(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 110675480 B
(45)授权公告日 2023.07.04

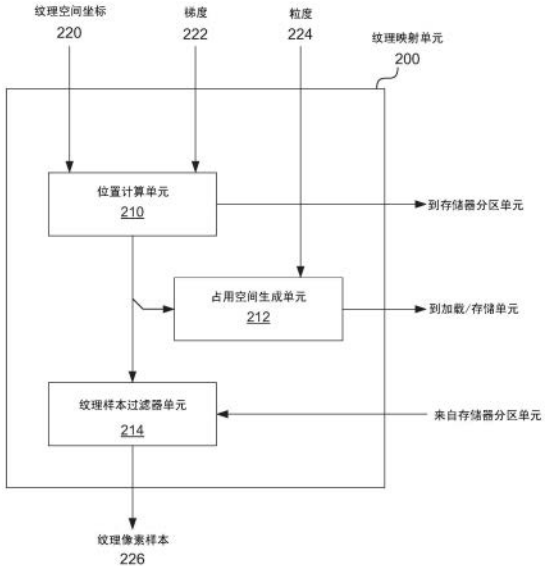
(21)申请号 201910456799.8
(22)申请日 2019.05.29
(65)同一申请的已公布的文献号
申请公布号 CN 110675480 A
(43)申请公布日 2020.01.10
(30)优先权数据
16/027,136 2018.07.03 US
(73)专利权人 辉达公司
地址 美国加利福尼亚州
(72)发明人 尤里·乌拉尔斯基 H·P·莫尔顿
埃里克·布雷恩·卢姆
乔纳森·J·迪奈斯卡
史蒂文·詹姆斯·海因里希
斯特凡诺·佩斯卡多尔
瑟利斯·加德雷
米夏埃尔·艾伦·费特曼
(74)专利代理机构 北京市磐华律师事务所 11336
专利代理师 高伟 娄晓丹
(51)Int.Cl.
*G06T 15/04*(2011.01)
(56)对比文件
CN 104050705 A,2014.09.17
CN 105684037 A,2016.06.15
CN 1806259 A,2006.07.19
US 2016035129 A1,2016.02.04
US 2017263046 A1,2017.09.14
US 2018158227 A1,2018.06.07
丁剑飞等.基于GPU的自由立体显示器通用渲染算法.《系统仿真学报》.2012,(第07期),
于平.基于GPU加速的辐射度光照算法的研究及应用.《国外电子测量技术》.2016,(第11期),
审查员 张媛媛
权利要求书2页 说明书18页 附图14页

(54)发明名称

# 用于获取纹理操作的采样位置的方法和装置

(57)摘要

本发明公开了用于获取纹理操作的采样位置的方法和装置,具体地公开了用于报告纹理占用空间信息的方法和装置。纹理占用空间标识将用于渲染场景中的像素的纹理的部分。所公开的方法和装置通过首先标识给定纹理映射中的哪些纹理像素在随后渲染场景时需要,从而有利地提高了解耦的着色系统的系统效率。因此,所生成和存储的纹理像素的数量可以减少,以包括被标识的纹理像素。没有被标识的纹理像素不需要被渲染和/或存储。




CN 110675480 B

1.一种用于报告纹理占用空间的方法,包括:

从着色器程序接收包括纹理映射坐标的纹理占用空间查询请求,其中所述纹理占用空间查询请求与光栅化的几何形状所覆盖的屏幕空间中的像素相关联;

标识包括用于纹理映射过滤操作的纹理占用空间的纹理像素,其对应于所述纹理占用空间查询请求;以及

向所述着色器程序发送包括所述纹理占用空间的查询结果信息,

其中所述发送包括将所述查询结果信息存储在所述着色器程序可访问的存储器电路中。

2.如权利要求1所述的方法,其中所述纹理映射坐标包括两个维度。

3.如权利要求1所述的方法,其中所述纹理映射坐标包括三个维度。

4.如权利要求1所述的方法,其中所述纹理映射坐标被配置为标识立方体映射中的位置。

5.如权利要求1所述的方法,其中所述纹理占用空间查询请求进一步包括在每个所述纹理映射坐标处的梯度。

6.如权利要求1所述的方法,其中所述纹理占用空间查询请求进一步包括与所述纹理占用空间相对应的详细参数的纹理映射级别。

7.如权利要求1所述的方法,其中所述纹理占用空间查询请求进一步包含用于所述查询结果信息的分辨率规范。

8.如权利要求7所述的方法,其中所述查询结果信息包括位图和粗化因子,所述位图被生成以根据所述分辨率规范指示所述纹理占用空间,所述粗化因子被生成以在所述分辨率规范的维度中指示其值为所述位图的每一位的一个或更多个纹理像素的比例尺。

9.如权利要求8所述的方法,其中所述比例尺是基于所述分辨率规范而被计算的,以包含所述维度中的所述纹理占用空间。

10.如权利要求1所述的方法,其中所述查询结果信息包括固定长度的位图,并且所述固定长度的位图的每个位代表所述纹理占用空间所覆盖的一个或更多个纹理像素的区域。

11.如权利要求10所述的方法,其中所述固定长度的位图包括六十四位。

12.如权利要求11所述的方法,其中所述六十四位指示被组织为8×8纹理像素、16×4纹理像素、32×2纹理像素或4×4×4纹理像素之一的一组纹理像素。

13.如权利要求1所述的方法,其中所述查询结果信息包括被计算用于指示纹理映射坐标空间中的所述纹理占用空间的位置的锚点。

14.如权利要求1所述的方法,其中标识包括将纹理采样区域投影到所述纹理映射坐标处的纹理像素空间中,并标记被所投影的纹理采样区域覆盖的每个纹理像素。

15.一种处理单元,被配置为:

从着色器程序接收包括纹理映射坐标的纹理占用空间查询请求,其中所述纹理占用空间查询请求与光栅化的几何形状覆盖的屏幕空间中的像素相关联;

标识包括用于纹理映射过滤操作的纹理占用空间的纹理像素,其对应于所述纹理占用空间查询请求;以及

向所述着色器程序发送包括所述纹理占用空间的查询结果信息,

其中,为了发送,所述处理单元将所述查询结果信息存储在所述着色器程序可访问的

存储器电路中。

16.如权利要求15所述的处理单元,其中所述纹理映射坐标包括以下之一:二维、三维、立方体映射中的位置。

17.如权利要求15所述的处理单元,其中所述纹理占用空间查询请求进一步包括以下中的至少一种:在每个所述纹理映射坐标处的梯度;与所述纹理占用空间相对应的详细参数的纹理映射级别;或用于所述查询结果信息的分辨率规范。

18.如权利要求15所述的处理单元,其中所述查询结果信息包含以下中的至少一个:固定长度的位图或锚点,所述固定长度的位图的每个位都配置为代表所述纹理占用空间所覆盖的一个或更多个纹理像素的区域;所述锚点被计算以指示纹理映射坐标空间中的所述纹理占用空间的位置。

19.如权利要求15所述的处理单元,其中标识包括将纹理采样区域投影到所述纹理映射坐标处的纹理像素空间中,并标记被所投影的纹理采样区域覆盖的每个纹理像素。

20.一种非暂时性计算机可读存储介质,其存储指令,当所述指令被处理器执行时,使所述处理器:

从着色器程序接收包括纹理映射坐标的纹理占用空间查询请求,其中所述纹理占用空间查询请求与光栅化的几何形状所覆盖的屏幕空间中的像素相关联;

标识包含用于纹理映射过滤操作的纹理占用空间的纹理像素,其对应于所述纹理占用空间查询请求;以及

向所述着色器程序发送包括所述纹理占用空间的查询结果信息,

其中,为了发送,所述处理器将所述查询结果信息存储在所述着色器程序可访问的存储器电路中。

# 用于获取纹理操作的采样位置的方法和装置

## 技术领域

[0001] 本发明涉及计算机生成的图形,并且更具体地涉及获取纹理操作的采样位置。

## 背景技术

[0002] 某些用于三维图形的解耦的着色算法在初始渲染传递(pass)中将场景对象的颜色值渲染为场景对象的一个或更多个相关的纹理映射。然后,在随后的渲染传递中使用纹理映射采样生成场景的最终像素值,这可以由专用的高性能纹理采样电路提供。通常整个纹理映射都被渲染,以保证生成最终像素值可能所需的所有纹理像素的可用性。虽然解耦的着色可以提供一定的优势,但是总体系统效率可能会降低,因为渲染到关联纹理映射中的大部分纹理像素实际上并没有使用。因此,有必要解决这些问题和/或与现有技术相关的其他问题。

## 发明内容

[0003] 公开了一种用于报告纹理占用空间(texture footprint)的方法、计算机可读介质和系统。所述方法包括:从着色器程序接收包括纹理映射坐标的纹理占用空间查询请求,标识包括用于纹理映射过滤操作的纹理占用空间的纹理像素,其对应于纹理占用空间查询请求,以及将包括纹理占用空间的查询结果信息发送到着色器程序。在一个实施例中,纹理占用空间查询请求与光栅化的几何形状所覆盖的屏幕空间中的像素相关联。在一个实施例中,发送包括将查询结果信息存储在着色器程序可访问的存储器电路中。

[0004] 计算机可读介质包括指令,所述指令当由处理单元执行时,使处理单元执行该方法。此外,系统还包括配置为执行该方法的处理单元。

## 附图说明

[0005] 图1A示出了根据一个实施例的用于报告纹理占用空间的方法的流程图。

[0006] 图1B示出了根据一个实施例的纹理映射。

[0007] 图1C示出了根据一个实施例的示例纹理占用空间。

[0008] 图1D示出了根据一个实施例的纹理占用空间的粗化。

[0009] 图1E示出了根据一个实施例的不同纹理占用空间的不同纹理像素组尺寸。

[0010] 图1F示出了根据一个实施例的相同纹理占用空间的不同粗化粒度。

[0011] 图2示出了根据一个实施例的配置为执行纹理占用空间查询操作的纹理映射单元。

[0012] 图3示出了根据一个实施例的并行处理单元。

[0013] 图4A示出了根据一个实施例的、图3的并行处理单元内的通用处理集群。

[0014] 图4B示出了根据一个实施例的、图3的并行处理单元的存储器分区单元。

[0015] 图5A示出了根据一个实施例的、图4A的流式多处理器。

[0016] 图5B是根据一个实施例的使用图3中的PPU实现的处理系统的概念图。

[0017] 图5C示出了一个示例系统，其中可以实现前面各种实施例的各种架构和/或功能。

[0018] 图6是根据一个实施例的、由图3的PPU实现的图形处理管线的概念图。

[0019] 详细描述

[0020] 本公开的实施例响应于发布到纹理映射单元的纹理占用空间查询，向着色器程序提供纹理占用空间信息。纹理占用空间查询指定采样信息，其可以包括纹理坐标、纹理坐标处的梯度、采样操作的类型、纹理映射标识符、报告纹理占用空间所需的分辨率以及任何其他技术上相关的采样细节。已报告的纹理占用空间信息标识将对哪些纹理像素(纹理映射样本)进行采样，以形成用于纹理映射单元执行的等效纹理采样操作的纹理占用空间。在一个实施例中，着色器程序生成纹理查询，以标识渲染给定场景需要哪些纹理像素。然后，在渲染场景几何形状之前先渲染所标识的纹理像素。

[0021] 给定的纹理占用空间可以位于纹理映射中的任何位置。在一个实施例中，纹理占用空间信息中的位图被配置为指定纹理占用空间中的哪些纹理像素被覆盖。可以在纹理占用空间信息中提供锚点(anchor point)，以便在纹理映射坐标空间中定位位图的边界区域。此外，所报告的纹理占用空间信息还可以指定一个或更多个粗化(coarsening)因子。可以计算给定维度的粗化因子，以指示该维度中的多少纹理像素对应于位图中的给定位。

[0022] 在各种实施例中，包括占用空间生成单元的电路被配置为在给定一组采样信息的情况下标识纹理占用空间中的纹理像素。占用空间生成单元的一个或更多个实例可用于生成对纹理占用空间查询的响应，以及执行等效的纹理采样操作。

[0023] 图1A示出了根据一个实施例的用于报告纹理占用空间的方法110的流程图。虽然方法110是在处理单元的上下文中描述的，但是方法110也可以由处理单元执行的程序、任何定制电路或定制电路与程序的组合执行。例如，方法110可以由GPU(图形处理单元)、CPU(中央处理单元)或任何其他技术上可行的处理器执行。此外，本领域技术人员将理解，任何执行方法110的系统都在各种实施例的范围和精神内。在一个实施例中，方法110由GPU中的纹理映射单元执行。

[0024] 在步骤112，纹理映射单元从着色器程序接收纹理占用空间查询请求。在一个实施例中，纹理占用空间查询请求包括纹理映射坐标。在一个实施例中，纹理占用空间查询请求与屏幕空间中由光栅化的几何形状所覆盖的像素相关联。在一个实施例中，纹理映射坐标在二维中指定。在另一个实施例中，纹理映射坐标在三维中指定。在另一个实施例中，纹理映射坐标配置为标识立方体映射(cube-map)中的位置。在某些实施例中，纹理占用空间查询请求进一步包括在每个纹理映射坐标处的梯度。在一个实施例中，纹理占用空间查询请求进一步包括与纹理占用空间相对应的详细参数的纹理映射级别。详细参数的级别可以指定特定的基本映射级别、相对于映射级别的细节偏置或箝位(clamp)的级别、与选择一个或更多个映射级别相关的任何其他参数或其任意组合。在一个实施例中，纹理占用空间查询请求进一步包括用于查询结果信息的分辨率规范(resolution specification)。分辨率规范指示纹理占用空间查询在位图的每个维度中请求多少个离散覆盖样本。每个覆盖样本被分配位图中的一个位。例如，4x4覆盖样本的分辨率规范将导致纹理映射单元生成4x4位图，而不管在相应的纹理占用空间中实际覆盖了多少个纹理像素。在这样的一个例子中，如果纹理占用空间是6个纹理像素宽，则可以为水平维度报告两个粗化因子，并为竖直维度报告一个粗化因子，从而给4x4的位图8个纹理像素的有效宽度和4个纹理像素的有效高度。使用

这些粗化因子,位图中的每个位代表两个水平相邻纹理像素的覆盖。

[0025] 在步骤114,纹理映射单元标识包括用于纹理映射过滤操作的纹理占用空间的纹理像素,所述纹理占用空间对应于纹理占用空间查询请求。纹理映射单元可以使用本领域中任何技术上可行的技术来标识纹理像素,具体地纹理映射单元使用为纹理像素占用空间查询和纹理采样操作两者标识同一组纹理像素的技术。在一个实施例中,纹理映射单元使用相同的技术来执行纹理像素占用空间查询和纹理采样操作两者。在某些实施例中,纹理映射单元可以实现逻辑电路的相同实例,用于执行纹理像素占用空间查询和纹理采样操作。

[0026] 在步骤116,纹理映射单元将包括纹理占用空间的查询结果信息发送到着色器程序。在一个实施例中,纹理映射通过将查询结果信息存储在着色器程序可访问的存储器电路中来发送查询结果信息。然后着色器程序从存储器电路中检索查询结果信息。

[0027] 现在,根据用户的需要,将阐述关于实现上述框架所采用的各种可选架构和特征的更多说明性信息。应该强烈注意,下列信息是为了说明目的而提出的,而不应被解释为以任何方式加以限制。下列任何特征都可以与所描述的其他特征可选地合并,或不排除这些特征。

[0028] 图1B说明了根据一个实施例的纹理映射。如图所示,照相机120被放置在三维(3D)空间中,以通过视图平面122来观看几何对象126。几何对象126被描绘为被矩形网格包围的金字塔,每个矩形对应于纹理映射132的纹理像素。纹理映射132是一个二维(2D)结构,配置为存储纹理图像,它可以是任何类型的图像,例如照片或计算机生成的图像。纹理映射位置134(A)、134(B)和134(C)分别映射到顶点130(A)、130(B)和130(C)。当几何对象126被渲染时,纹理图像的纹理像素样本将被从纹理映射132中提取并过滤,以使几何对象显现为纹理图像的一部分的外观。

[0029] 视图平面122上的像素124在几何对象126上具有相关联的投影区域128。投影区域128被描述为包括被映射到几何对象126以形成纹理映射132的矩形纹理像素样本。投影区域的形状取决于相机120相对于几何对象126上的相交曲面的位置。投影区域的形状也取决于用于生成像素124的采样类型。例如,点采样(未显示)使用一个点,通常取自投影到几何对象126的面的像素的中心区域。点采样通常产生相对较低质量的渲染图像和图像序列。较高质量的图像使用形成更大的投影区域128的多个纹理像素进行渲染。例如,各向异性渲染使用投影区域128内的多个纹理像素,其可以具有任意的形状和尺寸。如本例所示,投影区域128包括从纹理映射132中提取的14个样本。

[0030] 纹理占用空间143是纹理映射132中(即纹理空间中)中与投影区域128相对应的一个区域。在实际中,包含几何对象126的几何图元被投影和光栅化到视图平面122(即,在屏幕空间内),以标识被相机122的给定位置、视图平面122的给定位置和几何图元的位置的几何图元所覆盖的所有像素。这样,像素124就可以被标识为覆盖几何对象126。此外,对像素124的引用可以保留相应几何图元的各种属性,例如定义照相机和投影区域128之间的角关系的纹理坐标和梯度。此外,可以基于纹理坐标和相关梯度来标识包括纹理占用空间143的纹理像素。

[0031] 图1C说明了根据一个实施例的示例性纹理占用空间143。如图所示,边界区域142被定位为完全包围纹理占用空间143,其根据二维纹理坐标(Xs,Ys)位于纹理映射140内。在

纹理坐标处从纹理映射140中提取的给定样本是从包括纹理占用空间143的纹理像素计算得到的。计算锚点144以定位包围纹理占用空间143的边界区域142。锚点144可以沿着某纹理像素边界(例如,偶数纹理像素)、单词边界、高速缓存边界或任何其他技术相关的边界对齐。因此,虽然锚点144显示在边界区域142的左上角,但是根据不同的实现需求,锚点144也可以位于偏移左上角的位置。

[0032] 边界区域142在每个维度中包括一个尺寸。如图所示,对于2D纹理,边界区域142具有水平尺寸(XSize 146)和垂直尺寸(YSize 148)。纹理像素的网格包括边界区域142,其中被纹理占用空间143覆盖的纹理像素显示为黑色方块,未被纹理占用空间143覆盖的纹理像素显示为白色方块。在这个例子中,纹理占用空间143自然地适合于8×8边界区域142内。在一个实施例中,3D边界区域(未示出)可以是具有XSize 146、YSize 148和深度尺寸的体积。在另一个实施例中,可以将边界区域配置为覆盖立方体映射(未示出)的一个或更多个面。

[0033] 为了方便起见,纹理空间坐标在这里示出为X和Y,而非归一化的纹理空间坐标通常使用U和V,归一化的纹理空间坐标通常使用S和T。

[0034] 在一个实施例中,查询结果信息包括位图和粗化因子。位图被生成以根据分辨率规范指示纹理占用空间。粗化因子被生成以指示比例尺(scale),该比例尺的值为在分辨率规范的维度中位图的每个位具有一个或更多个纹理像素。此外,可以基于分辨率规范计算粗化因子,以在维度中包围纹理占用空间。

[0035] 图1D说明了根据一个实施例的纹理占用空间143的粗化。如图所示,纹理占用空间143的边界区域142是十六个纹理像素宽(XSize 146等于十六)和八个纹理像素高(YSize 148等于八)。要完整地表示一个8×16区域的全部细节将需要一百二十八位。在一个实施例中,使用六十四位近似,以及YSize的粗化因子为1,XSize的粗化因子为2来表示8×16区域。六十四位近似允许纹理占用空间143由固定长度的六十四位位图150表示。纹理像素的网格包括边界区域142,其中纹理占用空间143覆盖的纹理像素显示为黑色方块,纹理占用空间143未覆盖的纹理像素显示为白色方块。粗化操作149将边界区域142的纹理像素覆盖映射到固定长度的位图150(例如,六十四位位图),该位图包括用于指示给定纹理像素组是否被纹理占用空间143覆盖的位。如图所示,边界区域142中两个水平相邻的纹理像素构成一个1×2纹理像素组。一个1x2纹理像素组的覆盖由固定长度的位图150中的一个位指示。在一个实施例中,如果给定的纹理像素组中的任何纹理像素被纹理像素占用空间143覆盖,则在固定长度的位图150中设置相应的位。在一个实施例中,具有任意尺寸(如1×2、2×1、2×2、4×2、2×4等)的纹理像素组可以映射到固定长度的位图150中的位。在某些实施例中,纹理像素组的每个维度都被限制为两个整数的幂(例如,1、2、4、8、16等)。

[0036] 在一个实施例中,位图是固定长度的位图,该固定长度位图的每个位代表纹理占用空间所覆盖的一个或更多个纹理像素的区域。在一个实施例中,固定长度位图包括六十四位。在其他实施例中,固定长度位图包括更多的位或更少的位,并且可以包含与给定GPU的本地字尺寸一致的若干位。在操作中,纹理映射单元计算粗化因子,以将纹理占用空间映射到固定长度的位图中。例如,对于六十四位的位图和8×8的分辨率规范,8×8的纹理像素的纹理占用空间在每个维度中可以使用粗化因子为1来表示。在另一个示例中,对于六十四位的位图和8×8的分辨率规范,在每个维度中可以使用粗化因子为2来表示16×16纹理像素的纹理占用空间。在各种实施例中,六十四位的位图被组织为8×8纹理像素、16×4纹理

像素、32×2纹理像素的2D分辨率和一个或更多个3D分辨率(例如,包括4×4×4纹理像素)中的一个。

[0037] 图1E说明了根据一个实施例的不同纹理占用空间的不同纹理像素组尺寸。如图所示,固定长度的位图150可以直接表示纹理占用空间143(0),其适合于指定分辨率为8×8的纹理像素。在这个例子中,一个纹理像素包括一个1x1纹理像素组,并映射到固定长度位图150中的一个位。固定长度位图150的第一个位152指示左上角纹理像素组的覆盖,而固定长度位图150的另一个位154指示右下角纹理像素组的覆盖。对于1x1纹理像素组,固定长度位图150中的每个位代表一个纹理像素。

[0038] 固定长度位图150也可以表示垂直范围大于8个纹理像素的纹理占用空间143(1)。在这个例子中,两个纹理像素包括一个2x1纹理像素组,并且每个2x1纹理像素组映射到固定长度位图150中的一个位。在这个例子中,固定长度位图150的第一个位152指示对应的纹理像素组中的两个纹理像素的覆盖。如果这两个纹理像素中的任何一个被覆盖,则第一个位152被分配以指示2×1纹理像素组的覆盖。同样,固定长度位图150可以表示纹理占用空间142(2)和143(3),每个纹理占用空间具有不同的纹理像素组尺寸。

[0039] 在各种实施例中,粗化因子提供了从覆盖任意大数量纹理像素的高度加长纹理占用空间到被覆盖的纹理像素的固定长度位图表示的映射。当多个纹理像素由位图中的一个位表示时,如果多个纹理像素中的任何一个被纹理占用空间覆盖,则将该位设置为指示至少一个纹理像素被覆盖。通过这种方式,如果包含其中任何一个,那么所有的多个纹理像素都会被认为包含在纹理占用空间中。粗化因子由纹理映射单元计算并连同位图和其他查询结果信息一起发送到着色器程序。

[0040] 虽然这种用固定长度位图表示纹理占用空间的方法偶尔会包含超出要标记为被覆盖的严格最小值集的额外的纹理像素,但是额外的纹理像素在实际应用中表现出微小的低效率。此外,在实际的纹理映射单元实现中,为任意纹理占用空间维护固定长度的位图可以提高总体架构效率。

[0041] 图1F说明了根据一个实施例的相同纹理占用空间的不同粗化粒度。如图所示,边界区域142的高度为8个纹理像素(YSize=8),宽度为16个纹理像素(XSize=16)。根据分辨率规范,粗化操作149(0)将边界区域142映射到8×8位图(例如,固定长度位图150)。分辨率规范可以被指示为关联的纹理占用空间查询的一部分。根据不同的分辨率规范,另一个粗化操作149(1)将边界区域142映射到8x4位图。类似地,根据不同的分辨率规范,另一个粗化操作149(2)将边界区域142映射到4x4位图。如图所示,可以应用不同的粗化因子以将边界区域142映射到给定分辨率规范的位图。

[0042] 在一个实施例中,标识纹理占用空间内的纹理像素包括将纹理采样区域投影到指定的纹理映射坐标处的纹理像素空间中以产生纹理占用空间,并标记被投影的纹理采样区域所覆盖的每个纹理像素。在一个简单的双线性采样例子中,纹理坐标确定了纹理空间中2x2采样内核的中心。在这样的示例中,除了可能选择用于采样操作的详细级别(LOD)映射外,梯度不是必要的。使用各向异性采样可以显著提高图像质量,其使用指定的梯度将纹理采样区域从屏幕空间投射到几何对象上的纹理空间,以确定纹理占用空间。在某些情况下,纹理空间中的纹理占用空间可能被高度加长,如图1E中所示的纹理占用空间143(1)、143(2)和143(3)。

[0043] 图2例示了根据一个实施例的配置为执行纹理占用空间查询操作的纹理映射单元200。如图所示，纹理映射单元200包括位置计算单元210、占用空间生成单元212和纹理样本过滤器单元214。在一个实施例中，纹理映射单元200在图5A的特殊功能单元(SFU)552中实现。可以使用专用逻辑电路实现纹理映射单元200的一个或更多个功能。或者，纹理映射单元200可以至少部分实现为着色器程序，并且可以在图6的片段着色阶段670中操作。

[0044] 纹理映射单元200配置为接收纹理空间坐标220、梯度222和粒度224。纹理空间坐标220可以根据特定的纹理映射分辨率(例如u，v坐标)缩放，或者纹理坐标220可以被归一化(例如s，t坐标)。梯度222可以包括屏幕空间对纹理空间的偏导数。例如，对于二维纹理空间坐标220，梯度222可以包括四个偏导数：$\partial u/\partial x$、$\partial u/\partial y$、$\partial v/\partial x$、和$\partial v/\partial y$，其中x和y为屏幕空间坐标变量，u和v为纹理空间坐标变量。粒度224可以包括用于覆盖位图(例如，固定长度位图150)的指定分辨率。此外，细节级别(LOD)可以由纹理映射单元200指定和接收。

[0045] 在一个实施例中，将纹理映射单元200配置为执行纹理采样操作，并分别执行纹理占用空间查询操作。纹理采样操作和纹理占用空间查询操作可以作为请求或指令呈现给纹理映射单元200。任何技术上可行的指令集架构(ISA)构造都可以在不偏离本公开实施例的范围和精神的情况下针对该请求或指令实现。在一个实施例中，向纹理映射单元200发布纹理采样指令(包含例如纹理坐标、LOD等相关信息)表示纹理采样请求，而向纹理映射单元200发布纹理查询指令(包含相关信息)表示纹理占用空间查询请求。给定的着色器程序可以在执行时发布多个纹理采样指令和/或纹理查询指令。

[0046] 在一个实施例中，纹理映射单元200通过使位置计算单元210确定关联的纹理采样请求覆盖哪些纹理像素，并生成读取请求来检索被覆盖的纹理像素，从而执行纹理采样(例如，纹理获取)。读取请求被发送到一个或更多个存储器分区单元，如图3中的一个或更多个存储器分区单元380。纹理样本过滤器单元214配置为从位置计算单元210请求的存储器分区单元接收纹理像素，并将这些纹理像素相组合以生成纹理空间坐标的单一采样颜色。

[0047] 在一个实施例中，纹理映射单元200通过以下方式来执行占用空间查询：使占用空间生成单元212确定相关的纹理占用空间查询请求覆盖了哪些纹理像素，并将被覆盖的纹理像素的表示写入一个或更多个加载/存储单元，如图5A的加载/存储单元(LSU)554。每个纹理空间维度中指定的分辨率(例如，粒度224)可以被发送到占用空间生成单元212，其在指定的分辨率内生成纹理像素覆盖的表示(例如，纹理占用空间)。在一个实施例中，占用空间生成单元212生成位图来表示纹理像素覆盖。纹理占用空间的表示(诸如包含位图的数据)可以被发送到加载/存储单元，该单元将纹理占用空间写入着色器程序可访问的存储器电路。

[0048] 在一个实施例中，被发送到着色器程序的纹理占用空间信息被存储在应用程序数据结构中，以供以后处理。应用程序数据结构可以包括工作队列，其列出在随后的场景渲染传递之前需要对哪些纹理像素进行着色。给定的纹理像素可以在工作队列中多次被列出，与场景中的一个或更多个给定几何对象的光栅化结合，从而导致工作队列中纹理像素引用的重复。工作队列中的每个纹理像素引用(包括副本)都需要着色工作。虽然这样的重复可能会产生正确的结果，但是对同一个纹理像素进行不止一次着色可能效率低。在此上下文中，去重复是指删除工作队列中对相同纹理像素的重复引用。在一个实施例中，后续处理可以包括工作队列中的纹理像素引用的去重复和由纹理像素引用所标识的着色纹理像素。给

定的纹理像素引用可以识别(但不限于)特定的纹理映射和该纹理映射中的位置。在某些实施例中,访问用于纹理映射的两个或多个LOD以生成一个纹理样本(例如,三线采样)。在这种情况下,着色器程序可以发布单独的纹理占用空间查询请求,每个请求指定不同的LOD,并将报告的纹理占用空间信息存储在应用程序数据结构中。

[0049] 在一个实施例中,着色器程序配置为将位图与去重复的纹理像素引用(本地或全局之一)一起使用,并在应用程序数据结构中记录引用的纹理像素(例如,纹理像素地址),以便稍后处理。例如,去复制的纹理像素引用可以用于创建纹理像素和/或纹理页面的列表,其将被随后的纹理过滤器操作访问,以对给定的帧进行着色。这样的列表可以用作工作队列,用于渲染用于为框架着色的纹理像素的子集。

[0050] 在另一个实施例中,纹理占用空间查询操作为执行纹理数据的延迟处理的算法提供了一个构建块,例如虚拟纹理处理、按需分页、解耦着色和纹理像素着色。当与现有的对图块化的支持相结合时,占用空间查询操作还可以用于指导稀疏纹理的动态存储器分配,以及各种类型的其他操作。

[0051] 并行处理架构

[0052] 图3示出了根据实施例的并行处理单元(PPU)300。在一个实施例中,PPU 300是在一个或更多个集成电路器件上实现的多线程处理器。PPU 300是设计用于并行处理许多线程的延迟隐藏体系架构。线程(即,执行线程)是被配置为由PPU 300执行的指令集的实例。在一个实施例中,PPU 300是图形处理单元(GPU),其被配置为实现用于处理三维(3D)图形数据的图形渲染管线,以便生成用于在显示装置(诸如液晶显示(LCD)设备)上显示的二维(2D)图像数据。在其他实施例中,PPU 300可以用于执行通用计算。尽管为了说明的目的本文提供了一个示例性并行处理器,但应特别指出的是,该处理器仅出于说明目的进行阐述,并且可使用任何处理器来补充和/或替代该处理器。

[0053] 一个或更多个PPU 300可以被配置为加速数千个高性能计算(HPC)、数据中心和机器学习应用。PPU 300可被配置为加速众多深度学习系统和应用,包括自动驾驶汽车平台、深度学习、高精度语音、图像和文本识别系统、智能视频分析、分子模拟、药物发现、疾病诊断、天气预报、大数据分析、天文学、分子动力学模拟、金融建模、机器人技术、工厂自动化、实时语言翻译、在线搜索优化和个性化用户推荐等。

[0054] 如图3所示,PPU 300包括输入/输出(I/O)单元305、前端单元315、调度器单元320、工作分配单元325、集线器330、交叉开关(Xbar)370、一个或更多个通用处理集群(GPC)350以及一个或更多个分区单元380。PPU 300可以经由一个或更多个高速NVLink 310互连连接到主机处理器或其他PPU 300。PPU 300可以经由互连302连接到主机处理器或其他外围设备。PPU 300还可以连接到包括多个存储器设备304的本地存储器。在一个实施例中,本地存储器可以包括多个动态随机存取存储器(DRAM)设备。DRAM设备可以被配置为高带宽存储器(HBM)子系统,其中多个DRAM裸晶(die)堆叠在每个设备内。

[0055] NVLink 310互连使得系统能够扩展并且包括与一个或更多个CPU结合的一个或更多个PPU 300,支持PPU 300和CPU之间的高速缓存一致性,以及CPU主控。数据和/或命令可以由NVLink 310通过集线器330发送到PPU 300的其他单元或从其发送,例如一个或更多个复制引擎、视频编码器、视频解码器、电源管理单元等(未明确示出)。结合图5B更详细地描述NVLink 310。

[0056] I/O单元305被配置为通过互连302从主机处理器(未示出)发送和接收通信(即,命令、数据等)。I/O单元305可以经由互连302直接与主机处理器通信,或通过一个或更多个中间设备(诸如内存桥)与主机处理器通信。在一个实施例中,I/O单元305可以经由互连302与一个或更多个其他处理器(例如,一个或更多个PPU 300)通信。在一个实施例中,I/O单元305实现外围组件互连高速(PCIe)接口,用于通过PCIe总线进行通信,并且互连302是PCIe总线。在替代的实施例中,I/O单元305可以实现其他类型的已知接口,用于与外部设备进行通信。

[0057] I/O单元305对经由互连302接收的分组进行解码。在一个实施例中,分组表示被配置为使PPU 300执行各种操作的命令。I/O单元305按照命令指定将解码的命令发送到PPU 300的各种其他单元。例如,一些命令可以被发送到前端单元315。其他命令可以被发送到集线器330或PPU 300的其他单元,诸如一个或更多个复制引擎、视频编码器、视频解码器、电源管理单元等(未明确示出)。换句话说,I/O单元305被配置为在PPU 300的各种逻辑单元之间和之中路由通信。

[0058] 在一个实施例中,由主机处理器执行的程序在缓冲区中对命令流进行编码,该缓冲区向PPU 300提供工作负载用于处理。工作负载可以包括要由那些指令处理的许多指令和数据。缓冲区是存储器中可由主机处理器和PPU 300两者访问(即,读/写)的区域。例如,I/O单元305可以被配置为经由通过互连302传输的存储器请求访问连接到互连302的系统存储器中的缓冲区。在一个实施例中,主机处理器将命令流写入缓冲区,然后向PPU300发送指向命令流开始的指针。前端单元315接收指向一个或更多个命令流的指针。前端单元315管理一个或更多个流,从流读取命令并将命令转发到PPU 300的各个单元。

[0059] 前端单元315耦合到调度器单元320,其配置各种GPC 350以处理由一个或更多个流定义的任务。调度器单元320被配置为跟踪与由调度器单元320管理的各种任务相关的状态信息。状态可以指示任务被指派给哪个GPC 350,该任务是活动的还是不活动的,与该任务相关联的优先级等等。调度器单元320管理一个或更多个GPC 350上的多个任务的执行。

[0060] 调度器单元320耦合到工作分配单元325,其被配置为分派任务以在GPC 350上执行。工作分配单元325可以跟踪从调度器单元320接收到的多个调度任务。在一个实施例中,工作分配单元325为每个GPC 350管理待处理(pending)任务池和活动任务池。待处理任务池可以包括多个时隙(例如,32个时隙),其包含被指派为由特定GPC 350处理的任务。活动任务池可以包括多个时隙(例如,4个时隙),用于正在由GPC 350主动处理的任务。当GPC 350完成任务的执行时,该任务从GPC 350的活动任务池中逐出,并且来自待处理任务池的其他任务之一被选择和调度以在GPC350上执行。如果GPC 350上的活动任务已经空闲,例如在等待数据依赖性被解决时,那么活动任务可以从GPC 350中逐出并返回到待处理任务池,而待处理任务池中的另一个任务被选择并调度以在GPC 350上执行。

[0061] 工作分配单元325经由XBar(交叉开关)370与一个或更多个GPC 350通信。XBar370是将PPU 300的许多单元耦合到PPU 300的其他单元的互连网络。例如,XBar 370可以被配置为将工作分配单元325耦合到特定的GPC 350。虽然没有明确示出,但PPU 300的一个或更多个其他单元也可以经由集线器330连接到XBar 370。

[0062] 任务由调度器单元320管理并由工作分配单元325分派给GPC 350。GPC 350被配置为处理任务并生成结果。结果可以由GPC 350内的其他任务消耗,经由XBar 370路由到不同

的GPC 350,或者存储在存储器304中。结果可以经由分区单元380写入存储器304,分区单元380实现用于从存储器304读取数据和向存储器304写入数据的存储器接口。结果可以通过NVLink310发送到另一个PPU 304或CPU。在一个实施例中,PPU 300包括数目为U的分区单元380,其等于耦合到PPU 300的独立且不同的存储器设备304的数目。下面将结合图4B更详细地描述分区单元380。

[0063] 在一个实施例中,主机处理器执行实现应用程序编程接口(API)的驱动程序内核,其使得能够在主机处理器上执行一个或更多个应用程序以调度操作用于在PPU 300上执行。在一个实施例中,多个计算机应用程序由PPU 300同时执行,并且PPU 300为多个计算机应用程序提供隔离、服务质量(QoS)和独立地址空间。应用程序可以生成指令(即API调用),其使得驱动程序内核生成一个或更多个任务以由PPU 300执行。驱动程序内核将任务输出到正在由PPU 300处理的一个或更多个流。每个任务可以包括一个或更多个相关线程组,本文称为线程束(warp)。在一个实施例中,线程束包括可以并行执行的32个相关线程。协作线程可以指代包括执行任务的指令并且可以通过共享存储器交换数据的多个线程。结合图5A更详细地描述线程和协作线程。

[0064] 图4A示出了根据实施例的图3的PPU 300的GPC 350。如图4A所示,每个GPC 350包括用于处理任务的多个硬件单元。在一个实施例中,每个GPC 350包括管线管理器410、预光栅操作单元(PROP)415、光栅引擎425、工作分配交叉开关(WDX)480、存储器管理单元(MMU)490以及一个或更多个数据处理集群(DPC)420。应当理解,图4A的GPC 350可以包括代替图4A中所示单元的其他硬件单元或除图4A中所示单元之外的其他硬件单元。

[0065] 在一个实施例中,GPC 350的操作由管线管理器410控制。管线管理器410管理用于处理分配给GPC 350的任务的一个或更多个DPC 420的配置。在一个实施例中,管线管理器410可以配置一个或更多个DPC 420中的至少一个来实现图形渲染管线的至少一部分。例如,DPC 420可以被配置为在可编程流式多处理器(SM)440上执行顶点着色程序。管线管理器410还可以被配置为将从工作分配单元325接收的分组路由到GPC 350中适当的逻辑单元。例如,一些分组可以被路由到PROP 415和/或光栅引擎425中的固定功能硬件单元,而其他分组可以被路由到DPC 420以供图元引擎435或SM 440处理。在一个实施例中,管线管理器410可以配置一个或更多个DPC 420中的至少一个以实现神经网络模型和/或计算管线。

[0066] PROP单元415被配置为将由光栅引擎425和DPC 420生成的数据路由到光栅操作(ROP)单元,结合图4B更详细地描述。PROP单元415还可以被配置为执行颜色混合的优化,组织像素数据,执行地址转换等。

[0067] 光栅引擎425包括被配置为执行各种光栅操作的多个固定功能硬件单元。在一个实施例中,光栅引擎425包括设置引擎、粗光栅引擎、剔除引擎、裁剪引擎、精细光栅引擎和瓦片聚合引擎。设置引擎接收变换后的顶点并生成与由顶点定义的几何图元关联的平面方程。平面方程被发送到粗光栅引擎以生成图元的覆盖信息(例如,瓦片的x、y覆盖掩码)。粗光栅引擎的输出被发送到剔除引擎,其中与未通过z-测试的图元相关联的片段被剔除,并被发送到裁剪引擎,其中位于视锥体之外的片段被裁剪掉。那些经过裁剪和剔除后留下来的片段可以被传递到精细光栅引擎,以基于由设置引擎生成的平面方程生成像素片段的属性。光栅引擎425的输出包括例如要由在DPC 420内实现的片段着色器处理的片段。

[0068] 包括在GPC 350中的每个DPC 420包括M管线控制器(MPC)430、图元引擎435和一个

或更多个SM 440。MPC 430控制DPC 420的操作，将从管线管理器410接收到的分组路由到DPC 420中的适当单元。例如，与顶点相关联的分组可以被路由到图元引擎435，图元引擎435被配置为从存储器304提取与顶点相关联的顶点属性。相反，与着色程序相关联的分组可以被发送到SM 440。

[0069] SM 440包括被配置为处理由多个线程表示的任务的可编程流式处理器。每个SM 440是多线程的并且被配置为同时执行来自特定线程组的多个线程(例如，32个线程)。在一个实施例中，SM 440实现SIMD(单指令、多数据)体系架构，其中线程组(即，warp)中的每个线程被配置为基于相同的指令集来处理不同的数据集。线程组中的所有线程都执行相同的指令。在另一个实施例中，SM 440实现SIMT(单指令、多线程)体系架构，其中线程组中的每个线程被配置为基于相同的指令集处理不同的数据集，但是其中线程组中的各个线程在执行期间被允许发散。在一个实施例中，为每个线程束维护程序计数器、调用栈和执行状态，当线程束内的线程发散时，使线程束和线程束中的串行执行之间的并发成为可能。在另一个实施例中，为每个单独的线程维护程序计数器、调用栈和执行状态，从而在线程束内和线程束之间的所有线程之间实现相等的并发。当为每个单独的线程维护执行状态时，执行相同指令的线程可以被收敛并且并行执行以获得最大效率。下面结合图5A更详细地描述SM 440。

[0070] MMU 490提供GPC 350和分区单元380之间的接口。MMU 490可以提供虚拟地址到物理地址的转换、存储器保护以及存储器请求的仲裁。在一个实施例中，MMU 490提供用于执行从虚拟地址到存储器304中的物理地址的转换的一个或更多个转换后备缓冲器(TLB)。

[0071] 图4B示出了根据实施例的图3的PPU 300的存储器分区单元380。如图4B所示，存储器分区单元380包括光栅操作(ROP)单元450、二级(L2)高速缓存460和存储器接口470。存储器接口470耦合到存储器304。存储器接口470可以实现用于高速数据传输的32、64、128、1024位数据总线等。在一个实施例中，PPU 300合并了U个存储器接口470，每对分区单元380有一个存储器接口470，其中每对分区单元380连接到对应的存储器设备304。例如，PPU 300可以连接到多达Y个存储器设备304，诸如高带宽存储器堆叠或图形双数据速率版本5的同步动态随机存取存储器或其他类型的持久存储器。

[0072] 在一个实施例中，存储器接口470实现HBM2存储器接口，并且Y等于U的一半。在一个实施例中，HBM2存储器堆叠位于与PPU 300相同的物理封装上，提供与常规GDDR5SDRAM系统相比显著的功率高和面积节约。在一个实施例中，每个HBM2堆叠包括四个存储器裸晶并且Y等于4，其中HBM2堆叠包括每个裸晶两个128位通道，总共8个通道和1024位的数据总线宽度。

[0073] 在一个实施例中，存储器304支持单错校正双错检测(SECDED)纠错码(ECC)以保护数据。对于对数据损毁敏感的计算机应用程序，ECC提供了更高的可靠性。在大型集群计算环境中，可靠性尤其重要，其中PPU300处理非常大的数据集和/或长时间运行应用程序。

[0074] 在一个实施例中，PPU 300实现多级存储器分层结构。在一个实施例中，存储器分区单元380支持统一存储器以为CPU和PPU 300存储器提供单个统一的虚拟地址空间，使能虚拟存储器系统之间的数据共享。在一个实施例中，由PPU 300对位于其他处理器上的存储器的访问频率被跟踪，以确保存储器页面被移动到更频繁地访问页面的PPU 300的物理存储器。在一个实施例中，NVLink 310支持地址转换服务，其允许PPU 300直接访问CPU的页表

并且提供由PPU 300对CPU存储器的完全访问。

[0075] 在一个实施例中,复制引擎在多个PPU 300之间或在PPU 300与CPU之间传输数据。复制引擎可以为未映射到页表的地址生成页面错误。然后,存储器分区单元380可以服务页面错误,将地址映射到页表中,之后复制引擎可以执行传输。在常规系统中,针对多个处理器之间的多个复制引擎操作固定存储器(即,不可分页),其显著减少了可用存储器。由于硬件分页错误,地址可以传递到复制引擎而不用担心存储器页面是否驻留,并且复制过程是否透明。

[0076] 来自存储器304或其他系统存储器的数据可以由存储器分区单元380取回并存储在L2高速缓存460中,L2高速缓存460位于芯片上并且在各个GPC 350之间共享。如图所示,每个存储器分区单元380包括与对应的存储器设备304相关联的L2高速缓存460的一部分。然后可以在GPC 350内的多个单元中实现较低级高速缓存。例如,每个SM 440可以实现一级(L1)高速缓存。L1高速缓存是专用于特定SM 440的专用存储器。来自L2高速缓存460的数据可以被获取并存储在每个L1高速缓存中,以在SM 440的功能单元中进行处理。L2高速缓存460被耦合到存储器接口470和XBar370。

[0077] ROP单元450执行与像素颜色相关的图形光栅操作,诸如颜色压缩、像素混合等。ROP单元450还与光栅引擎425一起实现深度测试,从光栅引擎425的剔除引擎接收与像素片段相关联的样本位置的深度。测试与片段关联的样本位置相对于深度缓冲区中的对应深度的深度。如果片段通过样本位置的深度测试,则ROP单元450更新深度缓冲区并将深度测试的结果发送给光栅引擎425。将理解的是,分区单元380的数量可以不同于GPC350的数量,并且因此每个ROP单元450可以耦合到每个GPC 350。ROP单元450跟踪从不同GPC 350接收到的分组并且确定由ROP单元450生成的结果通过Xbar 370被路由到哪个GPC 350。尽管ROP单元450包括在图4B中的存储器分区单元380内,但是在其他实施例中,ROP单元450可以在存储器分区单元380之外。例如,ROP单元450可以驻留在GPC 350或另一个单元中。

[0078] 图5A示出了根据实施例的图4A的流式多处理器440。如图5A所示,SM 440包括指令高速缓存505、一个或更多个调度器单元510、寄存器文件520、一个或更多个处理核心550、一个或更多个特殊功能单元(SFU)552、一个或更多个加载/存储单元(LSU)554、互连网络580、共享存储器/L1高速缓存570。

[0079] 如上所述,工作分配单元325调度任务以在PPU 300的GPC 350上执行。任务被分配给GPC 350内的特定DPC 420,并且如果任务与着色器程序相关联,则该任务可以被分配给SM 440。调度器单元510接收来自工作分配单元325的任务并且管理指派给SM 440的一个或更多个线程块的指令调度。调度器单元510调度线程块以作为并行线程的线程束执行,其中每个线程块被分配至少一个线程束。在一个实施例中,每个线程束执行32个线程。调度器单元510可以管理多个不同的线程块,将线程束分配给不同的线程块,然后在每个时钟周期期间将来自多个不同的协作组的指令分派到各个功能单元(即,核心550、SFU 552和LSU 554)。

[0080] 协作组是用于组织通信线程组的编程模型,其允许开发者表达线程正在进行通信所采用的粒度,使得能够表达更丰富、更高效的并行分解。协作启动API支持线程块之间的同步性,以执行并行算法。常规的编程模型为同步协作线程提供了单一的简单结构:跨线程块的所有线程的栅栏(barrier)(即,syncthreads()函数)。然而,程序员通常希望以小于

线程块粒度的粒度定义线程组，并在所定义的组内同步，以集体的全组功能接口(collective group-wide function interface)的形式使能更高的性能、设计灵活性和软件重用。

[0081] 协作组使得程序员能够在子块(即，像单个线程一样小)和多块粒度处明确定义线程组并且执行集体操作，诸如协作组中的线程上的同步性。编程模型支持跨软件边界的干净组合，以便库和效用函数可以在本地环境中安全地同步，而无需对收敛进行假设。协作组图元启用合作伙伴并行的新模式，包括生产者-消费者并行、机会主义并行以及跨整个线程块网格的全局同步。

[0082] 分派单元515被配置为向一个或更多个功能单元传送指令。在该实施例中，调度器单元510包括两个分派单元515，其使得能够在每个时钟周期期间调度来自相同线程束的两个不同指令。在替代实施例中，每个调度器单元510可以包括单个分派单元515或附加分派单元515。

[0083] 每个SM 440包括寄存器文件520，其提供用于SM 440的功能单元的一组寄存器。在一个实施例中，寄存器文件520在每个功能单元之间被划分，使得每个功能单元被分配寄存器文件520的专用部分。在另一个实施例中，寄存器文件520在由SM 440执行的不同线程束之间被划分。寄存器文件520为连接到功能单元的数据路径的操作数提供临时存储。

[0084] 每个SM 440包括L个处理核心550。在一个实施例中，SM 440包括大量(例如128个等)不同的处理核心550。每个核心550可以包括完全管线化的、单精度、双精度和/或混合精度处理单元，其包括浮点运算逻辑单元和整数运算逻辑单元。在一个实施例中，浮点运算逻辑单元实现用于浮点运算的IEEE 754-2008标准。在一个实施例中，核心550包括64个单精度(32位)浮点核心、64个整数核心、32个双精度(64位)浮点核心和8个张量核心(tensor core)。

[0085] 张量核心被配置为执行矩阵运算，并且在一个实施例中，一个或更多个张量核心被包括在核心550中。具体地，张量核心被配置为执行深度学习矩阵运算，诸如用于神经网络训练和推理的卷积运算。在一个实施例中，每个张量核心在4×4矩阵上运算并且执行矩阵乘法和累加运算D＝A×B+C，其中A、B、C和D是4×4矩阵。

[0086] 在一个实施例中，矩阵乘法输入A和B是16位浮点矩阵，而累加矩阵C和D可以是16位浮点或32位浮点矩阵。张量核心在16位浮点输入数据以及32位浮点累加上运算。16位浮点乘法需要64次运算，产生全精度的积，然后使用32位浮点与4×4×4矩阵乘法的其他中间积相加来累加。在实践中，张量核心用于执行由这些较小的元素建立的更大的二维或更高维的矩阵运算。API(诸如CUDA 9C++API)公开了专门的矩阵加载、矩阵乘法和累加以及矩阵存储运算，以便有效地使用来自CUDA-C++程序的张量核心。在CUDA层面，线程束级接口假定16×16尺寸矩阵跨越线程束的全部32个线程。

[0087] 每个SM 440还包括执行特殊函数(例如，属性评估、倒数平方根等)的M个SFU 552。在一个实施例中，SFU 552可以包括树遍历单元，其被配置为遍历分层树数据结构。在一个实施例中，SFU 552可以包括被配置为执行纹理贴图过滤操作的纹理单元。在一个实施例中，纹理单元被配置为从存储器304加载纹理贴图(例如，纹理像素的2D阵列)并且对纹理贴图进行采样以产生经采样的纹理值，用于在由SM 440执行的着色器程序中使用。在一个实施例中，纹理贴图被存储在共享存储器/L1高速缓存470中。纹理单元实现纹理操作，诸如使

用mip贴图(即,不同细节层次的纹理贴图)的过滤操作。在一个实施例中,每个SM 440包括两个纹理单元。在一个实施例中,SFU 552中的至少一个被配置为包括纹理映射单元200的至少一个实例。

[0088] 每个SM 440还包括N个LSU 554,其实现共享存储器/L1高速缓存570和寄存器文件520之间的加载和存储操作。每个SM 440包括将每个功能单元连接到寄存器文件520以及将LSU 554连接到寄存器文件520、共享存储器/L1高速缓存570的互连网络580。在一个实施例中,互连网络580是交叉开关,其可以被配置为将任何功能单元连接到寄存器文件520中的任何寄存器,以及将LSU 554连接到寄存器文件和共享存储器/L1高速缓存570中的存储器位置。

[0089] 共享存储器/L1高速缓存570是片上存储器阵列,其允许数据存储和SM 440与图元引擎435之间以及SM 440中的线程之间的通信。在一个实施例中,共享存储器/L1高速缓存570包括128KB的存储容量并且在从SM440到分区单元380的路径中。共享存储器/L1高速缓存570可以用于高速缓存读取和写入。共享存储器/L1高速缓存570、L2高速缓存460和存储器304中的一个或更多个是后备存储。

[0090] 将数据高速缓存和共享存储器功能组合成单个存储器块为两种类型的存储器访问提供最佳的总体性能。该容量可由程序用作不使用共享存储器的高速缓存。例如,如果将共享存储器配置为使用一半容量,则纹理和加载/存储操作可以使用剩余容量。在共享存储器/L1高速缓存570内的集成使共享存储器/L1高速缓存570起到用于流式传输数据的高吞吐量管道的作用,并且同时提供对频繁重用数据的高带宽和低延迟的访问。

[0091] 当被配置用于通用并行计算时,与图形处理相比,可以使用更简单的配置。具体地,图3所示的固定功能图形处理单元被绕过,创建了更简单的编程模型。在通用并行计算配置中,工作分配单元325将线程块直接指派并分配给DPC 420。块中的线程执行相同的程序,使用计算中的唯一线程ID来确保每个线程生成唯一结果,使用SM 440执行程序并执行计算,使用共享存储器/L1高速缓存570以在线程之间通信,以及使用LSU 554通过共享存储器/L1高速缓存570和存储器分区单元380读取和写入全局存储器。当被配置用于通用并行计算时,SM 440还可以写入调度器单元320可用来在DPC 420上启动新工作的命令。

[0092] PPU 300可以被包括在台式计算机、膝上型计算机、平板电脑、服务器、超级计算机、智能电话(例如,无线、手持设备)、个人数字助理(PDA)、数码相机、运载工具、头戴式显示器、手持式电子设备等中。在一个实施例中,PPU 300包含在单个半导体衬底上。在另一个实施例中,PPU 300与一个或更多个其他器件(诸如附加PPU 300、存储器204、精简指令集计算机(RISC)CPU、存储器管理单元(MMU)、数字-模拟转换器(DAC)等)一起被包括在片上系统(SoC)上。

[0093] 在一个实施例中,PPU 300可以被包括在图形卡上,图形卡包括一个或更多个存储器设备304。图形卡可以被配置为与台式计算机的主板上的PCIe插槽接口。在又一个实施例中,PPU 300可以是包含在主板的芯片集中的集成图形处理单元(iGPU)或并行处理器。

[0094] 示例性计算系统

[0095] 具有多个GPU和CPU的系统被用于各种行业,因为开发者在应用(诸如人工智能计算)中暴露和利用更多的并行性。在数据中心、研究机构和超级计算机中部署具有数十至数千个计算节点的高性能GPU加速系统,以解决更大的问题。随着高性能系统内处理设备数量

的增加,通信和数据传输机制需要扩展以支持该增加的带宽。

[0096] 图5B是根据实施例的使用图3的PPU 300实现的处理系统500的概念图。处理系统500可以被配置为实现图1A中所示的方法110。处理系统500包括CPU 530、交换机510和多个PPU 300中的每一个以及相应的存储器304。NVLink 310提供每个PPU 300之间的高速通信链路。尽管图5B中示出了特定数量的NVLink 310和互连302连接,但是连接到每个PPU 300和CPU 530的连接的数量可以改变。交换机510在互连302和CPU 530之间接口。PPU 300、存储器304和NVLink 310可以位于单个半导体平台上以形成并行处理模块525。在一个实施例中,交换机510支持两个或更多个在各种不同连接和/或链路之间接口的协议。

[0097] 在另一个实施例(未示出)中,NVLink 310在每个PPU 300和CPU 530之间提供一个或更多个高速通信链路,并且交换机510在互连302和每个PPU 300之间进行接口。PPU 300、存储器304和互连302可以位于单个半导体平台上以形成并行处理模块525。在又一个实施例(未示出)中,互连302在每个PPU 300和CPU 530之间提供一个或更多个通信链路,并且交换机510使用NVLink 310在每个PPU 300之间进行接口,以在PPU 300之间提供一个或更多个高速通信链路。在另一个实施例(未示出)中,NVLink 310在PPU 300和CPU 530之间通过交换机510提供一个或更多个高速通信链路。在另一个实施例(未示出)中,互连302在每个PPU 300直接地提供一个或更多个通信链路。可以使用与NVLink 310相同的协议将一个或更多个NVLink 310高速通信链路实现为物理NVLink互连或者片上或裸晶上互连。

[0098] 在本说明书的上下文中,单个半导体平台可以指在裸晶或芯片上制造的唯一的单一的基于半导体的集成电路。应该注意的是,术语单个半导体平台也可以指具有增加的连接的多芯片模块,其模拟片上操作并通过利用常规总线实现方式进行实质性改进。当然,根据用户的需要,各种电路或器件还可以分开放置或以半导体平台的各种组合来放置。可选地,并行处理模块525可以被实现为电路板衬底,并且PPU 300和/或存储器304中的每一个可以是封装器件。在一个实施例中,CPU 530、交换机510和并行处理模块525位于单个半导体平台上。

[0099] 在一个实施例中,每个NVLink 310的信令速率是20到25千兆位/秒,并且每个PPU 300包括六个NVLink 310接口(如图5B所示,每个PPU 300包括五个NVLink 310接口)。每个NVLink 310在每个方向上提供25千兆位/秒的数据传输速率,其中六条链路提供300千兆位/秒。当CPU 530还包括一个或更多个NVLink 310接口时,NVLink 310可专门用于如图5B所示的PPU到PPU通信,或者PPU到PPU以及PPU到CPU的某种组合。

[0100] 在一个实施例中,NVLink 310允许从CPU 530到每个PPU 300的存储器304的直接加载/存储/原子访问。在一个实施例中,NVLink 310支持一致性操作,允许从存储器304读取的数据被存储在CPU 530的高速缓存分层结构中,减少了CPU 530的高速缓存访问延迟。在一个实施例中,NVLink 310包括对地址转换服务(ATS)的支持,允许PPU 300直接访问CPU 530内的页表。一个或更多个NVLink 310还可以被配置为以低功率模式操作。

[0101] 图5C示出了示例性系统565,其中可以实现各种先前实施例的各种体系架构和/或功能。示例性系统565可以被配置为实现图1A中所示的方法110。

[0102] 如图所示,提供系统565,其包括连接到通信总线575的至少一个中央处理单元530。通信总线575可以使用任何合适的协议来实现,诸如PCI(外围组件互连)、PCI-Express、AGP(加速图形端口)、超传输或任何其他总线或一个或更多个点对点通信协议。系

统565还包括主存储器540。控制逻辑(软件)和数据被存储在主存储器540中,主存储器540可以采取随机存取存储器(RAM)的形式。

[0103] 系统565还包括输入设备560、并行处理系统525和显示设备545,即常规CRT(阴极射线管)、LCD(液晶显示器)、LED(发光二极管)、等离子显示器等。可以从输入设备560(例如键盘、鼠标、触摸板、麦克风等)接收用户输入。前述模块和/或设备中的每一个甚至可以位于单个半导体平台上以形成系统565。可选地,根据用户的需要,各个模块还可以分开放置或以半导体平台的各种组合来放置。

[0104] 此外,系统565可以出于通信目的通过网络接口535耦合到网络(例如,电信网络、局域网(LAN)、无线网络、广域网(WAN)(诸如因特网)、对等网络、电缆网络等)。

[0105] 系统565还可以包括辅助存储(未示出)。辅助存储610包括例如硬盘驱动器和/或可移除存储驱动器、代表软盘驱动器、磁带驱动器、光盘驱动器、数字多功能盘(DVD)驱动器、记录设备、通用串行总线(USB)闪存。可移除存储驱动器以众所周知的方式从可移除存储单元读取和/或写入可移除存储单元。

[0106] 计算机程序或计算机控制逻辑算法可以存储在主存储器540和/或辅助存储中。这些计算机程序在被执行时使得系统565能够执行各种功能。存储器540、存储和/或任何其他存储是计算机可读介质的可能示例。

[0107] 各种在先附图的体系架构和/或功能可以在通用计算机系统、电路板系统、专用于娱乐目的的游戏控制台系统、专用系统和/或任何其他所需的系统的上下文中实现。例如,系统565可以采取台式计算机、膝上型计算机、平板电脑、服务器、超级计算机、智能电话(例如,无线、手持设备)、个人数字助理(PDA)、数字相机、运载工具、头戴式显示器、手持式电子设备、移动电话设备、电视机、工作站、游戏控制台、嵌入式系统和/或任何其他类型的逻辑的形式。

[0108] 图形处理管线

[0109] 在一个实施例中,PPU 300包括图形处理单元(GPU)。PPU 300被配置为接收指定用于处理图形数据的着色程序的命令。图形数据可以被定义为一组图元,例如点、线、三角形、四边形、三角形带等。典型地,图元包括指定图元的多个顶点(例如,在模型空间坐标系中)的数据以及与图元的每个顶点相关联的属性。PPU 300可以被配置为处理图元以生成帧缓冲区(即,用于显示器的像素中的每一个的像素数据)。

[0110] 应用程序将场景的模型数据(即,顶点和属性的集合)写入存储器(诸如系统存储器或存储器304)。模型数据定义可能在显示器上可见的对象中的每一个。然后应用程序对驱动程序内核进行API调用,其请求要被渲染和显示的模型数据。驱动程序内核读取模型数据并将命令写入一个或更多个流以执行操作来处理模型数据。这些命令可以参考要在PPU 300的SM440上实现的不同着色程序,包括顶点着色、外壳着色、域着色、几何着色和像素着色中的一个或更多个。例如,SM 440中的一个或更多个可以被配置为执行顶点着色程序,其处理由模型数据定义的多个顶点。在一个实施例中,不同的SM 440可以被配置为同时执行不同的着色程序。例如,SM 440的第一子集可以被配置为执行顶点着色程序,而SM 440的第二子集可以被配置为执行像素着色程序。SM 440的第一子集处理顶点数据以产生经处理的顶点数据,并将经处理的顶点数据写入L2高速缓存460和/或存储器304。在经处理的顶点数据被光栅化(即,从三维数据转换成屏幕空间中的二维数据)以产生片段数据之后,SM 440

的第二子集执行像素着色以产生经处理的片段数据，然后将其与其他经处理的片段数据混合并被写入存储器304中的帧缓冲区。顶点着色程序和像素着色程序可以同时执行，以管线方式处理来自同一场景的不同数据，直到该场景的所有模型数据已经被渲染到帧缓冲区。然后，帧缓冲区的内容被传送到显示控制器以在显示设备上显示。

[0111] 图6是根据实施例的由图3的PPU 300实现的图形处理管线600的概念图。图形处理管线600是被实现以从3D几何数据生成2D计算机生成图像的处理步骤的抽象流程图。众所周知，管线架构可以通过将操作分成多个阶段来更高效地执行长延迟操作，其中每个阶段的输出耦合到下一个连续阶段的输入。因此，图形处理管线600接收从图形处理管线600的一个阶段传送到下一阶段的输入数据601，以生成输出数据602。在一个实施例中，图形处理管线600可表示由OpenGL®API定义的图形处理管线。作为选择，图形处理管线600可以在先前附图和/或一个或更多个任何后续附图的功能和架构的上下文中实现。

[0112] 如图6所示，图形处理管线600包括包含多个阶段的管线架构。这些阶段包括但不限于数据组装阶段610、顶点着色阶段620、图元组装阶段630、几何着色阶段640、视口缩放、剔除和裁剪(scale,cull,and clip,SCC)阶段650、光栅化阶段660、片段着色阶段670和光栅操作阶段680。在一个实施例中，输入数据601包括命令，其配置处理单元以实现图形处理管线600的阶段，并配置几何图元(例如，点、线、三角形、四边形、三角形带或扇形等)以由这些阶段处理。输出数据602可以包括像素数据(即，颜色数据)，其被复制到存储器中的帧缓冲区或其他类型的表面数据结构中。

[0113] 数据组装阶段610接收输入数据601，其指定用于高阶表面、图元等的顶点数据。数据组装阶段610收集临时存储或队列中的顶点数据，诸如通过从主机处理器接收包括指向存储器中的缓冲区的指针的命令并从该缓冲区读取顶点数据。顶点数据然后被传送到顶点着色阶段620以进行处理。

[0114] 顶点着色阶段620通过对顶点中的每一个执行一次一组操作(即，顶点着色器或程序)来处理顶点数据。顶点可以例如被指定为与一个或更多个顶点属性(例如，颜色、纹理坐标、表面法线等)相关联的4坐标向量(即，<x,y,z,w>)。顶点着色阶段620可以操纵各个顶点属性，诸如位置、颜色、纹理坐标等。换句话说，顶点着色阶段620对与顶点相关联的顶点坐标或其他顶点属性执行操作。这些操作通常包括光照操作(即，修改顶点的颜色属性)和变换操作(即，修改顶点的坐标空间)。例如，可以使用对象坐标空间中的坐标来指定顶点，其通过将坐标乘以矩阵进行变换，该矩阵将坐标从对象坐标空间转换到世界空间或归一化设备坐标(normalized-device-coordinate,NCD)空间。顶点着色阶段620生成被传送到图元组装阶段630的经变换的顶点数据。

[0115] 图元组装阶段630收集由顶点着色阶段620输出的顶点并且将顶点分组成几何图元以由几何着色阶段640处理。例如，图元组装阶段630可以被配置为将每三个连续顶点分组为用于传送到几何着色阶段640的几何图元(即，三角形)。在一些实施例中，特定顶点可以被重新用于连续几何图元(例如，三角形带中的两个连续三角形可以共享两个顶点)。图元组装阶段630将几何图元(即，相关联的顶点的集合)传送到几何着色阶段640。

[0116] 几何着色阶段640通过对几何图元执行一组操作(即，几何着色器或程序)来处理几何图元。曲面细分(tessellation)操作可以从每个几何图元生成一个或更多个几何图元。换言之，几何着色阶段640可以将每个几何图元细分为两个或更多个几何图元的更精细

的网格，以由图形处理管线600的其余部分进行处理。几何着色阶段640将几何图元传送到视口SCC阶段650。

[0117] 在一个实施例中，图形处理管线600可以在流式多处理器和顶点着色阶段620、图元组装阶段630、几何着色阶段640、片段着色阶段670和/或与其相关联的硬件/软件内操作，可顺序地执行处理操作。一旦顺序处理操作完成，在一个实施例中，视口SCC阶段650可以利用数据。在一个实施例中，由图形处理管线600中的阶段的一个或更多个处理的图元数据可以被写入高速缓存(例如，L1高速缓存、顶点高速缓存等)中。在这种情况下，在一个实施例中，视口SCC阶段650可以访问高速缓存中的数据。在一个实施例中，视口SCC阶段650和光栅化阶段660被实现为固定功能电路。

[0118] 视口SCC阶段650执行几何图元的视口缩放、剔除和裁剪。正被渲染的每个表面都与抽象相机位置相关联。相机位置表示正观看该场景的观看者的位置并定义了包围该场景的对象的视锥体。视锥体可以包括观看平面、后平面和四个裁剪平面。完全位于视锥体之外的任何几何图元都可被剔除(即丢弃)，因为这些几何图元将不会对最终渲染的场景做出贡献。部分位于视锥体内并且部分位于视锥体外的任何几何图元可以被裁剪(即，转换为被包围在视锥体内的新的几何图元)。此外，可以基于视锥体的深度来对每个几何图元进行缩放。然后将所有可能可见的几何图元传送到光栅化阶段660。

[0119] 光栅化阶段660将3D几何图元转换成2D片段(例如，能够用于显示等)。光栅化阶段660可以被配置为利用几何图元的顶点来设置一组平面方程，从中可以内插各种属性。光栅化阶段660还可以计算多个像素的覆盖掩码，其指示像素的一个或更多个样本位置是否拦截几何图元。在一个实施例中，还可以执行z测试以确定几何图元是否被已经被光栅化的其他几何图元遮挡。光栅化阶段660生成片段数据(即，与每个被覆盖像素的特定样本位置相关联的内插顶点属性)，其被传送到片段着色阶段670。

[0120] 片段着色阶段670通过对片段中的每一个执行一组操作(即，片段着色器或程序)来处理片段数据。片段着色阶段670可以生成片段的像素数据(即，颜色值)，诸如通过使用片段的内插纹理坐标执行光照操作或采样纹理贴图。片段着色阶段670生成像素数据，其被发送到光栅操作阶段680。

[0121] 光栅操作阶段680可对像素数据执行各种操作，诸如执行阿尔法测试、模板测试(stencil test)以及将像素数据与对应于与像素相关联的其他片段的其他像素数据混合。当光栅操作阶段680已经完成对像素数据(即，输出数据602)的处理时，可以将像素数据写入渲染目标，诸如帧缓冲区、颜色缓冲区等。

[0122] 应当领会，除上述阶段中的一个或更多个以外或代替上述阶段中的一个或更多个，一个或更多个额外的阶段可以被包括在图形处理管线600中。抽象图形处理管线的各种实现方式可以实现不同的阶段。此外，在一些实施例中，上述阶段中的一个或更多个可以从图形处理管线中排除(诸如几何着色阶段640)。其他类型的图形处理管线被认为是在本公开的范围内所构想的。此外，图形处理管线600的任何阶段可以由图形处理器(诸如PPU300)内的一个或更多个专用硬件单元来实现。图形处理管线600的其他阶段可以由可编程硬件单元(诸如PPU 300的SM 440)来实现。

[0123] 图形处理管线600可以经由由主机处理器(诸如CPU)执行的应用程序来实现。在一个实施例中，设备驱动程序可以实现应用程序编程接口(API)，其定义可以被应用程序利用

以生成用于显示的图形数据的各种功能。设备驱动程序是软件程序，其包括控制PPU 300的操作的多个指令。API为程序员提供抽象，其允许程序员利用专用图形硬件(诸如PPU 300)来生成图形数据而不要求程序员利用PPU 300的特定指令集。应用程序可以包括被路由到PPU 300的设备驱动程序的API调用。设备驱动程序解释API调用并执行各种操作以响应API调用。在一些情况下，设备驱动程序可以通过在CPU上执行指令来执行操作。在其他情况下，设备驱动程序可以至少部分地通过利用CPU和PPU 300之间的输入/输出接口在PPU 300上启动操作来执行操作。在一个实施例中，设备驱动程序被配置为利用PPU300的硬件来实现图形处理管线600。

[0124] 可以在PPU 300内执行各种程序以便实现图形处理管线600的各个阶段。例如，设备驱动程序可以启动PPU 300上的内核以在一个SM 440(或多个SM 440)上执行顶点着色阶段620。设备驱动程序(或由PPU 400执行的初始内核)还可启动PPU 400上的其他内核以执行图形处理管线600的其他阶段，诸如几何着色阶段640和片段着色阶段670。另外，图形处理管线600的阶段中的一些可以在固定单元硬件(诸如在PPU 400内实现的光栅器或数据组装器)上实现。应当领会，在被SM 440上的后续内核处理之前，来自一个内核的结果可以由一个或更多个中间固定功能硬件单元处理。

[0125] 虽然上面已经描述了各种实施例，但是应该理解，它们只是作为示例而不是限制来提供的。因此，本申请的广度和范围不应受上述示例性实施例中的任何一个的限制，而应仅根据以下和之后提交的权利要求及其等同物来定义。

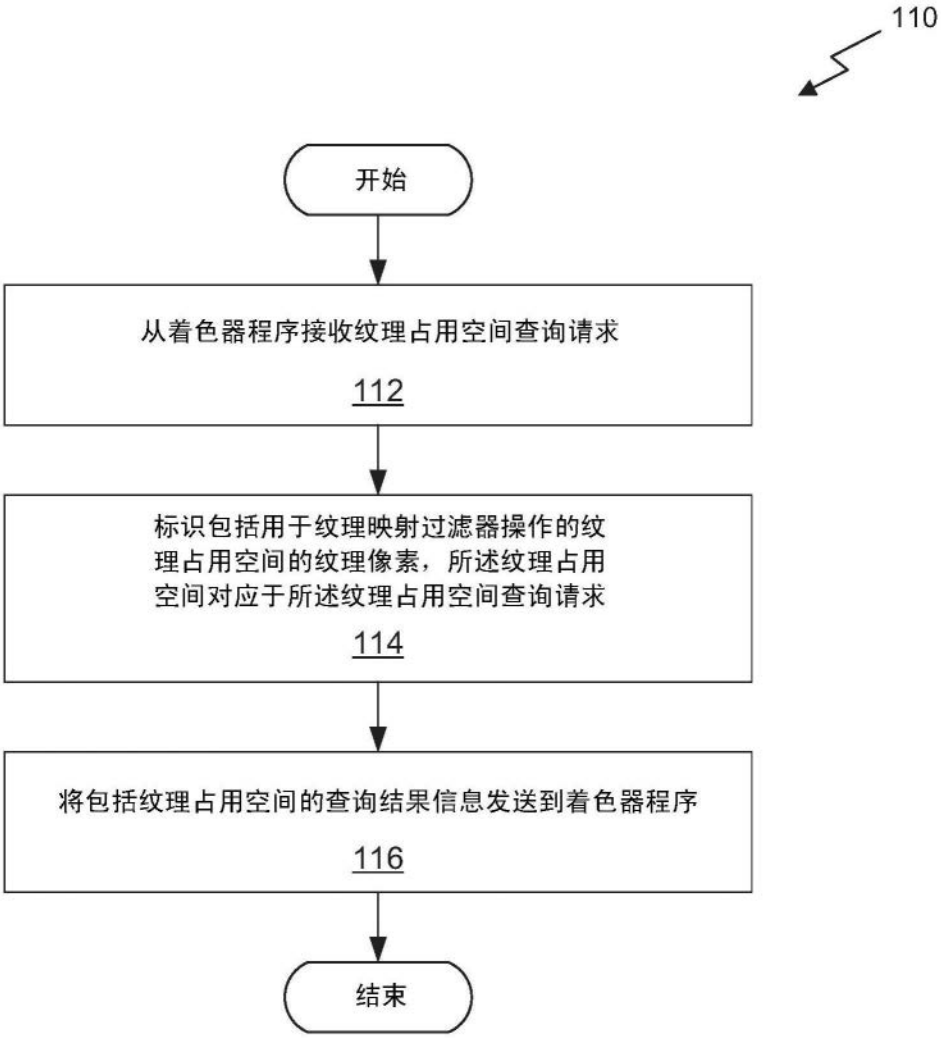

110
开始
从着色器程序接收纹理占用空间查询请求
112
标识包括用于纹理映射过滤器操作的纹理占用空间的纹理像素，所述纹理占用空间对应于所述纹理占用空间查询请求
114
将包括纹理占用空间的查询结果信息发送到着色器程序
116
结束

图1A

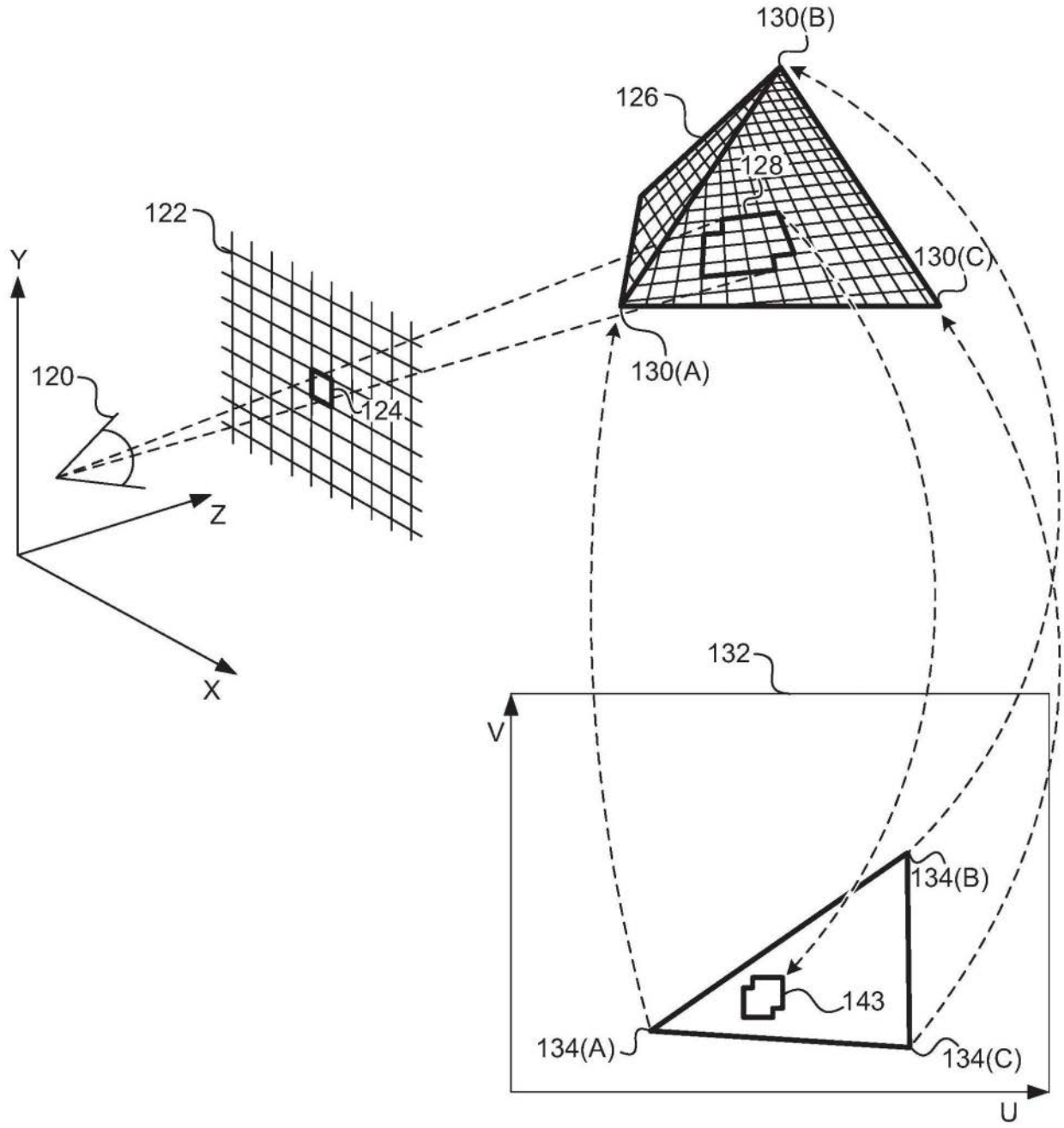

130(B)
126
128
122
130(C)
Y
120
130(A)
124
Z
X
132
V
134(B)
143
134(A)
134(C)
U

图1B

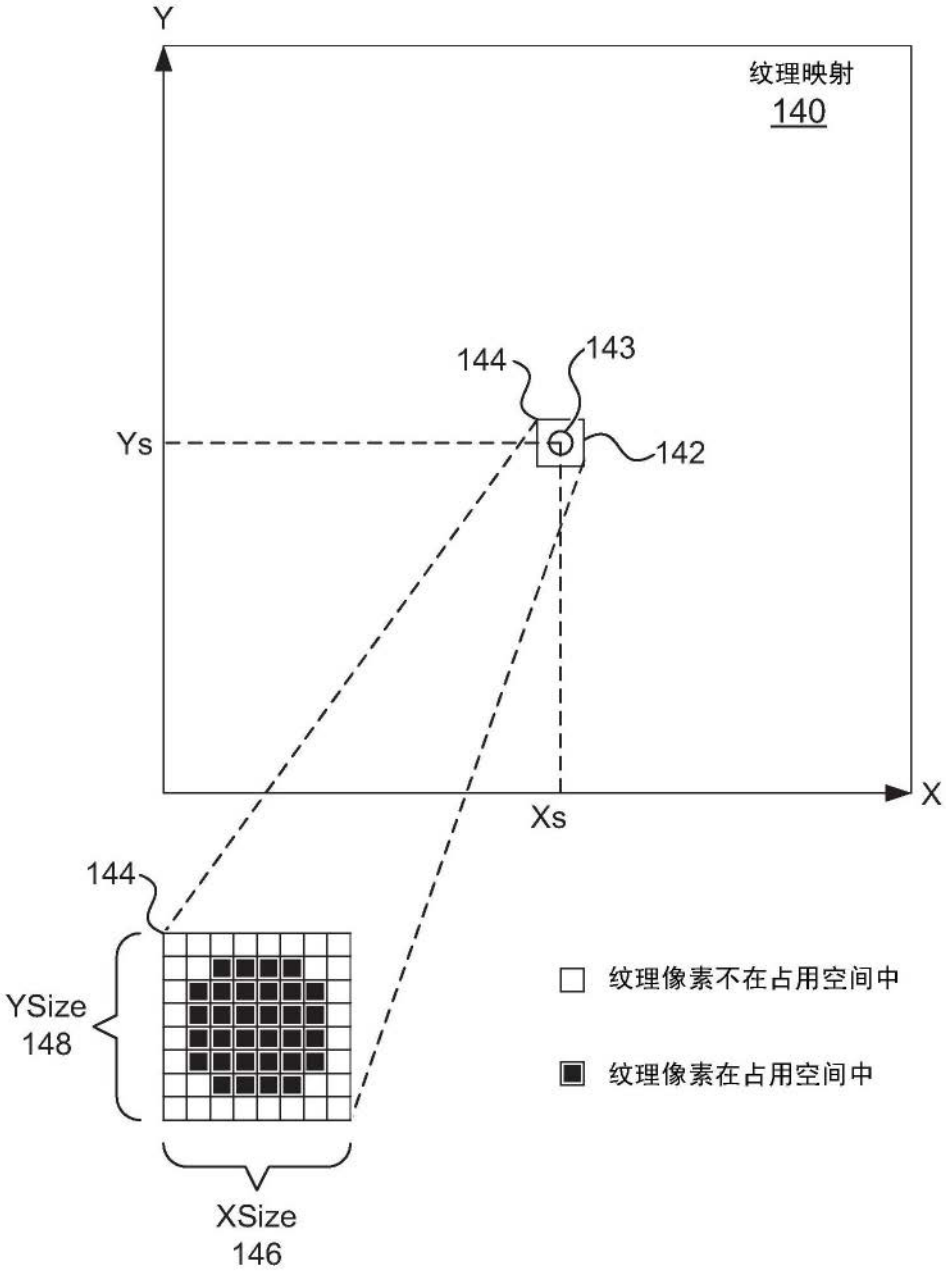

Y
纹理映射
140
144
143
Ys
142
X
Xs
144
YSize
148
纹理像素不在占用空间中
纹理像素在占用空间中
XSize
146

图1C

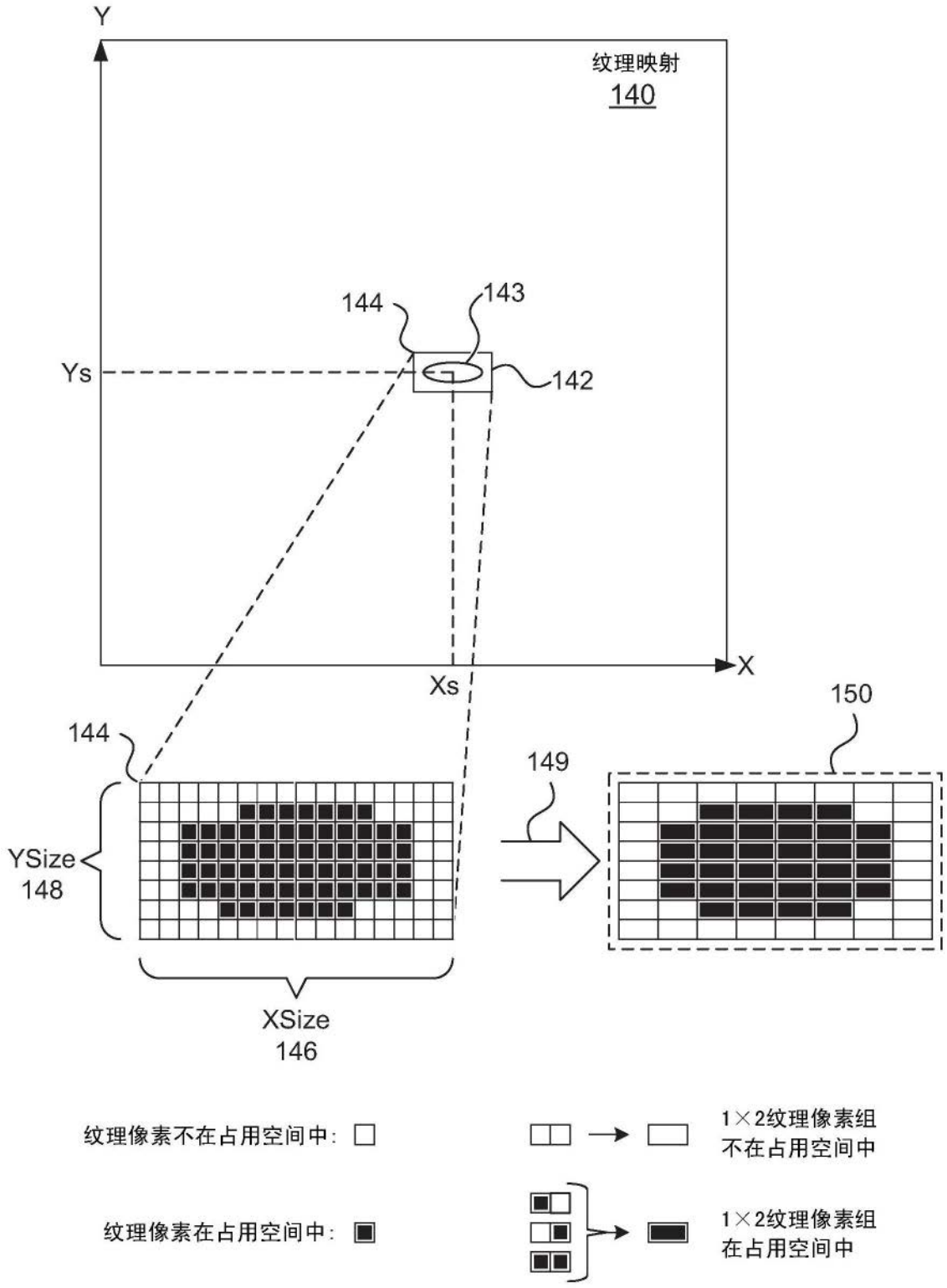

Y
纹理映射
140
144
143
142
Ys
X
Xs
150
144
149
YSize
148
XSize
146
纹理像素不在占用空间中:
1×2纹理像素组
不在占用空间中
纹理像素在占用空间中:
1×2纹理像素组
在占用空间中

图1D

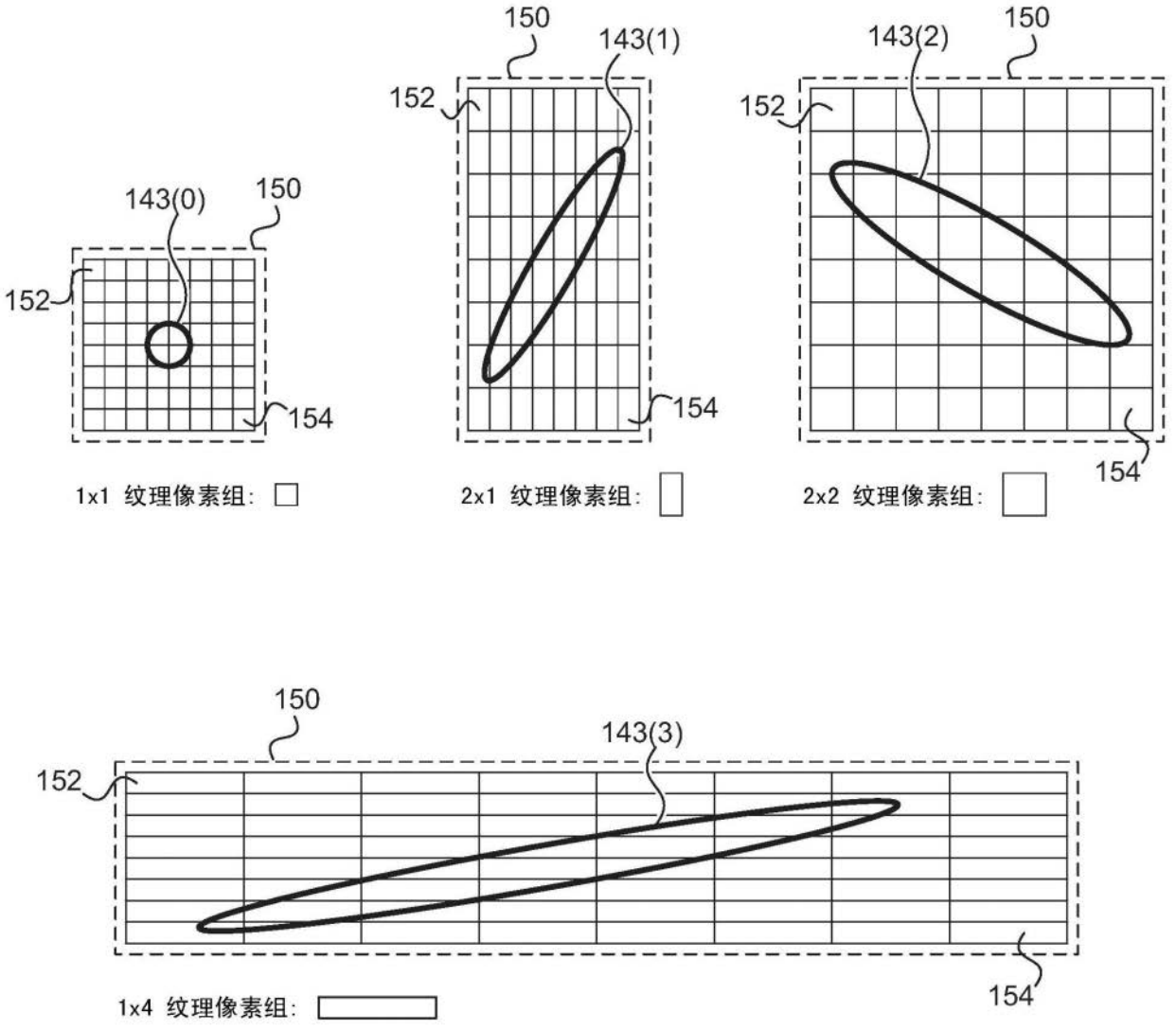

150
143(1)
152
154
143(0)
150
152
154
143(2)
150
152
154
1x1 纹理像素组:
2x1 纹理像素组:
2x2 纹理像素组:
150
143(3)
152
154
1x4 纹理像素组:

图1E

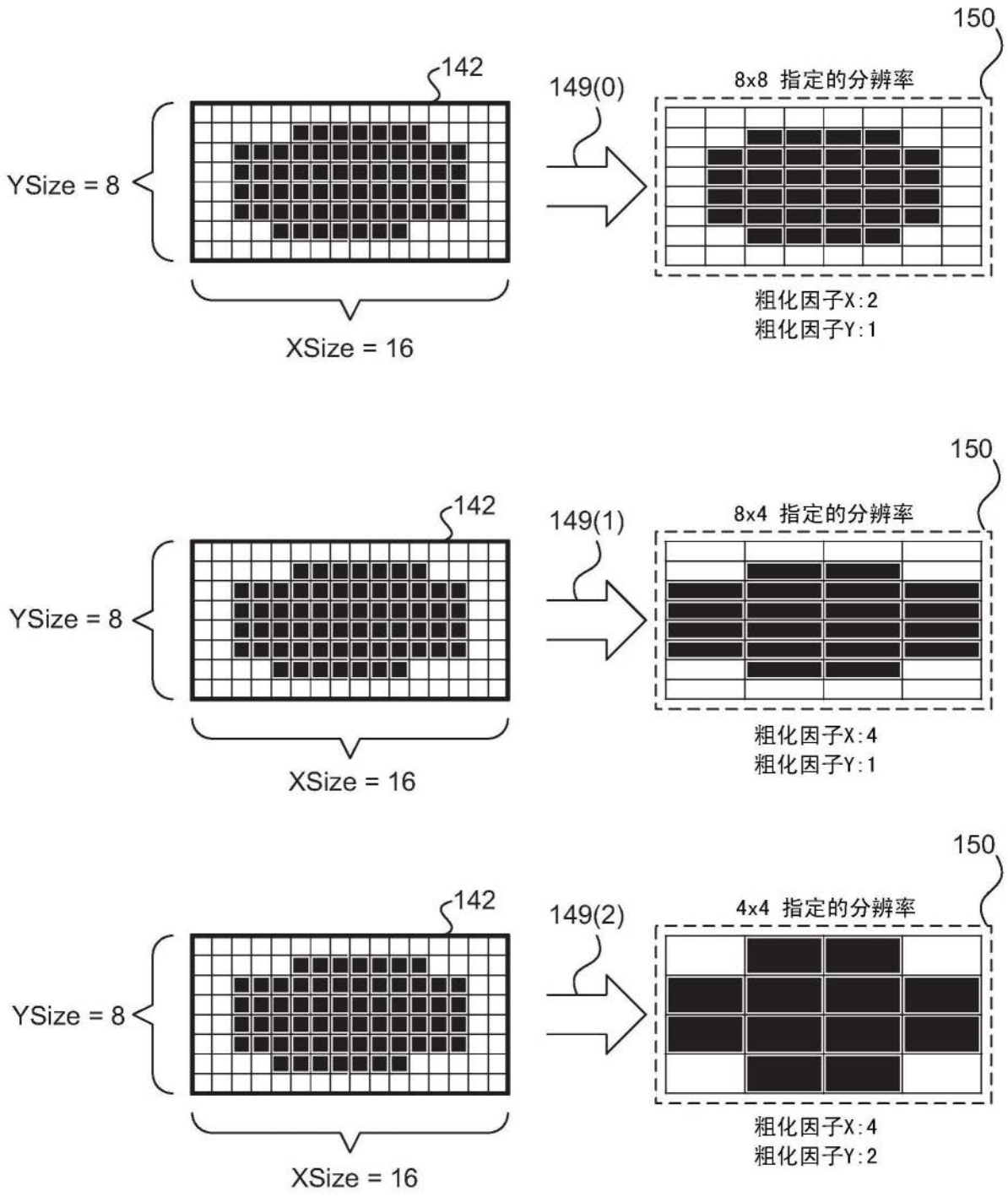

142
YSize = 8
XSize = 16
149(0)
150
8x8 指定的分辨率
粗化因子X:2
粗化因子Y:1
142
YSize = 8
XSize = 16
149(1)
150
8x4 指定的分辨率
粗化因子X:4
粗化因子Y:1
142
YSize = 8
XSize = 16
149(2)
150
4x4 指定的分辨率
粗化因子X:4
粗化因子Y:2

图1F

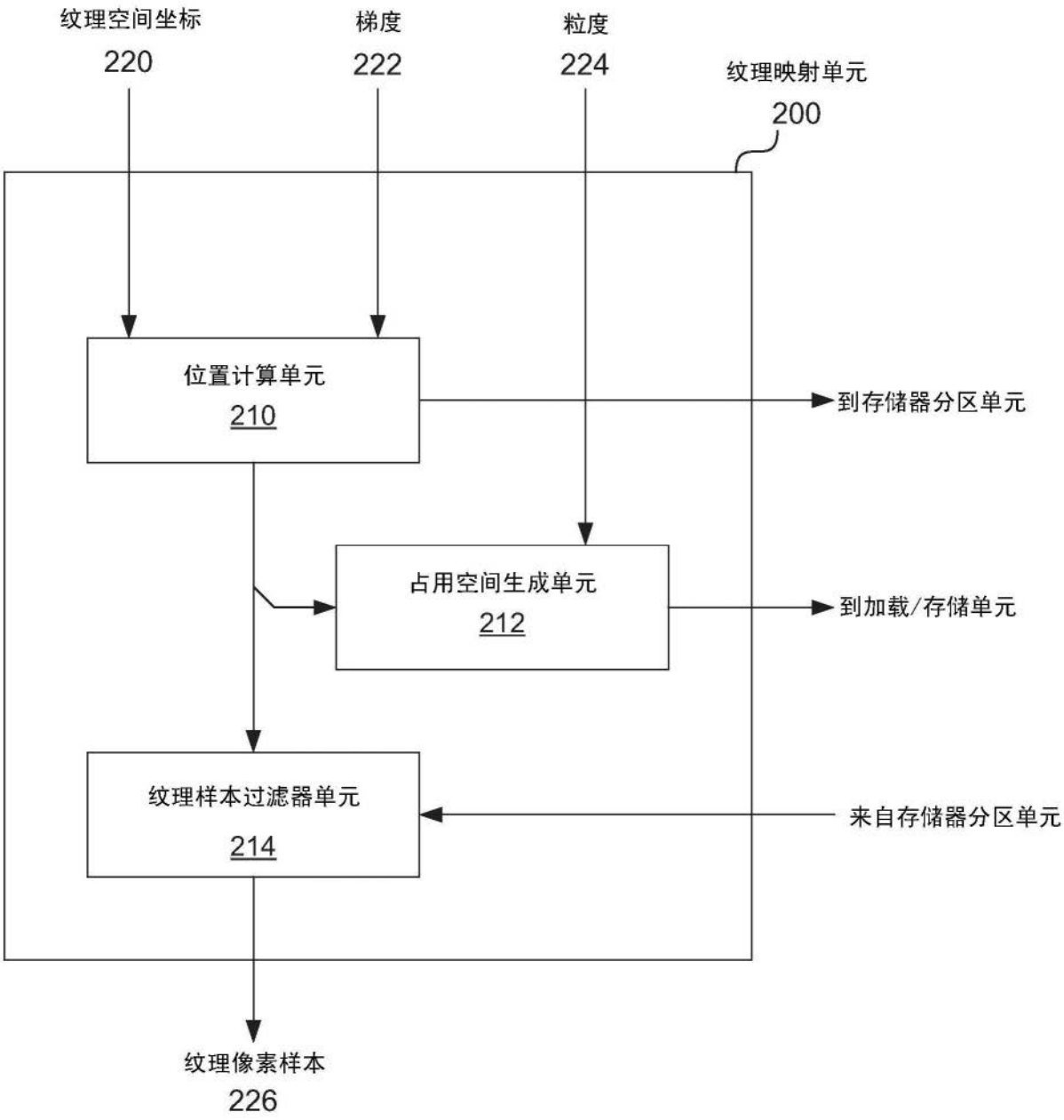

纹理空间坐标
220
梯度
222
粒度
224
纹理映射单元
200
位置计算单元
210
到存储器分区单元
占用空间生成单元
212
到加载/存储单元
纹理样本过滤器单元
214
来自存储器分区单元
纹理像素样本
226

图2

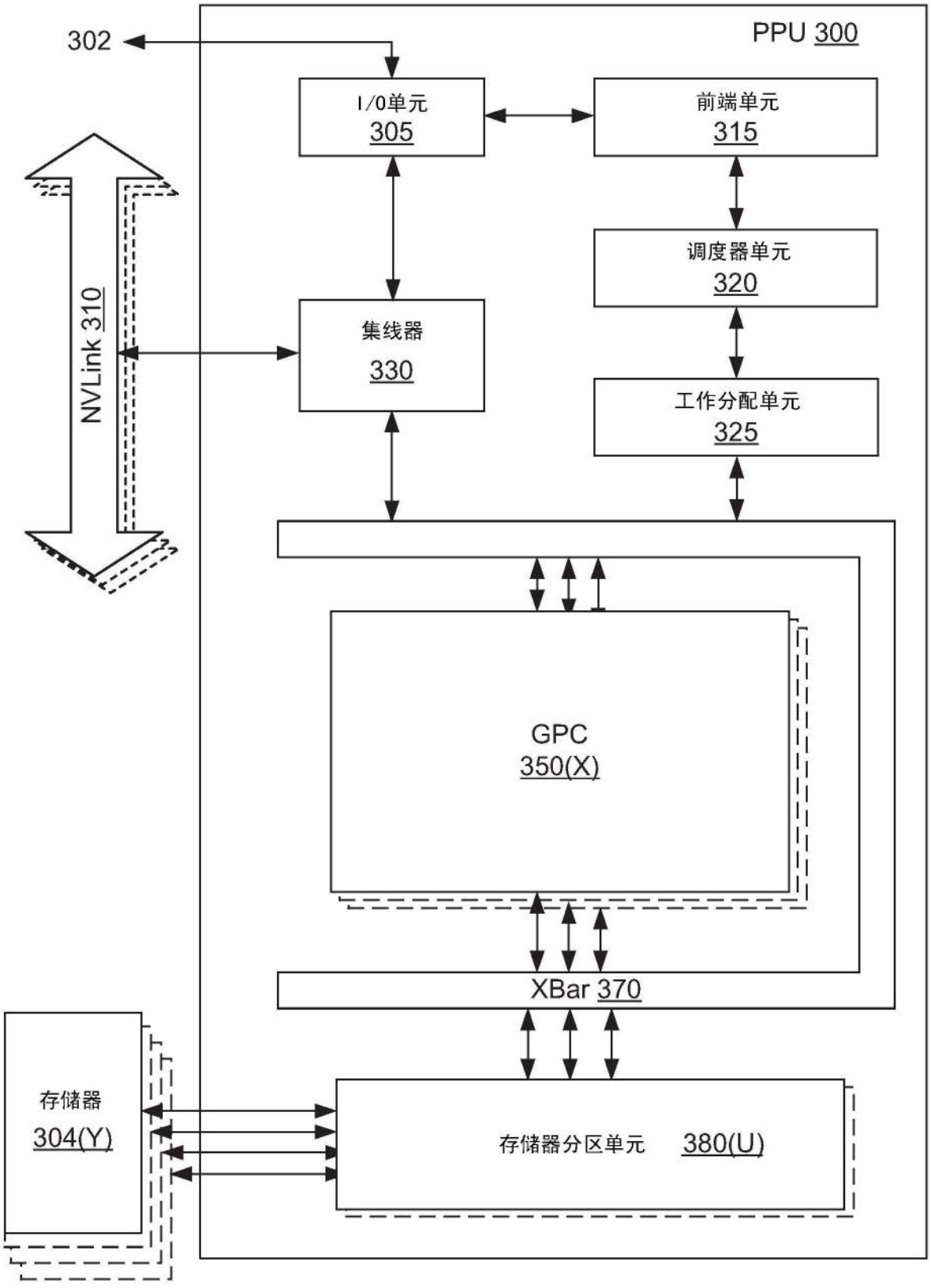

302
PPU 300
I/O单元
305
前端单元
315
调度器单元
320
集线器
330
工作分配单元
325
NVLink 310
GPC
350(X)
XBar 370
存储器
304(Y)
存储器分区单元 380(U)

图3

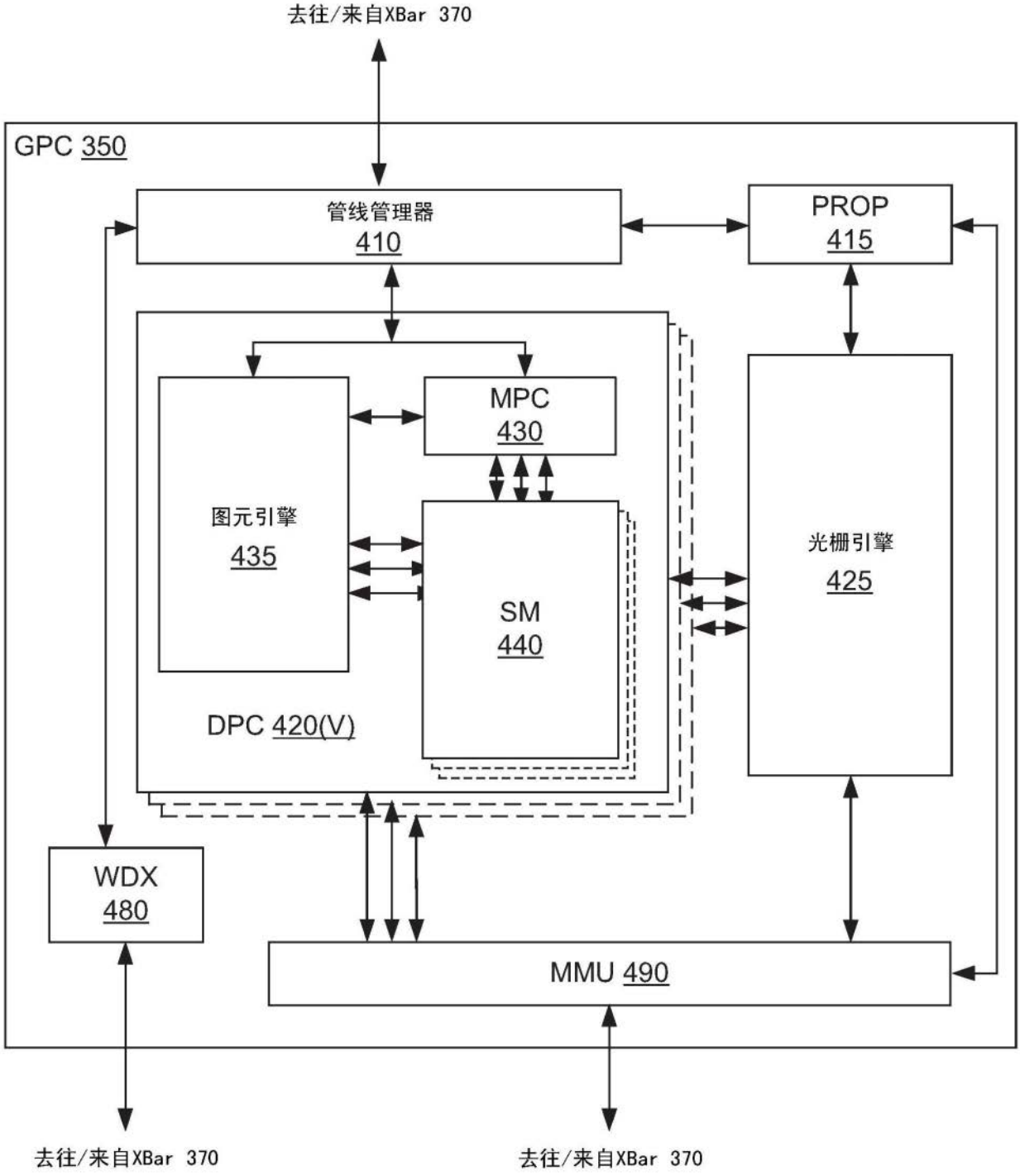

去往/来自XBar 370
GPC 350
管线管理器
410
PROP
415
MPC
430
图元引擎
435
SM
440
光栅引擎
425
DPC 420(V)
WDX
480
MMU 490
去往/来自XBar 370
去往/来自XBar 370

图4A

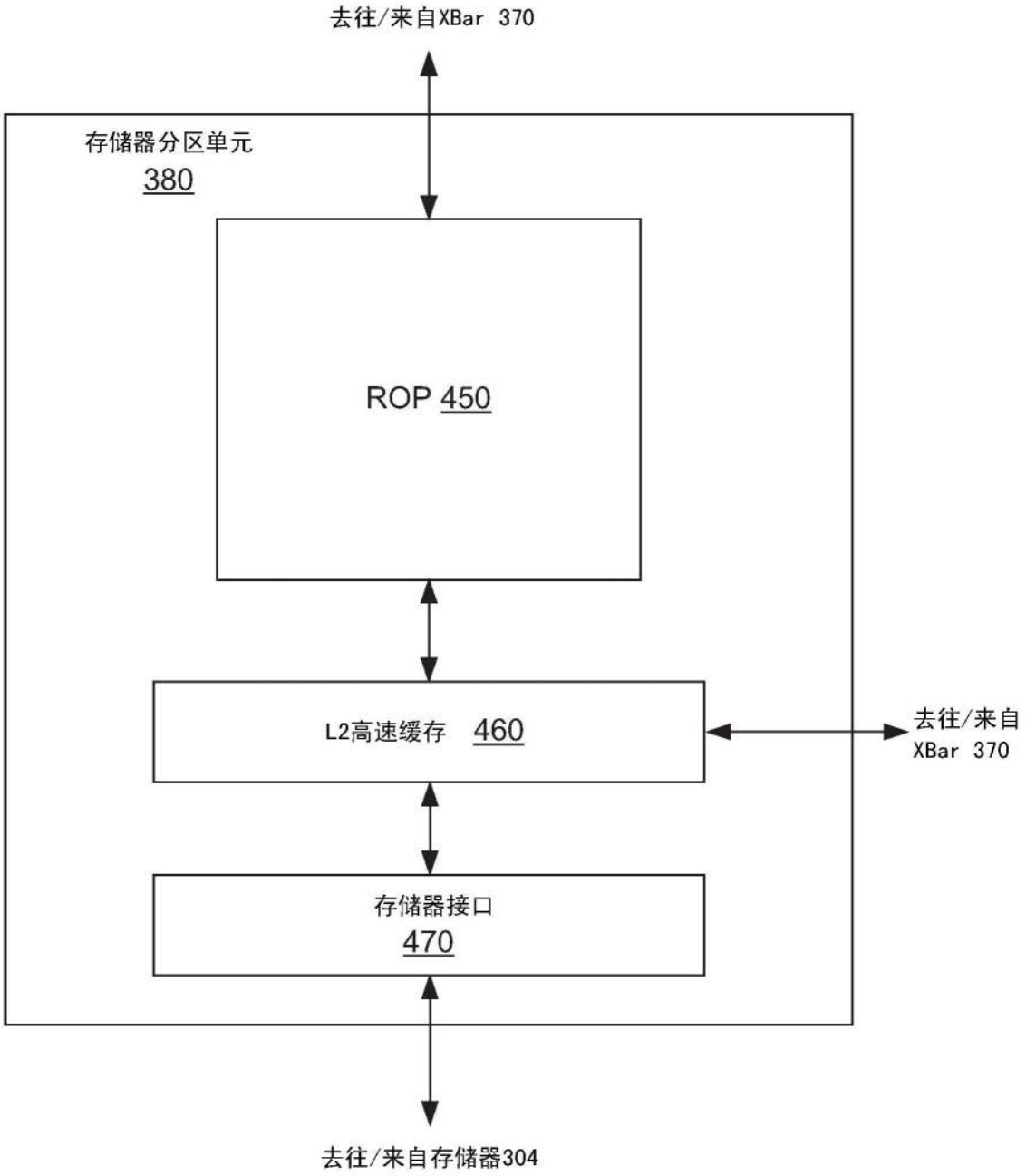

去往/来自XBar 370
存储器分区单元
380
ROP 450
L2高速缓存 460
去往/来自
XBar 370
存储器接口
470
去往/来自存储器304

图4B

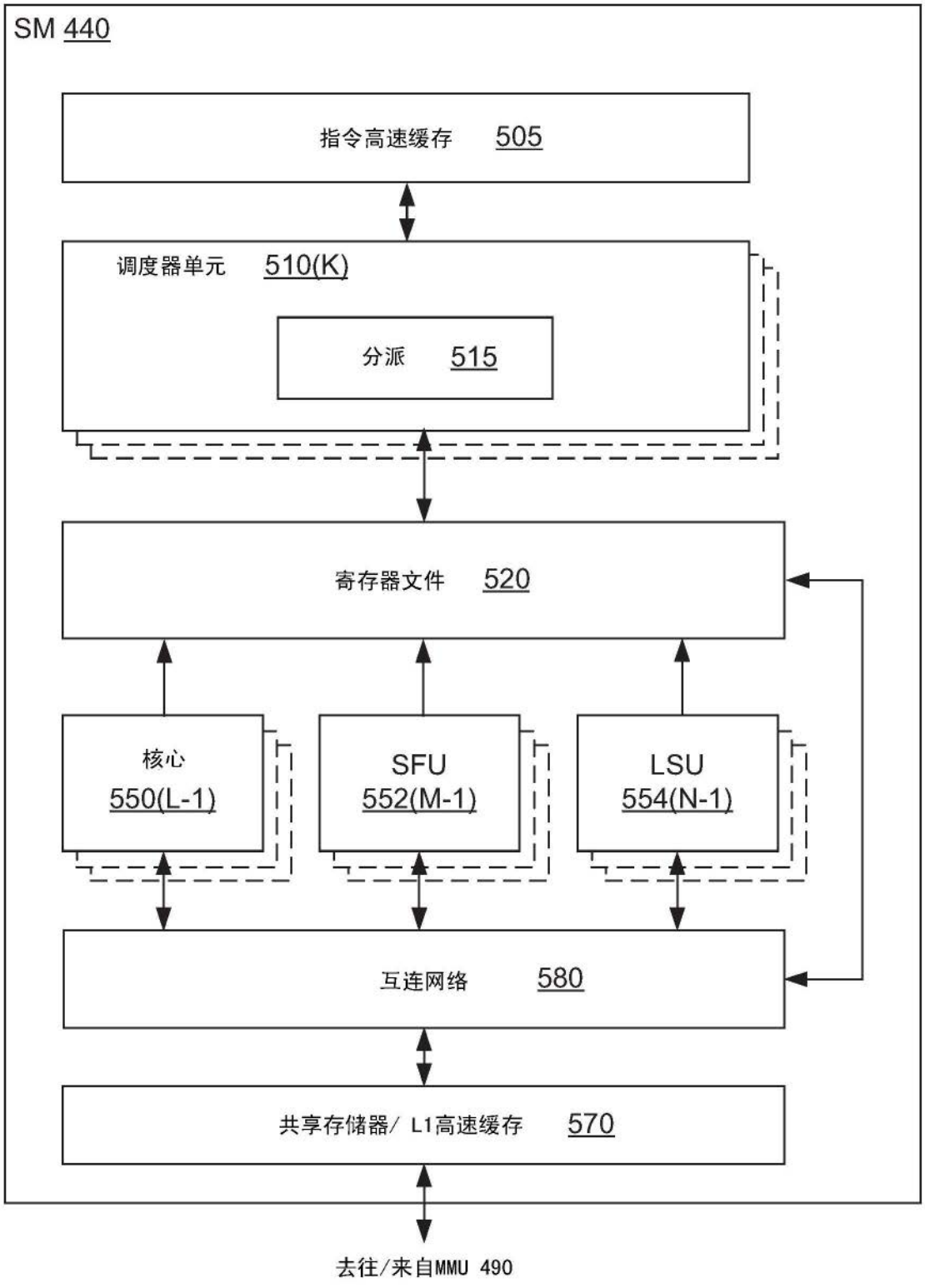

SM 440
指令高速缓存 505
调度器单元 510(K)
分派 515
寄存器文件 520
核心 550(L-1)
SFU 552(M-1)
LSU 554(N-1)
互连网络 580
共享存储器/ L1高速缓存 570
去往/来自MMU 490

图5A

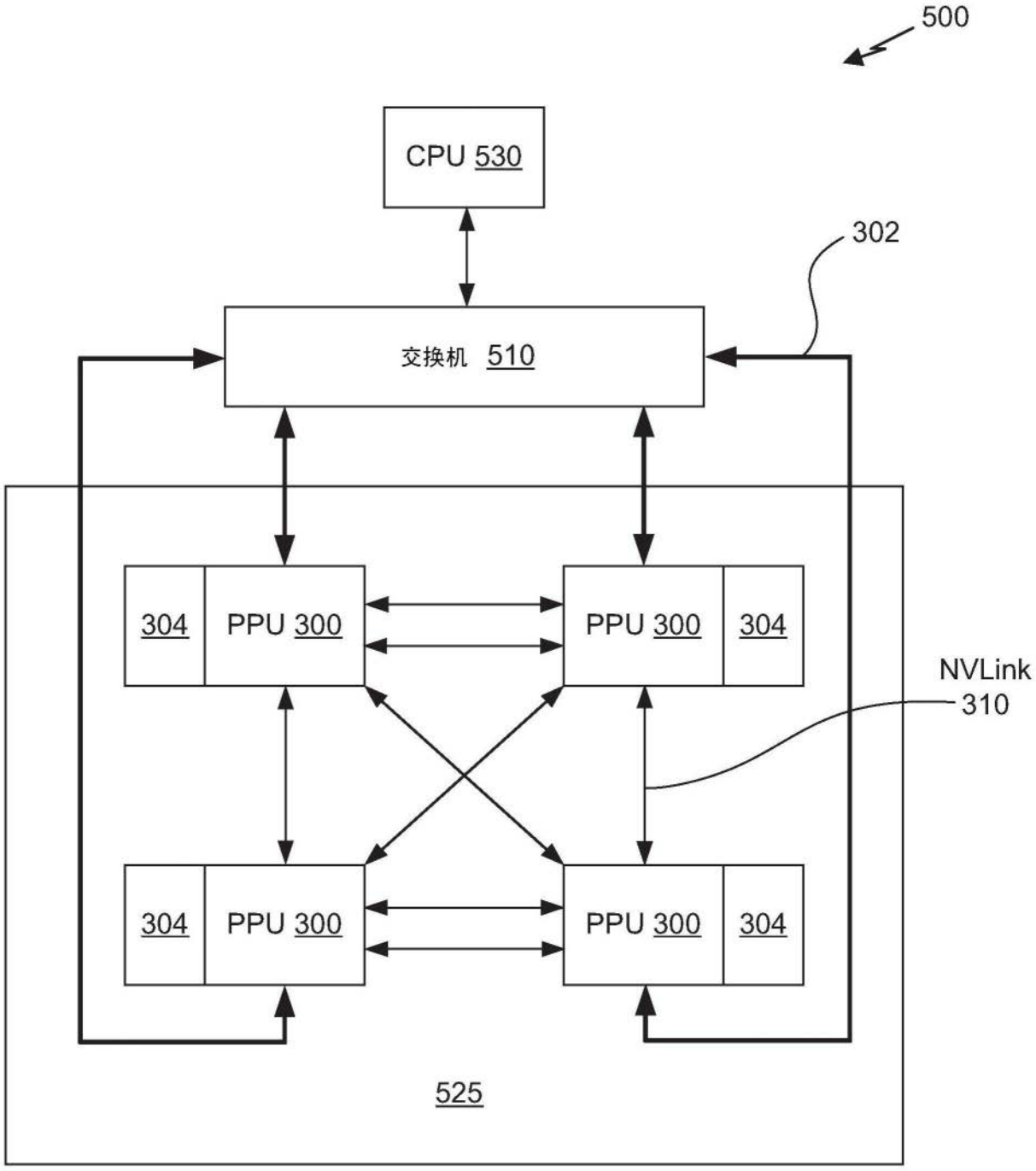

500
CPU 530
302
交换机 510
304
PPU 300
PPU 300
304
NVLink
310
304
PPU 300
PPU 300
304
525

图5B

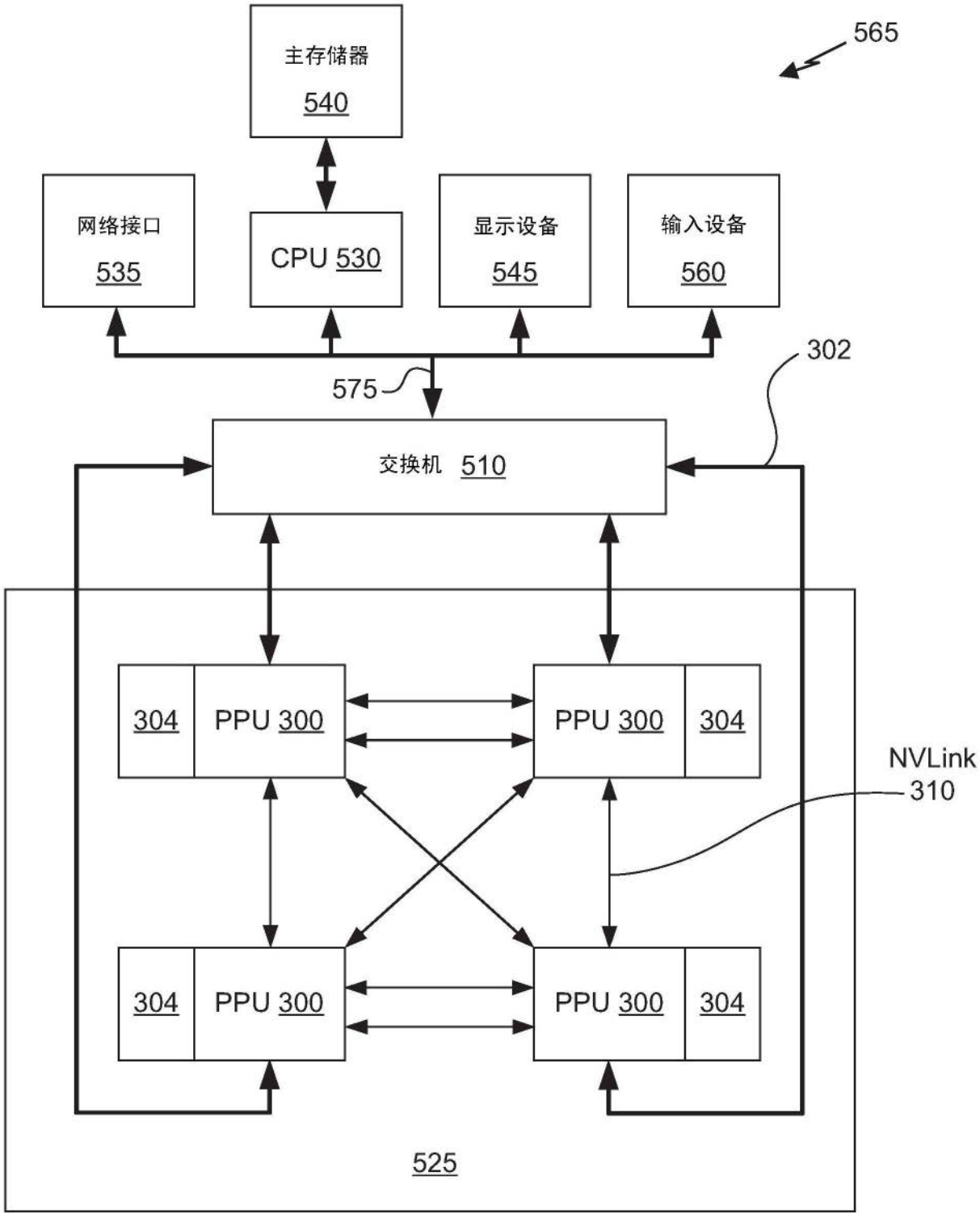

565
主存储器
540
网络接口
535
CPU 530
显示设备
545
输入设备
560
575
302
交换机 510
304
PPU 300
PPU 300
304
NVLink
310
304
PPU 300
PPU 300
304
525

图5C


600
输入数据
601
数据组装
610
顶点着色
620
图元组装
630
几何着色
640
视口SCC
650
光栅化
660
片段着色
670
光栅操作
680
输出数据
602

图6