引用格式：罗玲, 崔峻豪, 王基岙, 等. 基于缓存区搬移策略的 GPU 高效信道化方法 [J]. 电子科技大学学报, 2026, 55(2): 184-190.
LUO L, CUI J H, WANG J A, et al. An efficient channelization method on GPU with buffer shift strategy[J]. Journal of University of Electronic Science and Technology of China, 2026, 55(2): 184-190.

基于缓存区搬移策略的 GPU 高效信道化方法

罗 玲[1]，崔峻豪[2]，王基岙[2]，陈章鑫[2*]

（1. 中国西南电子技术研究所复杂航空系统仿真全国重点实验室，成都 610036；2. 电子科技大学信息与通信工程学院，成都 611731）

摘要：信道化处理是现代电子战数字系统中的首要任务。基于 CPU 的信道化处理以多相滤波为基础，使用分段卷积的方式保证信道化结果的相位连续。然而，随着数据量的增大，该方法无法满足实时处理的要求，研究如何基于 GPU 实现高性能信道化处理是目前急需解决的问题。首先分析了传统分段卷积方法在 GPU 架构上的低效性，随后结合 GPU 架构特点提出了一种缓存空间需求更低、计算量更低、逻辑控制更方便的缓存区搬移策略来保证信道化结果的相位连续性。此外，分析了在 GPU 架构下基于多相滤波和直接滤波的两种多级滤波方式，说明了 GPU 架构下采用直接滤波方式的优越性。仿真实验表明所提方法能正确保证信道化结果的相位连续性，给出了 GPU 和 CPU 下实现直接滤波的加速比，直观说明了基于 GPU 的高效信道化方法带来运算效率的巨大提升，同时 GPU 架构下直接滤波快于多相滤波。所提的基于缓存区搬移策略的 GPU 高效信道化方法在处理速度和数据相位连续性上具有显著优势，尤其适用于大规模数据的实时处理应用。

关键词：信道化处理；GPU；相位连续性；多相滤波；直接滤波
中图分类号：TN911 **文献标志码：**A DOI: 10.12178/1001-0548.2024327

An efficient channelization method on GPU with buffer shift strategy

LUO Ling[1], CUI Junhao[2], WANG Ji'ao[2], and CHEN Zhangxin[2*]

(1. National Key Laboratory of Complex Aviation System Simulation, Southwest China Institute of Electronic Technology, Chengdu 610036, China;
2. School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract: Channelization processing is the first task in modern electronic warfare digital systems. CPU-based channelization processing is based on polyphase filtering and segmented convolution to ensure phase continuity of the channelized results. However, as the increasing of data amount, it cannot meet the requirements of real-time processing. The study on how to implement high-performance channelization processing based on GPU is currently an urgent issue that needs to be addressed. The inefficiency of traditional segmented convolution methods on GPU architecture is first analyzed, then a buffer shift strategy is proposed to ensure the phase continuity of channelization results with lower buffer space, reduced computational overhead and simplified logic control. Additionally, the analysis of two multistage filtering approaches, i.e., polyphase filtering and direct filtering, demonstrates the superiority of direct filtering on GPU architecture. Simulation experiments show that the proposed method correctly ensures the phase continuity of the channelization results and confirms that direct filtering is faster than polyphase filtering on GPU architecture. The acceleration ratios of direct filtering using GPU and CPU are provided to illustrate the substantial improvement in computational efficiency by the proposed strategy. In a word, the proposed buffer shift strategy-based GPU-efficient channelization method has significant advantages in processing speed and data phase continuity and is particularly suitable for real-time processing of large-scale data.

Key words: channelization processing; GPU; phase continuity; polyphase filtering; direct filtering

信道化技术是电子战系统中提高频谱利用效率、增强目标检测能力以及降低多径干扰的重要手段[1]。信道化技术通过滤波器组将宽带数字信号分割成多个低速率子频带信号，从而实现频域信道化和抽取操作[2]。不同的硬件平台被应用于信道化的实现中。CPU 的信道化实现通常采用串行方式来

收稿日期：2024−12−01
作者简介：罗玲，博士，高级工程师，主要从事机载嵌入式系统、机载云服务与高性能处理方面的研究。
*通信作者 E-mail: zhangxinchen@uestc.edu.cn

完成信道化任务，在多信道的情况下效率较低，难以满足实时性要求[3-4]。在 FPGA 平台上，信道化处理是实时的，可采用基于频率响应屏蔽结构[5-6]及改进系数抽取方法的可变数字滤波器设计[7-8]，减少 FPGA 上的计算复杂度；但由于 FPGA 的特点，信道数量和参数是固定的，复杂信号处理需要将任务分配到多个 FPGA 中实现[9]。在 GPU 平台上，GPU 的并行计算能力可以加速信道化过程，信道数量和参数可以灵活配置，是高性能信道化实现的重要方式[3-4, 10]。

滤波抽取是信道化技术中的一个关键步骤，用于降低采样率和减少数据量，从而减轻后续处理的计算负担[6, 11]。为了实现任意抽取倍数，通常采用多级滤波抽取的方法，每一级滤波抽取使用一个低阶低通滤波器和较小的下采样因子，从而提高计算效率并减少滤波器设计的难度。这种方法能显著提高计算效率，减少滤波器的复杂度，并降低计算和存储资源的需求。无论是 CPU 还是 GPU 实现方式，在信号抽取过程中，需要将数据分包传输，再由 CPU 或者 GPU 对每个数据包进行处理，如果没有做预先处理，分包卷积结果与原始连续信号卷积结果相比，数据包分割处的相位是不连续的[12]。相位不连续会导致信号失真，影响后续解调、测量等处理的准确性。

为了解决这一问题，本研究基于 GPU 平台，给出了一种基于缓存区搬移策略的 GPU 高效信道化方法，用于解决传统架构在处理信道化任务时效率低下的问题。基于该缓存区搬移策略的信道化方法在滤波抽取过程中有效地保证了相位连续性，避免了传统分段卷积带来的信号失真。同时，分析对比了 GPU 架构下的多相滤波方式和直接滤波方式，给出了多相滤波不适合于 GPU 架构的原因。最后，通过实验说明了本文方法的有效性并给出了计算效率提升的统计结果。

1　分包滤波抽取中的相位连续性问题

1.1　相位连续性问题原理

在将待滤波数据分包传输场景下，原本由滤波器对完整待滤波数据进行信道化的操作被变为多个短序列的滤波操作，此过程可由图 1 描述。

图 1 以 4 包输入数据为例，分别按时序进入信道化处理模块。图中，数字 1、2、3、4 分别表示数据所在包编号。在理想情况下，信道化模块需要使用滤波器对 4 包数据组成的整体数据进行多级滤波抽取。然而，对整体数据进行处理在实际中是不可行的。具体来说，信道化模块需要获得所有的输入序列，并完全进行处理后，输出序列才是有效的，因此整个系统必然存在较大的延时，不能符合实时处理的要求，在采样率高或者通道数多的场合需要极其庞大的缓存区。

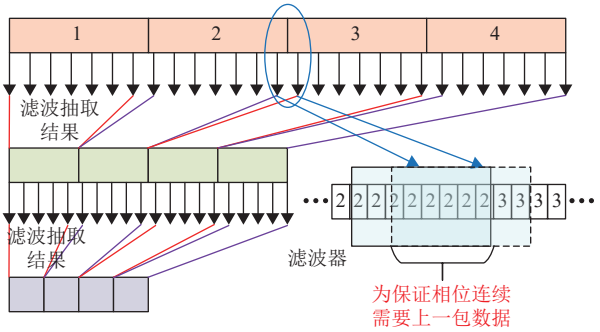


图 1　分包滤波抽取中的相位不连续原理示意图

因此，需要对整体数据进行分包处理，每一包数据通过多级滤波抽取再合并拼接得到最终的结果数据。然而，不能对每包数据进行独立的滤波抽取操作，否则会存在相位不连续问题。以数据 3 为例，可以看到在对数据 3 的起始部分进行滤波抽取时，为了保证滤波后的结果数据相位连续，如图中虚线所示，需要用到数据 2 中的部分数据。所以在设计信道化算法时，不能仅考虑对每一包输入数据做独立的信道化处理，还需要考虑之前的输入数据。

现有的保证抽取滤波后相位连续的算法都是基于 CPU 架构提出的，采用分段卷积的方式，根据分段方式的不同分为重叠相加法和重叠保留法。

1.2　重叠相加法

重叠相加法将输入信号分段并在每段数据后面补零进行线性卷积，然后将前一段卷积结果后面与后一段起始的重叠部分进行累加，其过程如下。

给定两个输入序列：输入数据序列$\boldsymbol{x}(n)$，长度为N，滤波器冲激响应序列$\boldsymbol{h}(n)$，长度为M。假设L是分段的长度，满足$L \geqslant M$。将$\boldsymbol{x}(n)$分为多个子段，每个子段长度为L，并在计算时补$M-1$个零以进行线性卷积。将$\boldsymbol{x}(n)$表示为K个子段的和：

$$\boldsymbol{x}(n)=\sum_{i=0}^{K-1}\boldsymbol{x}_i(n) \tag{1}$$

式中，$\boldsymbol{x}_i(n)$为第i个长度为L的子段。

将补零部分卷积得到的$M-1$个结果与后续序列卷积得到$M-1$个结果相加，由于：

$$\boldsymbol{x}_i(n)*\boldsymbol{h}(n)+\boldsymbol{x}_{i+1}(n)*\boldsymbol{h}(n)=$$
$$\sum_{k=-\infty}^{\infty}\boldsymbol{x}_i(k)\boldsymbol{h}(n-k)+\sum_{k=-\infty}^{\infty}\boldsymbol{x}_{i+1}(k)\boldsymbol{h}(n-k)=$$
$$\sum_{k=-\infty}^{\infty}(\boldsymbol{x}_i(k)+\boldsymbol{x}_{i+1}(k))\boldsymbol{h}(n-k)=$$
$$(\boldsymbol{x}_i(k)+\boldsymbol{x}_{i+1}(k))*\boldsymbol{h}(n) \tag{2}$$

式中，“+”表示重叠部分相加，非重叠部分合并。由于前一个序列补的是$M-1$个零，因此分段卷积的输出与完整数据的线性卷积的结果相一致，在信号边界处保证了相位的连续性。

1.3 重叠保留法

重叠保留法通过将输入信号分段并在每段数据前面添加前一段数据最后$M-1$个数据进行线性卷积，保留每个卷积段结果中的有效部分拼接出最终结果，具体过程如下。

假设滤波器冲激响应长度为M，分段长度为L，第一个子段前补$M-1$个零点，记为：

$$[0,0,\cdots,0,\boldsymbol{x}_0,\boldsymbol{x}_1,\cdots,\boldsymbol{x}_{L-1}] \tag{3}$$

从第二个子段开始，段前添加与上一子段邻接的$M-1$个点，记为：

$$[\boldsymbol{x}_{(iL-(M-1))},\boldsymbol{x}_{(iL-(M-1)+1)},\cdots,\boldsymbol{x}_{(iL)},\boldsymbol{x}_{(iL+1)},\cdots,\boldsymbol{x}_{(iL+L-1)}] \tag{4}$$

对每个添加了重叠部分的子段进行长度为$L+M-1$的循环卷积，得到卷积结果$\boldsymbol{y}_i(n)$。对于每个卷积结果$\boldsymbol{y}_i(n)$，将每段前$M-1$个产生混叠的结果舍弃，保留后L个点，即保留：

$$\boldsymbol{y}_i(M-1:L+M-2) \tag{5}$$

将保留部分相连，通过观察第i个子段卷积结果的开头$\boldsymbol{y}(iL)$，如下：

$$\boldsymbol{y}(iL)=\sum_{k=iL-M+1}^{iL}\boldsymbol{x}_i(k)\boldsymbol{h}(iL-k) \tag{6}$$

式 (6) 包含了$\boldsymbol{x}_{(iL-(M-1))}$到$\boldsymbol{x}_{(iL)}$的数据，结果与完整数据的线性卷积的结果相一致，且保证了相位的连续性。

2 基于 GPU 的高效信道化方法

面对数据量越来越大的实时处理要求，基于 CPU 架构的信道化方法已经完全无法适用，而 GPU 由于其强大的并行计算能力受到广泛关注。因此，如何基于 GPU 实现高效信道化处理是一个急需解决的问题。

2.1 传统算法应用在 GPU 架构下的问题

在实际滤波抽取过程中，抽取倍数往往较大（例如为 1 024），需要进行多级滤波。对于重叠相加法和重叠保留法，这个过程可以描述为：假设采用N级滤波，则对于第T包输入数据进行第n级滤波抽取后还需要累加上第$T-1$包输入数据的第n级滤波抽取的部分结果，才能进行第$n+1$级滤波抽取。因此这两种算法需要的缓存空间与滤波级数成正比，且在每一级滤波后还需要进行一次数据拼接后才能进行下一级滤波。在 GPU 架构下这种操作会严重影响整体运算效率，导致信道化处理速度严重下降。

为此，本文设计了一种基于 GPU 的高效信道化方法，该方法能在保证运算速度的前提下使用简单的逻辑控制策略保证滤波抽取结果的相位连续。

2.2 基于缓存区搬移策略的 GPU 高效信道化处理

在信道化处理中，设每包输入数据的最大长度为L_{MAX}。在本文方法中，每包输入数据的长度可以变化但需小于等于L_{MAX}。

本文方法的核心在于需在 GPU 显存上开辟大小为$2L_{\mathrm{MAX}}$的缓存空间，分为大小L_{MAX}的前后两个缓存区。该缓存空间缓存两包连续的输入数据，前一包数据以中线为终点放入前半缓存区，后一包数据以中线为起始放入后半缓存区。

1）初始化步骤：首先在显存上开辟大小为$2L_{\mathrm{MAX}}$缓存区的空间，将后半缓存区中大小为L_{MAX}的数据初始化为全 0，如图 2 所示。

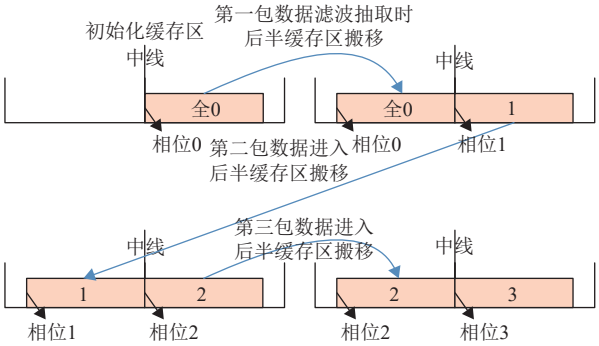


图 2 输入数据进入缓存区示意图

2）数据输入步骤：初始化完成后，输入数据开始分包进入信道化处理模块，每包数据在缓存区的输入流程如图 2 所示。首先将位于后半缓存区的数据搬移到前半缓存区，随后将当前输入数据包放入后半缓存区内。

可以知道，在数据搬移过程中涉及一次显存到显存的数据拷贝（后半缓存区拷贝到前半缓存区）和一次内存到显存的数据拷贝（输入数据拷贝到后半缓存区）。显存到显存的拷贝时间明显低于内存到显存的拷贝时间，因此数据输入步骤的时间主要由一次内存到显存数据拷贝时间决定，这个属于必要的拷贝时间没有带来额外的时间开销。

3）数据变频步骤：为了对输入数据做变频操作且保证相位连续，需要记录缓存区中两个位置的相位，也就是前后两块缓存空间的两包数据起始位置相位，如图 2 所示。为方便描述，由于初始化缓存区时设置的全 0 数据可以被视为第一包输入数据，因此其起始相位被定义为相位 0。随着每包输入数据的输入，每包数据的起始位置相位被按顺序定义为相位 1、相位 2 等。

首先，在初始化缓存区时，将包含初始相位值的复数设置为p_0。当第一包数据输入后，先将全 0 数据包移动到前半缓存区，从而可以计算得到包含第一包数据的起始相位的复数为：

$$p_1 = p_0 \exp(2\pi f_s N_0) \tag{7}$$

式中，f_s是变频的数字频率；N_0是全 0 数据的长度。因此，可以将p_0设置为包含任意需要相位的复数，也可根据N_0计算p_0，从而使得p_1变为包含需要的初始相位的复数。

当第i包数据输入后，先将第$i-1$包数据移动到前半缓存区，再计算得到包含第i包数据的起始相位的复数为：

$$p_i = p_{i-1} \exp(2\pi f_s N_{i-1}) \tag{8}$$

重复上述操作，每包数据输入时都执行此相位计算方法。在信道化中，每次以当前的前半缓存区第一个数据对应的相位为基准对缓存区内所有数据进行变频处理从而保证了数据变频操作的连续性。

4）滤波抽取步骤：滤波器在输入数据上进行滑动卷积，为了保证结果的正确性最重要的是确定滤波器在当前输入数据上滑动卷积的起点位置，如图 3 所示。其原因在于抽取时可能会出现跨越两个数据包的情况。第一次滤波起点偏移量一般可以指定为 0，下一次滤波时滤波器的起点偏移量是以中线为基准计算的。下面描述计算起点偏移量的方法。

在第一包数据滤波抽取时，原后半缓存区内数据搬移到前半缓存区，第一包数据放入后半缓存区，因此缓存区内的所有数据组成了第一次滤波时的整体数据；以滤波起点偏移量（此时一般设置为$\Delta_0 = 0$）为基准，滤波器开始进行滑动卷积完成滤波抽取操作得到第一次的结果数据。即：

$$\boldsymbol{x}_1^1[n] = \sum_{k=0}^{K-1} \boldsymbol{h}[k] \cdot \boldsymbol{x}_{01}[nD + k + \Delta_0] \tag{9}$$

式中，D为抽取倍数，$\boldsymbol{x}_{01} = [\boldsymbol{x}_0, \boldsymbol{x}_1] \in \mathbb{R}^{1\times(N_0+N_1)}$。

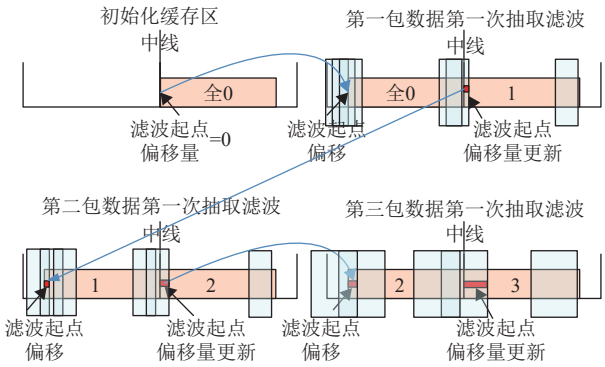


图 3　分包滤波抽取过程示意图

为了计算下一次滤波时的偏移量，需要计算滤波器第一次滑过中线时滤波器最后一个系数相对于中线的距离，以此距离更新滤波起点偏移量Δ_2。

在第二包数据滤波抽取时，首先完成缓存区搬移，再以更新后的滤波起点偏移量为基准开始滑动卷积完成滤波抽取得到第二次的结果数据：

$$\boldsymbol{x}_2^1[n] = \sum_{k=0}^{K-1} \boldsymbol{h}[k] \cdot \boldsymbol{x}_{12}[nD + k + \Delta_1] \tag{10}$$

式中，$\boldsymbol{x}_{12} = [\boldsymbol{x}_1, \boldsymbol{x}_2] \in \mathbb{R}^{1\times(N_1+N_2)}$。

随后，再更新滤波起点偏移量。重复上述过程，即可在每次滤波时以正确的滤波器起点偏移量设置滤波器的起始位置。偏移量的计算过程如下。

设当前滤波起点偏移量为Δ_i，并且当前滤波器的滑动步长为D，令当前缓存区内所有数据的长度为$L_{\text{now}} = N_{i-1} + N_i$，则缓存区内的有效数据长度被定义为所有数据的长度减去当前的滤波起点偏移量，记为：

$$L_{\text{valid}} = L_{\text{now}} - \Delta_i \tag{11}$$

于是当前的滤波起点偏移量更新为：

$$\Delta_{i+1} = \begin{cases} 0 & \text{if } L_{\text{valid}} - \left\lfloor \dfrac{L_{\text{valid}}}{M} \right\rfloor M = 0 \\ M - \left(L_{\text{valid}} - \left\lfloor \dfrac{L_{\text{valid}}}{M} \right\rfloor M \right) & \text{if } L_{\text{valid}} - \left\lfloor \dfrac{L_{\text{valid}}}{M} \right\rfloor M \neq 0 \end{cases} \tag{12}$$

正确的变频操作和滤波器起始位置保证了滤波抽取后的数据相位连续性。但是，所提的滤波抽取操作会造成每次输出结果中存在不需要的结果数

据。因此每次信道化操作后最后还需要计算实际需要的结果数据长度再输出。

5）结果输出步骤：滤波抽取后的结果数据中存在冗余，如图 4 所示。在滤波时，前半缓存区内数据长度和后半缓存区内数据长度提前已知，且已知滤波器的起点偏移量。因此，很容易计算出当前信道化操作后真实需要的滤波结果数据，于是将前部分结果数据丢弃，留下后部分结果数据。

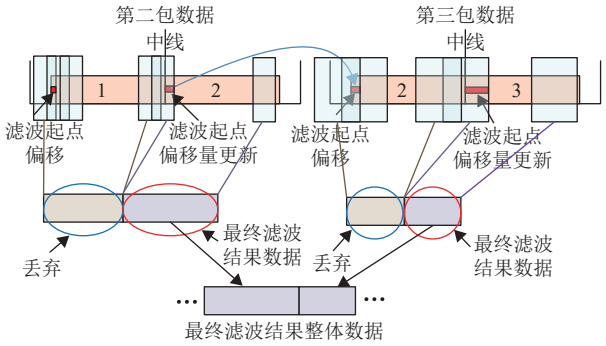


图 4 分包滤波抽取后输出结果示意图

需要注意的是，这里的丢弃不需要实质的丢弃，因为一般来说信道化需要完成多级滤波抽取，一级滤波抽取后的结果在显存上按上述过程流水线式地进行下一级滤波。因此，丢弃操作仅需要调用一次显存到显存的拷贝并结合指针偏移即可一次性完成。随着抽取的进行，数据量越来越少，所需的缓存也越来越小。最后一级滤波抽取完成后，才从显存拷贝到内存。

2.3 关于本文方法的性能讨论

本文方法存在的唯一缺点是，每次针对缓存区的滤波抽取中存在冗余操作。具体来说，进行滤波抽取操作的数据量是上一包数据长度加上当前包数据长度，从而导致了计算量的稍许增加。

然而，从整体信道化过程来看，该缺点不会对整体运行效率造成明显影响。这是由于在基于 GPU 的信道化处理模块中，处理流程主要由 3 部分组成：

1）将位于内存的输入数据拷贝进显存；

2）滤波器组进行多级滤波；

3）将位于显存的结果数据拷贝回内存。

在 GPU 程序中，显存到内存的拷贝和内存到显存的拷贝花费时间往往开销较大，而本文方法在内存拷贝上几乎不存在额外的开销。

此外，利用本文所提的基于缓存区搬移策略的信道化方法，GPU 信道化程序可以专注于处理缓存区内的数据，避免了复杂的为保证相位连续的逻辑控制。由于在相位连续的逻辑控制上简单且在计算时间上有着优异的表现，因此有着较好的实用价值。

2.4 关于多级滤波方式的讨论

实际应用中往往需要指定不同的抽取倍数，这可以通过设计多级不同阶数滤波器来实现。对于每一级的滤波操作，通常存在两种滤波方式：直接滤波和多相滤波。

在 FPGA 或者 CPU 架构下，通常使用多相滤波结合流水线操作的方式减少运算量，提升滤波速率。然而，在 GPU 架构下，多相滤波的方式并不合适，可从基于 GPU 的多相滤波处理流程中看出。

1）输入数据重排步骤：在考虑输入数据给定的情况下（即输入数据内部的顺序给定），为了完成多相滤波需要首先将输入的一维数据视为矩阵，并进行矩阵转置，以达到分组的目的。

2）多相滤波步骤：对矩阵化后的数据按行进行并行化滤波处理。

3）输出数据重排步骤：对最终得到的结果矩阵再进行矩阵转置，恢复原本的输出结果顺序。

从上述流程中可以看到，基于 GPU 的多相滤波处理过程中涉及两次矩阵转置。由于在实际应用中，一包输入数据往往较大，因此，基于 GPU 的多相滤波处理时间主要由两次矩阵转置的时间决定。换句话说，在 GPU 架构下，多相滤波的处理方式不仅没有带来性能提升反而带来了数据重排的时间开销，导致运算效率低下。

在本文所提的基于 GPU 的高效信道化方法中采用的是直接滤波方式。相比于多相滤波的方式，直接滤波不需要两次额外的数据重排操作，仅需直接对输入数据进行并行化滤波处理。并且，得益于 GPU 并行架构，在每个线程中，直接滤波也可以视为并行分组滤波。此外，若采用多相滤波的方式，还额外需要输入数据的长度满足能够矩阵化的要求。而直接滤波方式适用于任意长度的输入数据。

因此，在 GPU 架构下，每一级均采用直接滤波方式优于多相滤波的方式。两种滤波方式的效率对比将在后续实验部分给出。

3 实验

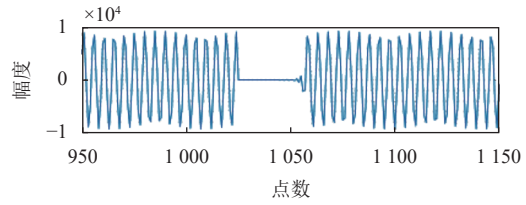
3.1 相位连续性实验

在实验中，使用 FPGA 采集信号源信号以每

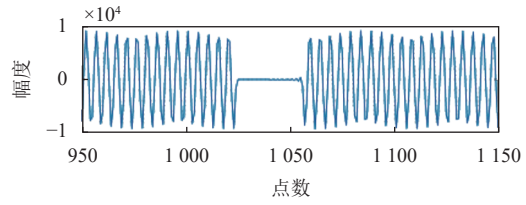
包 8 MB（1 M 个 float2）进入信道化模块，将每次处理的信道化结果写入文件。每处理 80 MB 字节（10 包数据）得到的信道化结果为一个文件。抽取倍数设置为 1 024。

实验中，使用的 GPU 型号为 Nvidia GeForce RTX 3080，其 float 计算能力为 29.77 TFLOPS，带宽为 760.3 GB/S，CPU 处理器为 Intel（R） Core（TM） i7-10700。算法中，每个线程块被设置为一维的 512 个线程，因此共有 2 048 个网格块。

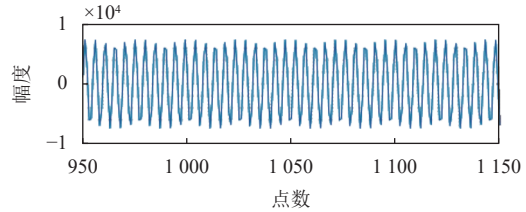
图 5 是两包连续数据处理结果的拼接处波形图。图 5a 和图 5b 为两包数据独立滤波后的结果，可以看到在中点位置相位不连续且滤波数据缺失，图 5c 和图 5d 为本文处理结果，从实验结果中可以看到，本文方法保证了相位的连续性。



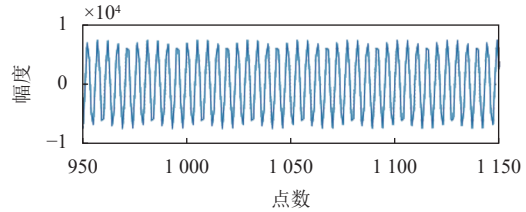
a. 独立滤波后两包数据通道I连接处



b. 独立滤波后两包数据通道Q连接处



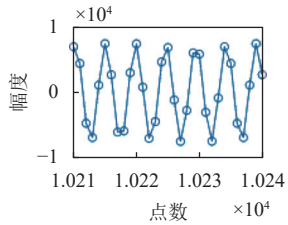
c. 本文处理后两包数据通道I连接处



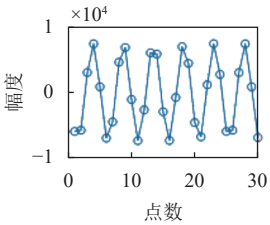
d. 本文处理后两包数据通道Q连接处

图 5　两包数据滤波抽取后结果连接处波形图

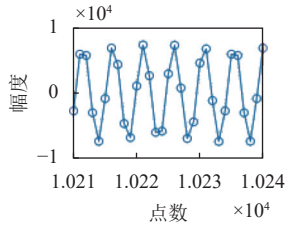
图 6a～图 6d 是两个连续的数据文件（文件 1 和文件 2）处理后的末尾和起始部分波形图，其中左列为文件 1、右列为文件 2，对比观察文件 1 和文件 2 的通道 I 和通道 Q，可以看到文件 1 的末尾相位与文件 2 的起始相位连续。上述结果说明了所提方法的正确性，能够保证滤波抽取后结果的相位连续性。图 6e～图 6h 是独立滤波得到的文件 1 和文件 2，可以看到在文件 2 的起始位置相位不连续且滤波数据缺失。



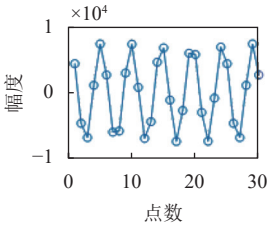
a. 本文处理后的通道I末尾



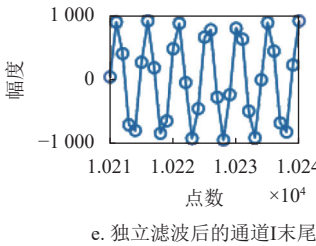
b. 本文处理后的通道I起点



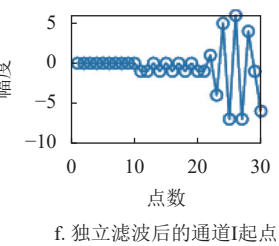
c. 本文处理后的通道Q末尾



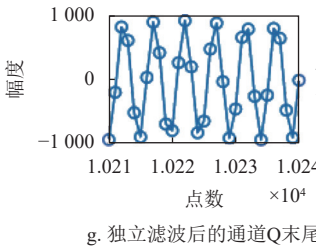
d. 本文处理后的通道Q起点



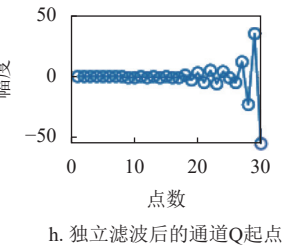
e. 独立滤波后的通道I末尾



f. 独立滤波后的通道I起点



g. 独立滤波后的通道Q末尾



h. 独立滤波后的通道Q起点

图 6　两个连续滤波抽取结果文件的末尾与起始波形图

3.2　基于 GPU 的直接滤波和多相滤波速度对比

通过改变一包输入数据的大小，得到基于 GPU 的不同滤波方式下的运算时间对比图，如图 7 所示。

可以看到基于 GPU 的直接滤波方式运算效率明显高于多相滤波方式，且随着输入数据长度增加，基于 GPU 的多相滤波方式速度与直接滤波方式的差距被扩大。这是因为多相滤波的时间主要由两次矩阵转置决定，矩阵转置的时间受到输入数据长度的影响更大。

这说明了在 GPU 架构下，直接滤波方式相比于多相滤波方式更适合用于实现信道化处理。

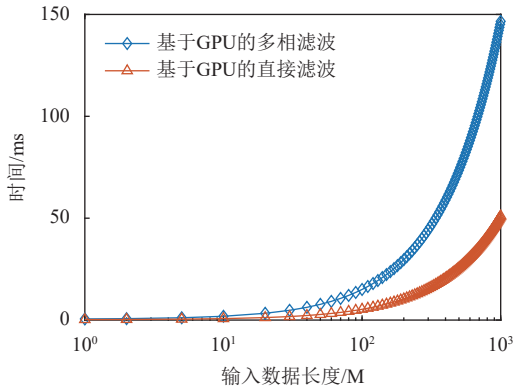


图 7 基于 GPU 的不同滤波方式运算时间对比图

3.3 基于 GPU 的直接滤波和基于 CPU 的直接滤波时间对比

实验通过改变一包输入数据的大小，得到 CPU 和 GPU 架构下的平均运算时间，如表 1 所示。

表 1 CPU 和 GPU 架构下直接滤波平均时间对比

数据长度/MB	CPU下直接滤波平均时间/ms	GPU下直接滤波平均时间/ms	加速比
8	775.7	0.227 3	3 324
16	1 537.8	0.279 6	5 500
32	3 112.4	0.380 9	8 171
48	4 671.9	0.380 9	12 265
64	6 231.8	0.589 8	10 565
80	7 851.1	0.702 5	11 175

可以看到直接滤波算法在 GPU 架构下远远快于在 CPU 架构下的运算速度。观察 GPU 架构相比于 CPU 架构的加速比，可以看到在典型应用下，如输入数据长度为 32 MB 以上，加速比达到 8 000 倍以上，输入数据长度为 48 MB 以上，加速比达到一万倍以上。

从中可以看到，GPU 相对于 CPU 的加速比随着输入数据长度的增长而增长，这说明越长的输入数据 GPU 处理带来的速度增益越高，当然由于访存速度的限制这种增长并非线性增长。

4 结束语

本文研究了如何基于 GPU 实现高效信道化处理。针对实际中分包滤波抽取导致的信道化结果相位不连续性问题，给出了一种基于缓存区搬移策略的 GPU 高效信道化方法。相比于传统分段卷积方法，本文方法在多级滤波的架构下计算效率更高。本文还分析了在 GPU 架构下基于多相滤波和基于直接滤波的多级滤波方法，由于多相滤波带来了额外的数据重排操作因此并不适合用于 GPU 架构下的信道化处理。通过一系列实验证明了本文方法能保证信道化结果的相位连续性，验证了基于 GPU 的直接滤波优于基于 GPU 的多相滤波，并给出了基于 GPU 和基于 CPU 的直接滤波时间对比，直观地表明了本文所提的基于 GPU 的高效信道化方法处理在运算效率上的显著提升。

参 考 文 献

[1] HAN W W, LIU Z H, XU Y F, et al. Photonic RF channelization applications of microcombs[J]. IEEE Journal of Selected Topics in Quantum Electronics, 2024, 30(5): 7600417.

[2] 杨静, 吕幼新. 高效数字信道化 IFM 接收机的研究[J]. 电子科技大学学报, 2005, 34(4): 444-447.
YANG J, LYU Y X, Efficient digital channelized IFM receiver research[J]. Journal of University of Electronic Science and Technology of China, 2005, 34(4): 444-447.

[3] KIM S, PLISHKER S, BHATTACHARYYA S. An efficient GPU implementation of an arbitrary resampling polyphase channelizer[C]//2013 Conference on Design and Architectures for Signal and Image Processing. Cagliar: [s.n.], 2013: 231-238.

[4] ADHINARAYANAN V, FENG W C. Wideband channelization for software-defined radio via mobile graphics processors[C]//Proceedings of the International Conference on Parallel and Distributed Systems. New York: IEEE, 2013: 86-93.

[5] ZHANG W X, YAO Y S, ZHAO Z K, et al. Design and FPGA implementation of a novel efficient FRM-based channelized receiver structure[J]. IEEE Access, 2019, 7: 114778-114787.

[6] MAHESH R, VINOD A P. Reconfigurable frequency response masking filters for software radio channelization [J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2008, 55(3): 274-278.

[7] AMBEDE A, SHREEJITH S, VINOD A P, et al. Design and realization of variable digital filters for software-defined radio channelizers using an improved coefficient decimation method[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2016, 63(1): 59-63.

[8] 匡宏印. 一种基于时间–频率特征分析的自适应信道化技术[J]. 电讯技术, 2022, 62(3): 367-372.
KUANG H Y. An adaptive channelized technology based on time-frequency analysis[J]. Telecommunication Engineering, 2022, 62(3): 367-372.

[9] XU S C, LIU G M, GAO M G. Design and implementation of digital channelized receiver in multi-FPGA[C]// Proceedings of the 1st International Conference on Information Science and Engineering. New York: IEEE, 2009: 178-181.

[10] DEL MUNDO C, ADHINARAYANAN V, FENG W C. Accelerating fast Fourier transform for wideband channelization[C]//Proceedings of the IEEE International Conference on Communications. New York: IEEE, 2013: 4776-4780.

[11] MAHESH R, VINOD A P. Low complexity flexible filter banks for uniform and non-uniform channelisation in software radios using coefficient decimation[J]. IET Circuits, Devices & Systems, 2011, 5(3): 232-242.

[12] 徐庆征, 彭丽. 基于 MATLAB 的分段卷积计算[J]. 苏州科技学院学报: 工程技术版, 2006, 19(2): 88-90.
XU Q Z, PENG L. Calculation of partition convolution based on the MATLAB[J]. Journal of University of Science and Technology of Suzhou: Engineering and Technology, 2006, 19(2): 88-90.

责任编辑 张 莉