aerospace and Electronic Systems, 2000, 36(4): 1090-1098.

[6] 成亚勇, 闫冬, 孙晓锋, 等. 基于 GPU 实现的调频遥测解调方法[J]. 无线电工程, 2016, 46(4): 35-38.

[7] 翁涛. 基于 GPU 架构的 OFDM 波形外辐射源雷达信号处理并行化设计及实现[D]. 南昌: 南昌大学, 2020.

[8] TIAN Q Y, XU C Y, ZHAO Q. Signal Processing of Software Radar Based on GPU[J]. Shipboard Electronic Countermeasure, 2020, 43(1): 58-63, 100.

[9] 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.

[10] 田乾元, 徐朝阳, 赵泉. 基于 GPU 的软件雷达信号处理[J]. 舰船电子对抗, 2020, 43(1): 58-63, 100.

[11] 托乎提努尔, 张海龙, 王杰. 基于 CUDA 的射电天文多相滤波器组设计[J]. 天文研究与技术, 2017, 14(1): 117-123.

[12] 张海龙, 张萌, 聂俊, 等. 脉冲星数字终端技术综述[J]. 中国科学: 物理学 力学 天文学, 2019, 49(9): 19-32.

[13] KIM S C, BHATTACHARYYA S S. Implementation of a Multirate Resampler for Multi-carrier Systems on GPUs[J]. Journal of Signal Processing Systems, 2017, 89(3): 445-455.

[14] 孙坚武. 遥测遥控信号分析算法与 GPU 实现技术研究[D]. 成都: 电子科技大学, 2020.

[15] 胡蕊. 基于 GPU 并行计算的实时谱处理软件设计[D]. 成都: 电子科技大学, 2019.

[16] 赵仁良. 航天遥测信号高性能解调算法与系统研究与实现[D]. 北京: 中国科学院大学计算机应用技术, 2014.

[17] 吴升. 基于 FPGA 的遥测接收机解调技术的设计与实现[D]. 重庆: 重庆邮电大学, 2018.

[18] 张飞. PCM/FM 数字遥测接收机基带模块设计与实现[D]. 成都: 电子科技大学, 2016.

[19] 卢欣, 吴中川, 杨春, 等. PCM/FM 调制器的设计仿真与实现[J]. 通信技术, 2009, 42(11): 16-18.

[20] NVIDIA. Cuda C++ Programming Guide[EB/OL]. (2021-05-20)https://docs.nvidia.com/cuda/pdf/CUDA_C_programming_guide. pdf.