(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104050705 A
(43)申请公布日 2014.09.17

(21)申请号 201310752242.1

(22)申请日 2013.12.31

(30)优先权数据
13/802,182 2013.03.13 US

(71)申请人 辉达公司
地址 美国加利福尼亚州

(72)发明人 埃里克·B·卢姆 鲁伊·巴斯托斯
杰尔姆·F·小杜鲁克
亨利·帕尔德·莫尔顿
尤里·Y·乌拉尔斯基

(74)专利代理机构 北京市磐华律师事务所 11336
代理人 董巍 谢栒

(51)Int.Cl.
G06T 15/00(2011.01)

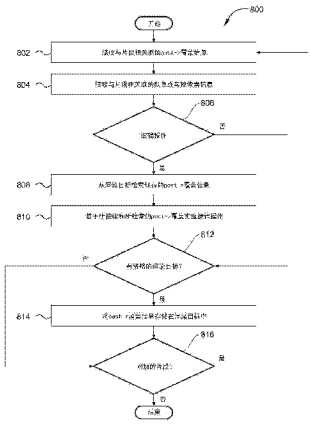
权利要求书2页 说明书16页 附图12页

(54)发明名称

处置光栅操作中的post-z覆盖数据

(57)摘要

提供了处置光栅操作中的post-z覆盖数据。公开了用于将post-z覆盖数据存储在渲染目标中的技术。颜色光栅操作(CROP)单元接收与多个样本的一部分相关联的覆盖掩码，其中图形基元相交包括多个样本的像素，并且该部分覆盖至少一个样本。CROP单元将覆盖掩码存储在渲染目标中的与像素相关联的位置处的数据字段中。所公开技术的一个优势在于，GPU仅计算用于如由post-z覆盖数据所确定的可见片段的颜色和其他像素信息。GPU不计算用于遮蔽片段的颜色和其他像素信息，从而减少总功耗并且改进总渲染性能。



CN 104050705 A

1. 一种子系统，包括：

光栅操作单元，其配置为通过实施以下步骤将覆盖信息存储在渲染目标中：

接收与第一图形基元的第一部分相关联的第一覆盖掩码，其中所述第一图形基元相交包括多个样本的像素，并且所述第一部分覆盖包括在所述多个样本中的至少一个样本；以及

将所述第一覆盖掩码存储在所述渲染目标中的与所述像素相关联的第一位置处的数据字段中。

2. 根据权利要求 1 所述的子系统，其中所述光栅操作单元进一步配置为实施包括检测所述渲染目标有资格存储覆盖信息的步骤。

3. 根据权利要求 1 所述的子系统，其中所述光栅操作单元进一步配置为实施以下步骤：

从所述第一位置检索第二覆盖掩码；以及

在存储所述第一覆盖掩码之前，通过基于所述第一覆盖掩码和所述第

二覆盖掩码实施逻辑操作来修改所述第一覆盖掩码。

4. 根据权利要求 1 所述的子系统，其中所述数据字段选自与所述渲染目标相关联的多个数据字段。

5. 根据权利要求 1 所述的子系统，其中所述第一覆盖掩码包括多个位，并且其中每个位与所述多个样本中的不同样本相对应。

6. 根据权利要求 1 所述的子系统，其中所述第一覆盖掩码指示哪些样本由所述第一图形基元的所述第一部分覆盖。

7. 根据权利要求 1 所述的子系统，其中所述光栅操作单元进一步配置为实施基于所述第一覆盖掩码计算所述像素的由所述第一图形基元的所述第一部分覆盖的百分比的步骤。

8. 根据权利要求 1 所述的子系统，其中所述光栅操作单元进一步配置为实施以下步骤：

接收与第二图形基元的第二部分相关联的第二覆盖掩码，其中所述第二基元相交所述像素，并且所述第二部分覆盖包括在所述多个样本中的至少一个样本；以及

将所述第二覆盖掩码存储在所述渲染目标中的与所述像素相关联的第二位置处的数据字段中；

其中所述第二覆盖掩码指示哪些样本由所述第二图形基元的所述第二部分覆盖并且不被所述第一图形基元的所述第一部分遮蔽。

9. 根据权利要求 8 所述的子系统，其中所述光栅操作单元进一步配置为实施基于所述第二覆盖掩码计算所述第二部分的质心的步骤。

10. 根据权利要求 8 所述的子系统，其中所述光栅操作单元进一步配置为实施以下步骤：

将与所述第一图形基元的所述第一部分相关联的第一属性存储在所述渲染目标中的与所述像素相关联的第三位置处的数据字段中；

将与所述第二图形基元的所述第二部分相关联的第二属性存储在所述渲染目标中的与所述像素相关联的第四位置处的数据字段中；

基于所述第一覆盖掩码和所述第二覆盖掩码中的至少一个对所述第一属性和所述第

二属性实施混合操作；以及

将所述混合操作的结果存储在所述渲染目标中的与所述像素相关联的第五位置处的数据字段中。

处置光栅操作中的 post-z 覆盖数据

技术领域

[0001] 本发明总地涉及三维(3D)图形处理，并且更具体地，涉及处置光栅操作中的 post-z 覆盖数据。

背景技术

[0002] 包括 2D 和 3D 图形对象的计算机生成的图像典型地使用具有一个或多个多级图形处理管线的图形处理单元(GPU)来渲染。这类图形管线包括各种可编程的和固定的功能级。可编程级包括各种处理单元，该处理单元执行着色器程序以渲染图形对象以及生成与图形对象相关联的各种视觉效果。固定的功能级实施不由可编程级所实施的附加的图形操作。一个这类固定功能单元是光栅操作单元，其实施诸如模板(stencil)、z 测试、混合等等的操作以及输出像素数据作为经处理的图形数据用于图形存储器中的存储。GPU 将图形对象渲染成一个或多个渲染目标。每个渲染目标包括画像元素(像素)，并且每个像素进而可以包括多个样本。一旦渲染完成，则一个或多个渲染目标中的像素可以传送到显示设备用于经渲染图像的视觉显示。

[0003] 在渲染期间，GPU 确定用于每个图形基元的覆盖信息，其中图形基元包括 3D 空间中的点、线和三角形。图形基元被再分成片段，其中每个片段与渲染目标中的特定像素的一个或多个样本相应。一旦 GPU 确定用于片段的覆盖信息，则 GPU 然后可以计算用于所覆盖样本每个片段的颜色或其他像素信息。该方法的一个缺点是，GPU 可能浪费计算用于在最后经渲染图像中不可见的片段的颜色和其他像素信息的计算工作量。与特定图形基元相关联的片段可由与离显示设备的屏幕表面较近的不同图形基元相关联的片段来覆盖。计算用于遮蔽(obscured)片段的颜色和其他像素信息减少可用于渲染最后图像中可见的图形对象的时间，因此降低总 GPU 性能。

[0004] 该问题的一个可能的解决方案是计算用于通过深度测试的片段的颜色和其他像素信息，其中深度测试基于离屏幕表面的距离来确定哪些片段是可见的以及哪些片段是不可见的。然而，在一些情况下，这类深度测试可以在图形处理管线中的晚级处实施。因此，深度测试结果在计算颜色和其他像素信息时是未知的。在其他情况下，典型地实施这类深度测试的单元可处于低功率状态以增加电池寿命。因此，深度测试的结果在 GPU 渲染片段时可能是不可用的。

[0005] 如前述所示，本领域需要的是降低针对遮蔽片段所实施的处理量的、用于渲染图形处理管线中的片段数据的技术。

发明内容

[0006] 本发明的一个实施例阐述用于将 post-z 覆盖数据存储在渲染目标中的方法。方法包括接收与图形基元的一部分相关联的覆盖掩码(mask)，其中图形基元相交包括多个样本的像素，并且该部分覆盖至少一个样本。方法进一步包括将覆盖掩码存储在渲染目标中的与像素相关联的位置处的数据字段中。

[0007] 其他实施例包括但不限于，包括指令的计算机可读介质，指令使处理单元能够实现所公开方法的一个或多个方面。其他实施例包括但不限于，包括配置为实现所公开方法的一个或多个方面的处理单元以及配置为实现所公开方法的一个或多个方面的系统的子系统。

[0008] 所公开技术的一个优势在于，GPU 仅计算用于如由 post-z 覆盖数据所确定的可见片段的颜色和其他像素信息。GPU 不计算用于遮蔽片段的颜色和其他像素信息，从而减少总功耗并且改进总渲染性能。

附图说明

[0009] 因此，可以详细地理解本发明的上述特征，并且可以参考示范性实施例得到对如上面所简要概括的本发明更具体的描述，其中一些实施例在附图中示出。然而，应当注意的是，附图仅示出了本发明的典型实施例，因此不应被认为是对其范围的限制，本发明可以具有其他等效的实施例。

[0010] 图 1 是示出了配置为实现本发明的一个或多个方面的计算机系统的框图；

[0011] 图 2 是根据本发明的一个实施例的、用于图 1 的计算机系统的并行处理子系统的框图；

[0012] 图 3A 是根据本发明的一个实施例的、图 2 的 PPU 中的一个内的分区单元的框图；

[0013] 图 3B 是根据本发明的一个实施例的、图 2 的通用处理集群（GPC）内的流多处理器（SM）的一部分的框图；

[0014] 图 4 是根据本发明的一个实施例的、图 2 的并行处理单元中的一个或多个可配置为实现其的图形处理管线的示意图。

[0015] 图 5 示出了根据本发明的一个实施例的、图 4 的片段处理单元和光栅操作单元。

[0016] 图 6 示出了根据本发明的一个实施例的、如存储在图 2 的分区单元中的一个或多个中的渲染目标的集合。

[0017] 图 7A-7D 示出了根据本发明的一个实施例的、相交多个图形基元的像素；以及

[0018] 图 8 阐述了根据本发明的一个实施例的、用于存储 post-z 覆盖数据的方法步骤的流程图。

具体实施方式

[0019] 在下面的描述中，将阐述大量的具体细节以提供对本发明更透彻的理解。然而，本领域的技术人员应该清楚，本发明可以在没有一个或多个这些具体细节的情况下得以实践。

[0020] 系统概述

[0021] 图 1 是示出了配置为实现本发明的一个或多个方面的计算机系统 100 的框图。计算机系统 100 包括经由可以包括存储器桥 105 的互连路径通信的中央处理单元（CPU）102 和系统存储器 104。存储器桥 105 可以是例如北桥芯片，经由总线或其他通信路径 106（例如超传输（Hyper Transport）链路）连接到 I/O（输入 / 输出）桥 107。I/O 桥 107，其可以是例如南桥芯片，从一个或多个用户输入设备 108（例如键盘、鼠标）接收用户输入并且经由通信路径 106 和存储器桥 105 将该输入转发到 CPU102。并行处理子系统 112 经由总线或第

二通信路径 113（例如外围部件互连（PCI）Express、加速图形端口或超传输链路）耦连到存储器桥 105；在一个实施例中，并行处理子系统 112 是将像素传递到显示设备 110 的图形子系统，该显示设备 110 可以是任何常规的基于阴极射线管、液晶显示器、发光二极管显示器等等。系统盘 114 也连接到 I/O 桥 107 并可配置为存储内容应用和数据用于由 CPU102 和并行处理子系统 112 使用。系统盘 114 为应用和数据提供非易失性存储并且可以包括固定的或可移动的硬盘驱动器、闪存设备以及 CD-ROM（压缩光盘只读存储器）、DVD-ROM（数字多用光盘 -ROM）、蓝光、HD-DVD（高清晰度 DVD）或其他磁性、光学或固态存储设备。

[0022] 交换器 116 提供 I/O 桥 107 与诸如网络适配器 118 以及各种插卡 120 和 121 的其他部件之间的连接。其他部件（未明确示出），包括通用串行总线（USB）或其他端口连接、压缩光盘（CD）驱动器、数字多用光盘（DVD）驱动器、胶片录制设备及类似部件，也可以连接到 I/O 桥 107。图 1 所示的各种通信路径包括具体命名的通信路径 106 和 113 可以使用任何适合的协议实现，诸如 PCI-Express、AGP（加速图形端口）、超传输或者任何其他总线或点到点通信协议，并且如本领域已知的，不同设备间的连接可使用不同协议。

[0023] 在一个实施例中，并行处理子系统 112 包含经优化用于图形和视频处理的电路，包括例如视频输出电路，并且构成图形处理单元（GPU）。在另一个实施例中，并行处理子系统 112 包含经优化用于通用处理的电路，同时保留底层（underlying）的计算架构，本文将更详细地进行描述。在又一个实施例中，可以将并行处理子系统 112 与一个或多个其他系统元件集成在单个子系统中，诸如结合存储器桥 105、CPU102 以及 I/O 桥 107，以形成片上系统（SoC）。

[0024] 应该理解，本文所示系统是示例性的，并且变化和修改都是可能的。连接拓扑，包括桥的数目和布置、CPU102 的数目以及并行处理子系统 112 的数目，可根据需要修改。例如，在一些实施例中，系统存储器 104 直接连接到 CPU102 而不是通过桥，并且其他设备经由存储器桥 105 和 CPU102 与系统存储器 104 通信。在其他替代性拓扑中，并行处理子系统 112 连接到 I/O 桥 107 或直接连接到 CPU102，而不是连接到存储器桥 105。而在其他实施例中，I/O 桥 107 和存储器桥 105 可能被集成到单个芯片上而不是作为一个或多个分立设备存在。大型实施例可以包括两个或更多个 CPU102 以及两个或更多个并行处理子系统 112。本文所示的特定部件是可选的；例如，任何数目的插卡或外围设备都可能得到支持。在一些实施例中，交换器 116 被去掉，网络适配器 118 和插卡 120、121 直接连接到 I/O 桥 107。

[0025] 图 2 示出了根据本发明的一个实施例的并行处理子系统 112。如所示的，并行处理子系统 112 包括一个或多个并行处理单元（PPU）202，每个并行处理单元 202 都耦连到本地并行处理（PP）存储器 204。通常，并行处理子系统包括 U 个 PPU，其中 U ≥ 1。（本文中，类似对象的多个实例需要时以标识对象的参考数字和标识实例的括号中的数字来表示。）PPU202 和并行处理存储器 204 可使用一个或多个集成电路设备来实现，诸如可编程处理器、专用集成电路（ASIC）或存储器设备，或者以任何其他技术可行的方式来实现。

[0026] 再参考图 1 以及图 2，在一些实施例中，并行处理子系统 112 中的一些或所有 PPU202 是具有渲染管线的图形处理器，其可以配置为实施与下述相关的各种操作：经由存储器桥 105 和第二通信路径 113 从 CPU102 和 / 或系统存储器 104 所供应的图形数据生成像素数据，与本地并行处理存储器 204（可被用作图形存储器，包括例如常规帧缓冲区（buffer））交互以存储和更新像素数据，传递像素数据到显示设备 110 等等。在一些实施例

中，并行处理子系统 112 可包括一个或多个作为图形处理器而操作的 PPU202 以及一个或多个用于通用计算的其他 PPU202。这些 PPU 可以是同样的或不同的，并且每个 PPU 可具有专用并行处理存储器设备或不具有专用并行处理存储器设备。并行处理子系统 112 中的一个或多个 PPU202 可输出数据到显示设备 110，或者并行处理子系统 112 中的每个 PPU202 可输出数据到一个或多个显示设备 110。

[0027] 在操作中，CPU102 是计算机系统 100 的主处理器，控制和协调其他系统部件的操作。具体地，CPU102 发出控制 PPU202 的操作的命令。在一些实施例中，CPU102 写入用于每个 PPU202 的命令流到数据结构中（在图 1 或图 2 中未明确示出），该数据结构可位于系统存储器 104、并行处理存储器 204、或 CPU102 和 PPU202 都可访问的其他存储位置中。将指向每个数据结构的指针写到入栈缓冲区（pushbuffer）以发起对数据结构中的命令流的处理。PPU202 从一个或多个入栈缓冲区读取命令流，然后相对于 CPU102 的操作异步地执行命令。可以经由设备驱动程序 103 由应用程序为每个入栈缓冲区指定执行优先级以控制对不同入栈缓冲区的调度。

[0028] 现在返回参考图 2 和图 1，每个 PPU202 包括经由连接到存储器桥 105（或者，在一个替代性实施例中，直接连接到 CPU102）的通信路径 113 与计算机系统 100 的其余部分通信的 I/O（输入 / 输出）单元 205。PPU202 到计算机系统 100 的其余部分的连接也可以变化。在一些实施例中，并行处理子系统 112 可实现为可插入到计算机系统 100 的扩展槽中的插卡。在其他实施例中，PPU202 可以和诸如存储器桥 105 或 I/O 桥 107 的总线桥集成在单个芯片上。而在其他实施例中，PPU202 的一些或所有元件可以和 CPU102 集成在单个芯片上。

[0029] 在一个实施例中，通信路径 113 是 PCI Express 链路，如本领域所知的，其中专用通道被分配到每个 PPU202。也可以使用其他通信路径。I/O 单元 205 生成用于在通信路径 113 上传送的包（或其他信号），并且还从通信路径 113 接收所有传入的包（或其他信号），将传入的包引导到 PPU202 的适当部件。例如，可将与处理任务相关的命令引导到主机接口 206，而将与存储器操作相关的命令（例如，对并行处理存储器 204 的读取或写入）引导到存储器交叉开关单元 210。主机接口 206 读取每个入栈缓冲区，并且将存储在入栈缓冲区中的命令流输出到前端 212。

[0030] 有利地，每个 PPU202 都实现高度并行处理架构。如详细示出的，PPU202（0）包括处理集群阵列 230，该阵列 230 包括 C 个通用处理集群（GPC）208，其中 $C \geq 1$。每个 GPC208 能够并发执行大量的（例如，几百或几千）线程，其中每个线程是程序的实例（instance）。在各种应用中，可分配不同的 GPC208 用于处理不同类型的程序或用于实施不同类型的计算。GPC208 的分配可以取决于因每种类型的程序或计算所产生的工作量而变化。

[0031] GPC208 从任务 / 工作单元 207 内的工作分布单元接收所要执行的处理任务。工作分布单元接收指向编码为任务元数据（TMD）并存储在存储器中的处理任务的指针。指向 TMD 的指针包括在存储为入栈缓冲区并由前端单元 212 从主机接口 206 接收的命令流中。可以编码为 TMD 的处理任务包括所要处理的数据的索引，以及定义数据将被如何处理（例如，什么程序将被执行）的状态参数和命令。任务 / 工作单元 207 从前端 212 接收任务并确保在每一个 TMD 所指定的处理发起前，将 GPC208 配置为有效状态。可以为每个 TMD 指定用来调度处理任务的执行的优先级。还可从处理集群阵列 230 接收处理任务。可选地，TMD 可包括控制将 TMD 添加到处理任务列表（或指向处理任务的指针的列表）的头部还是尾部的参

数，从而提供除优先级以外的另一级别的控制。

[0032] 存储器接口 214 包括 D 个分区单元 215，每个分区单元 215 直接耦连到并行处理存储器 204 的一部分，其中 D ≥ 1。如所示的，分区单元 215 的数目一般等于动态随机存取存储器（DRAM）220 的数目。在其他实施例中，分区单元 215 的数目也可以不等于存储器设备的数目。本领域的普通技术人员应该理解 DRAM220 可以用其他合适的存储设备来替代并且可以是一般常规的设计。因此省略了详细描述。诸如帧缓冲区或纹理映射图的渲染目标可以跨 DRAM220 加以存储，这允许分区单元 215 并行写入每个渲染目标的各部分以有效地使用并行处理存储器 204 的可用带宽。

[0033] 任何一个 GPC208 都可以处理要被写到并行处理存储器 204 内的任何 DRAM220 的数据。交叉开关单元 210 配置为路由每个 GPC208 的输出到任何分区单元 215 的输入或到另一个 GPC208 用于进一步处理。GPC208 通过交叉开关单元 210 与存储器接口 214 通信，以对各种外部存储器设备进行读取或写入。在一个实施例中，交叉开关单元 210 具有到存储器接口 214 的连接以和 I/O 单元 205 通信，以及到本地并行处理存储器 204 的连接，从而使得在不同 GPC208 内的处理内核能够与系统存储器 104 或对于 PPU202 而言非本地的其他存储器通信。在图 2 所示的实施例中，交叉开关单元 210 直接与 I/O 单元 205 连接。交叉开关单元 210 可使用虚拟信道来分开 GPC208 与分区单元 215 之间的业务流。

[0034] 另外，GPC208 可被编程以执行与种类繁多的应用相关的处理任务，包括但不限于，线性和非线性数据变换、视频和 / 或音频数据过滤、建模操作（例如，应用物理定律以确定对象的位置、速率和其他属性）、图像渲染操作（例如，曲面细分着色器、顶点着色器、几何着色器、和 / 或像素着色器程序）等等。PPU202 可将数据从系统存储器 104 和 / 或本地并行处理存储器 204 转移到内部（片上）存储器中，处理该数据，并且将结果数据写回到系统存储器 104 和 / 或本地并行处理存储器 204，其中这样的数据可以由其他系统部件访问，所述其他系统部件包括 CPU102 或另一个并行处理子系统 112。

[0035] PPU202 可配备有任何容量（amount）的本地并行处理存储器 204，包括没有本地存储器，并且可以以任何组合方式使用本地存储器和系统存储器。例如，在统一存储器架构（UMA）实施例中，PPU202 可以是图形处理器。在这样的实施例中，将不提供或几乎不提供专用的图形（并行处理）存储器，并且 PPU202 会以排他或几乎排他的方式使用系统存储器。在 UMA 实施例中，PPU202 可集成到桥式芯片中或处理器芯片中，或作为具有高速链路（例如，PCI Express）的分立芯片提供，所述高速链路经由桥式芯片或其他通信手段将 PPU202 连接到系统存储器。

[0036] 如上所示，在并行处理子系统 112 中可以包括任何数目的 PPU202。例如，可在单个插卡上提供多个 PPU202、或可将多个插卡连接到通信路径 113、或可将一个或多个 PPU202 集成到桥式芯片中。在多 PPU 系统中的 PPU202 可以彼此同样或不同。例如，不同的 PPU202 可能具有不同数目的处理内核、不同容量的本地并行处理存储器等等。在存在多个 PPU202 的情况下，可并行操作那些 PPU 从而以高于单个 PPU202 所可能达到的吞吐量来处理数据。包含一个或多个 PPU202 的系统可以以各种配置和形式因素来实现，包括台式电脑、笔记本电脑或手持式个人计算机、服务器、工作站、游戏控制台、嵌入式系统等等。

[0037] 图 3A 是根据本发明的一个实施例的、图 2 的 PPU 中的一个内的分区单元 215 的框图。如所示，分区单元 215 包括 L2 高速缓存 350、帧缓冲区（FB）DRAM 接口 355 以及光栅操

作单元(ROP)360。L2 高速缓存 350 是配置为实施从交叉开关单元 210 和 ROP360 所接收的加载和存储操作。读未命中和紧急回写请求由 L2 高速缓存 350 输出到 FB DRAM 接口 355 用于处理。脏(dirty)更新也发送到 FB355 用于伺机性(opportunistic)处理。FB355 直接与 DRAM220 配合,输出读和写请求以及接收从 DRAM220 所读取的数据。

[0038] 在图形应用中,ROP360 是实施光栅操作以及输出像素数据作为经处理的图形数据用于图形存储器中的存储的处理单元,所述光栅操作诸如模板、z 测试、混合等等。在本发明的实施例中,ROP 包括在每个 GPC208 而不是分区单元 215 内,并且像素读和写请求在交叉开关单元 210 而不是像素片段数据之上进行传送。

[0039] 经处理的图形数据可以显示在显示设备 110 上或者被路由用于由 CPU102 或由并行处理子系统 112 内的处理实体中的一个进一步处理。每个分区单元 215 包括 ROP360 以分布光栅操作的处理。在一些实施例中,ROP360 可配置为压缩被写到存储器的 z 或颜色数据以及解压缩从存储器所读取的 z 或颜色数据。

[0040] 图 3B 为根据本发明的一个实施例的、图 2 的通用处理集群(GPC)208 内的流多处理器(SM)310 的一部分的框图。每个 GPC208 可配置为并行执行大量线程,其中术语“线程”是指在特定输入数据集上执行的特定程序的实例。在一些实施例中,单指令、多数据(SIMD)指令发出技术用于在不提供多个独立指令单元的情况下支持大量线程的并行执行。在其他实施例中,单指令、多线程(SIMT)技术用于使用配置为向 GPC208 中的每一个内的处理引擎集发出指令的公共指令单元来支持大量一般来说同步的线程的并行执行。不同于所有处理引擎通常都执行同样指令的 SIMD 执行机制,SIMT 执行通过给定线程程序允许不同线程更容易跟随分散执行路径。本领域普通技术人员应该理解 SIMD 处理机制代表 SIMT 处理机制的功能子集。

[0041] 经由将处理任务分布到一个或多个流多处理器(SM)310 的管线管理器(未示出)来有利地控制 GPC208 的操作。其中每个 SM310 配置为处理一个或多个线程组。每个 SM310 包括配置为经由 GPC208 内的 L1.5 高速缓存(未示出)从存储器接收指令和常量的 L1 高速缓存 370。线程束(warp)调度器和指令单元 312 从指令 L1 高速缓存 370 接收指令和常量并且根据指令和常量来控制本地寄存器堆 304 和 SM310 功能单元。SM310 功能单元包括 N 个 exec (执行或处理) 单元 302 和 P 个加载 - 存储单元(LSU)303。SM 功能单元可以是管线化的,其允许在前一个指令完成之前发出新指令。可提供功能执行单元的任何组合。在一个实施例中,功能单元支持各种各样的操作,包括整数和浮点运算(例如加法和乘法)、比较操作、布尔操作(AND、OR、XOR)、移位和各种代数函数的计算(例如平面插值、三角函数、指数函数和对数函数等等);以及相同功能单元硬件可均衡地用来实施不同的操作。

[0042] 如本文之前所定义的,传送到特定 GPC208 的一系列指令构成线程,并且跨 SM310 内的并行处理引擎(未示出)的某一数目的并发执行线程的集合在本文中称为“线程束”或“线程组”。如本文所使用的,“线程组”是指对不同输入数据并发执行相同程序的一组线程,所述组的一个线程被指派到 SM310 内的不同处理引擎。线程组可以包括比 SM310 内的处理引擎数目少的线程,在这种情况下一些处理引擎将在该线程组正在被处理的周期期间处于闲置状态。线程组还可以包括比 SM310 内的处理引擎数目多的线程,在这种情况下处理将在连续的时钟周期内发生。因为每个 SM310 可以并发支持多达 G 个线程组,结果是包括 M 个流多处理器 310 的 GPC208 中的系统在任何给定时间在 GPC208 中可以执行多达 G*M 个线

程组。

[0043] 此外，多个相关线程组可以在 SM310 内同时活动(在执行的不同阶段)。该线程组集合在本文中称为“协作线程阵列”(“CTA”)或“线程阵列”。特定 CTA 的大小等于 m*k，其中 k 是线程组中并发执行线程的数目并且通常是 SM310 内的并行处理引擎数目的整数倍，以及 m 是 SM310 内同时活动的线程组的数目。CTA 的大小一般由编程者以及可用于 CTA 的硬件资源诸如存储器或寄存器的容量来确定。

[0044] 在本发明的实施例中，使用计算系统的 PPU202 或其他处理器来使用线程阵列执行通用计算是可取的。为线程阵列中的每个线程指派在线程的执行期间对于线程可访问的唯一的线程标识符(“线程 ID”)。可被定义为一维或多维数值的线程 ID 控制线程处理行为的各方面。例如，线程 ID 可用于确定线程将要处理输入数据集的哪部分和 / 或确定线程将要产生或写输出数据集的哪部分。

[0045] 每线程指令序列可包括定义线程阵列的代表性线程和一个或多个其他线程之间的协作行为的至少一个指令。例如，每线程指令序列可能包括在序列中的特定点处挂起用于代表性线程的操作执行直到诸如其他线程的一个或多个到达该特定点的时间为止的指令、用于代表性线程将数据存储在其他线程的一个或多个有权访问的共享存储器中的指令、用于代表性线程原子地读取和更新存储在其他线程的一个或多个基于它们的线程 ID 有权访问的共享存储器中的数据的指令等等。CTA 程序还可以包括计算数据将从其读取的共享存储器中的地址的指令，该地址是线程 ID 的函数。通过定义合适的函数并提供同步技术，可以以可预测的方式由 CTA 的一个线程将数据写入共享存储器中的给定位置并由同一个 CTA 的不同线程从该位置读取数据。因此，数据在线程之间共享的任何期望模式可以得到支持，以及 CTA 中的任何线程可以与同一个 CTA 中的任何其他线程共享数据。如果存在数据在 CTA 的线程之间的共享，则其范围由 CTA 程序确定；因此，应该理解的是，在使用 CTA 的特定应用中，CTA 的线程可能会或可能不会真正互相共享数据，这取决于 CTA 程序，术语“CTA”和“线程阵列”在本文作为同义词使用。

[0046] SM310 提供具有不同级别的可访问性的片上(内部)数据存储。特殊寄存器(未示出)对于 LSU303 可读但不可写并且用于存储定义每个线程的“位置”的参数。在一个实施例中，特殊寄存器包括每线程(或 SM310 内的每 exec 单元 302)一个的存储线程 ID 的寄存器；每个线程 ID 寄存器仅由各自的 exec 单元 302 可访问。特殊寄存器还可以包括附加寄存器，其对于执行由任务元数据(TMD)(未示出)所代表的同一个处理任务的所有线程(或由所有 LSU303)可读，其存储 CTA 标识符、CTA 维数、CTA 所属网格(grid)的维数(或队列位置，如果 TMD 编码队列任务而不是网格任务的话)、以及 CTA 被指派到的 TMD 的标识符。

[0047] 如果 TMD 是网格 TMD，则 TMD 的执行会启动和执行固定数目的 CTA 以处理存储在队列 525 中的固定量的数据。将 CTA 的数目指定为网格宽度、高度和深度的乘积。可以将固定量的数据存储在 TMD 中或 TMD 可以存储指向将由 CTA 所处理的数据的指针。TMD 还存储由 CTA 所执行的程序的开始地址。

[0048] 如果 TMD 是队列 TMD，那么使用 TMD 的队列特点，这意味着将要被处理的数据量不一定是固定的。队列条目存储用于由指派到 TMD 的 CTA 所处理的数据。队列条目还可以代表在线程执行期间由另一个 TMD 所生成的子任务，从而提供嵌套并行性。通常线程或包括线程的 CTA 的执行被挂起直到子任务的执行完成。可以将队列存储在 TMD 中或与 TMD 分开

存储，在该情况下 TMD 存储指向该队列的队列指针。有利地，当代表子任务的 TMD 正在执行时可以将由子任务所生成的数据写到队列。队列可以实现为循环队列以使得数据的总量不限于队列的大小。

[0049] 属于网格的 CTA 具有指示网格内各自 CTA 的位置的隐含网格宽度、高度和深度参数。在初始化期间响应于经由前端 212 从设备驱动程序 103 所接收的命令来写特殊寄存器并且在处理任务的执行期间特殊寄存器不改变。前端 212 调度每个处理任务用于执行。每个 CTA 与具体 TMD 相关联用于一个或多个任务的并发执行。此外，单个 GPC208 可以并发执行多个任务。

[0050] 参数存储器(未示出)存储可由同一个 CTA 内的任何线程(或任何 LSU303)读取但不可由其写入的运行时间参数(常量)。在一个实施例中，设备驱动程序 103 在引导 SM310 开始执行使用参数的任务之前将这些参数提供给参数存储器。任何 CTA 内的任何线程(或 SM310 内的任何 exec 单元 302)可以通过存储器接口 214 访问全局存储器。可以将全局存储器的各部分存储在 L1 高速缓存 320 中。

[0051] 每个线程将本地寄存器堆 304 用作暂存空间；每个寄存器被分配以专用于一个线程，并且在本地寄存器堆 304 的任何部分中的数据仅对于寄存器被分配到的线程可访问。本地寄存器堆 304 可以实现为物理上或逻辑上分为 P 个通道的寄存器堆，每个通道具有一定数目的条目(其中每个条目可以存储例如 32 位字)。将一个通道指派到 N 个 exec 单元 302 和 P 个下载 - 存储单元 LSU303 的每一个，并且利用用于执行同一个程序的不同线程的数据来填充不同通道中的相应条目以帮助 SIMD 执行。可以将通道的不同部分分配到 G 个并发线程组中的不同线程组，以使得本地寄存器堆 304 中的给定条目仅对于特定线程可访问。在一个实施例中，保留本地寄存器堆 304 内的某些条目用于存储线程标识符，实现特殊寄存器之一。此外，一致 L1 高速缓存 375 存储用于 N 个 exec 单元 302 和 P 个下载 - 存储单元 LSU303 的每个通道的一致值或常量值。

[0052] 共享存储器 306 对于单个 CTA 内的线程可访问；换言之，共享存储器 306 中的任何位置对于同一个 CTA 内的任何线程(或对于 SM310 内的任何处理引擎)可访问。共享存储器 306 可以实现为具有允许任何处理引擎对共享存储器中的任何位置读取或写入的互连的共享寄存器堆或共享片上高速缓存存储器。在其他实施例中，共享状态空间可能映射到片外存储器的每 CTA 区上并被高速缓存在 L1 高速缓存 320 中。参数存储器可以实现为在实现共享存储器 306 的同一个共享寄存器堆或共享高速缓存存储器内的指定部分，或者实现为 LSU303 对其具有只读访问权限的分开的共享寄存器堆或片上高速缓存存储器。在一个实施例中，实现参数存储器的区域还用于存储 CTA ID 和任务 ID，以及 CTA 和网格维数或队列位置，实现特殊寄存器的各部分。SM310 中的每个 LSU303 耦连到统一地址映射单元 352，统一地址映射单元 352 将为在统一存储器空间中所指定的加载和存储指令所提供的地址转换为每个各异存储器空间中的地址。因此，指令可以用于通过指定统一存储器空间中的地址来访问本地、共享或全局存储器空间中的任何一个。

[0053] 每个 SM310 中的 L1 高速缓存 320 可以用于高速缓存私有的每线程本地数据还有每应用全局数据。在一些实施例中，可以将每 CTA 共享数据高速缓存在 L1 高速缓存 320 中。LSU303 经由存储器和高速缓存互连 380 耦连到共享存储器 306 和 L1 高速缓存 320。

[0054] 应该理解，本文所描述的核心架构是示例性的并且变化和修改是可能的。任何数

目的处理单元例如 SM310 可以包括在 GPC208 内。进一步地，如图 2 所示，PPU202 可以包括任何数目的 GPC208，其有利地功能上彼此类似使得执行行为不取决于哪个 GPC208 接收特定处理任务。进一步地，每个 GPC208 有利地使用分开并且各异的处理单元、L1 高速缓存独立于其他 GPC208 进行操作以执行用于一个或多个应用程序的任务。

[0055] 本领域普通技术人员将理解的是，图 1-3B 所描述的架构决不限制本发明的范围并且本文所教导的技术可以实现在任何经适当配置的处理单元上而不脱离本发明的范围，所述处理单元包括但不限于一个或多个 CPU、一个或多个多核心 CPU、一个或多个 PPU202、一个或多个 GPC208、一个或多个图形或专用处理单元等等。

[0056] 图形管线架构

[0057] 图 4 是根据本发明的一个实施例的、图 2 的 PPU202 中的一个或多个可配置为实现其的图形处理管线 400 的示意图。例如，SM310 中的一个可配置为实现以下各项中的一个或多个的各功能：顶点处理单元 415、曲面细分初始化处理单元 420、曲面细分处理单元 440、几何处理单元 445 以及片段处理单元 460。基元分布器和顶点属性获取 410、任务生成单元 425、任务分布器 430、拓扑生成单元 435、视口缩放、剔除以及修剪单元 450、光栅器 455 以及光栅操作单元 465 的各功能也可由 GPC208 内的其他处理引擎以及相应的分区单元 215 来实施。替代地，图形处理管线 400 可以使用用于一个或多个功能的专用处理单元来实现。

[0058] 图形处理管线还包括本地于 SM310 内的图形处理管线 400 的共享存储器 306。如下文进一步所描述的，共享存储器 306 内的级间缓冲区（未示出）由图形处理管线 400 内的各处理单元按需要进行分配和解除分配。处理单元从一个或多个级间缓冲区读取输入数据，处理输入数据以产生输出数据，以及将产生的输出数据存储在一个或多个级间缓冲区中。随后的处理单元处理数据并且将输出数据存储在一个或多个级间缓冲区中等等。共享存储器 306 和图形处理管线的各其他级经由存储器接口 214 与外部存储器连接。

[0059] 基元分布器 410 处理单元采集用于高阶表面的顶点数据、基元等等，并且将包括顶点属性的顶点数据输出到顶点处理单元 415。在一些实施例中，基元分布器 410 包括检索顶点属性和将顶点属性存储在共享存储器 306 中的顶点属性获取单元（未示出）。顶点处理单元 415 是可编程执行单元，其配置为执行顶点着色器程序，照明和变换如由顶点着色器程序所指定的顶点数据。例如，顶点处理单元 415 可被编程以将顶点数据从基于对象的坐标表示（对象空间）变换到可替代地基于的坐标系，诸如世界空间或规格化设备坐标（NDC）空间。顶点处理单元 415 可通过基元分布器 410 来读取存储在共享存储器 306、L1 高速缓存 320、并行处理存储器 204 或系统存储器 104 中的数据用于在处理顶点数据中使用。顶点处理单元 415 将经处理的顶点存储在共享存储器 306 内的级间缓冲区中。

[0060] 曲面细分初始化处理单元 420 是配置为执行曲面细分初始化着色器程序的可编程执行单元。曲面细分初始化处理单元 420 处理由顶点处理单元 415 所产生的顶点并且生成被称为补丁（patch）的图形基元。曲面细分初始化处理单元 420 还生成各种补丁属性。曲面细分初始化处理单元 420 然后将补丁数据和补丁属性存储在共享存储器 306 内的级间缓冲区中。在一些实施例中，曲面细分初始化着色器程序可以叫做外壳（hull）着色器或曲面细分控制着色器。

[0061] 任务生成单元 425 检索用于来自共享存储器 306 的级间缓冲区的顶点和补丁的数据和属性。任务生成单元 425 生成用于处理顶点的任务以及用于由图形处理管线 400 中的

稍后的级处理的补丁。

[0062] 任务分布器 430 重新分布由任务生成单元 425 所产生的任务。由顶点着色器程序和曲面细分初始化程序的各实例所产生的任务可以在一个图形处理管线 400 和另一个图形处理管线 400 之间显著地变化。任务分布器 430 重新分布这些任务，使得每个图形处理管线 400 在稍后管线级期间具有接近相同的工作量。

[0063] 拓扑生成单元 435 检索由任务分布器 430 所分布的任务。拓扑生成单元 435 对包括与补丁相关联的顶点的顶点进行索引并且计算与顶点相应的纹理坐标。拓扑生成单元 435 然后将经索引的顶点存储在共享存储器 306 内的级间缓冲区中。

[0064] 曲面细分处理单元 440 是配置为执行曲面细分着色器程序的可编程执行单元。曲面细分处理单元 440 从共享存储器 306 的级间缓冲区读取输入数据并且将输出数据写到共享存储器 306 的级间缓冲区。级间缓冲区中的该输出数据被移交到下一个着色器级几何处理单元 445 作为输入数据。在一些实施例中，曲面细分着色器程序可以叫做域着色器或曲面细分评估着色器。

[0065] 几何处理单元 445 是配置为执行几何着色器程序从而变换图形基元的可编程执行单元。顶点被分组以构建图形基元用于处理，其中图形基元包括三角形、线段、点等等。例如，几何处理单元 445 可经编程以将图形基元再分成一个或多个新图形基元以及计算用来将新图形基元进行光栅化的参数，诸如平面方程系数。

[0066] 在一些实施例中，几何处理单元 445 也可以添加或删除几何流中的元素。几何处理单元 445 将指定新图形基元的参数和顶点输出到视口缩放、剔除以及修剪单元 450。几何处理单元 445 可以读取存储在共享存储器 306、并行处理存储器 204 或系统存储器 104 中的数据用于在处理几何数据中使用。视口缩放、剔除和修剪单元 450 实施修剪、剔除和视口缩放并且将经处理的图形基元输出到光栅器 455。

[0067] 光栅器 455 扫描对新图形基元进行转换并且将片段和覆盖数据输出到片段处理单元 460。此外，光栅器 455 可配置为实施 z 剔除以及其他基于 z 的优化。

[0068] 片段处理单元 460 是配置为执行片段着色器程序的可编程执行单元，其变换由片段着色器程序所指定、从光栅器 455 所接收的片段。例如，片段处理单元 460 可被编程以实施诸如透视校正、纹理映射、着色、混合等等的操作以产生被输出到光栅操作单元 465 的经着色片段。片段处理单元 460 可以读取存储在共享存储器 306、并行处理存储器 204 或系统存储器 104 中的数据用于在处理片段数据中使用。片段可以在像素、样本或其他粒度（granularity）上进行着色，这取决于所编程的采样率。

[0069] 光栅操作单元 465 是实施光栅操作以及输出像素数据作为经处理的图形数据用于图形存储器中的存储的处理单元，所述光栅操作诸如模板、z 测试、混合等等。经处理的图形数据可以包括在图形存储器例如并行处理存储器 204 和 / 或系统存储器 104 中用于显示设备 110 上的显示或者用于由 CPU102 或并行处理子系统 112 进一步处理。在本发明的一些实施例中，光栅操作单元 465 配置为压缩写到存储器的 z 或颜色数据以及解压缩从存储器读取的 z 或颜色数据。在各实施例中，ROP465 可以位于存储器接口 214、GPC208、GPC 外的处理集群阵列 230 或 PPU202 内的分开的单元（未示出）中。

[0070] 处置光栅操作中的 post-z 覆盖数据

[0071] 在图形处理管线 400 的某些级处实施 z 测试允许图形处理管线 400 提早丢弃某些

片段，使得进一步的处理周期不浪费在最后的经渲染图像中不可见的片段上。典型地，这类z测试的结果不直接存储在渲染目标中。如下文所进一步描述的，一些应用从将z测试结果存储在渲染目标中用于随后的处理操作中获益。

[0072] 图5示出了根据本发明的一个实施例的、图4的片段处理单元460和光栅操作单元465。如所示，片段处理单元460包括早z-光栅(z-raster)操作(ZROP)单元510、pre-z/post-z多路复用器520以及片段着色器530。如进一步所示，光栅操作单元465包括晚ZROP单元540、颜色/覆盖多路复用器545以及颜色光栅操作(CROP)单元550。

[0073] 早ZROP单元510从光栅器455接收片段数据，其中片段数据包括但不限于z平面方程数据、用于颜色分量和纹理坐标的平面方程数据、屏幕空间位置(x,y)以及光栅化覆盖掩码。早ZROP单元510使用由多样本模式所指定的样本位置和由光栅器455所提供的平面方程数据计算用于每个样本的z信息。早ZROP单元510将用于当前样本的z值比作先前所存储的用于相应样本位置的z值。这类过程常规地被称为“z测试”或“隐藏面消除”。在一些实施例中，早ZROP单元510可将通过z测试的样本的z值写到经配置作为深度(或z)缓冲区的渲染目标。早ZROP单元510丢弃未通过z测试的样本，并且早ZROP单元510不写入任何用于这些所丢弃样本的z值。

[0074] 早ZROP单元510还基于给定像素的哪些样本由当前被处理的片段所覆盖来计算覆盖信息。当早ZROP单元510单独地基于当前被处理的片段计算覆盖掩码时，这类覆盖掩码被称为pre-z覆盖信息570。当早ZROP单元510基于当前被处理的片段和先前经渲染的片段数据二者来计算覆盖掩码时，这类覆盖掩码被称为post-z覆盖信息580。早ZROP单元510将用于每个片段的覆盖掩码提供到pre-z/post-z多路复用器520。在一些实施例中，早ZROP单元510也可以计算当确定特定样本通过或者未通过z测试时结合z值使用的模板值。

[0075] pre-z/post-z多路复用器520基于控制信号560的状态选择从早ZROP单元510传达到片段着色器530的pre-z覆盖信息570或post-z覆盖信息580。pre-z/post-z多路复用器520从SM310中的机构(未示出)接收控制信号560作为状态属性。然而，图形处理管线400中的任何技术上可行的单元可以提供控制信号560。

[0076] 片段着色器530使用经由pre-z/post-z多路复用器520从光栅器455所接收的片段数据来计算和处理用于像素的一个或多个样本位置的颜色值和其他像素信息。在一些实施例中，片段着色器530还可以使用由光栅器455所提供的平面方程数据针对一个或多个样本位置来修改从pre-z/post-z多路复用器520所接收的z值。可替代地，片段着色器530可以计算新z值而不使用由光栅器455所提供的平面方程数据。可替代地，片段着色器530可在没有修改的情况下传达从从pre-z/post-z多路复用器520所接收的z值。片段着色器530将颜色信息582传送到颜色/覆盖多路复用器545。

[0077] 除晚ZROP单元在片段着色器530完成对片段的处理之后实施z测试之外，晚ZROP单元540功能本质上与早ZROP单元510相同。当片段着色器530针对一个或多个片段创建z信息时、当片段着色器530修改与一个或多个片段相关联的现存的z信息时或者当可由早ZROP单元510计算的z信息否则无效时，晚ZROP单元540用于z测试。晚ZROP单元540将覆盖信息572传达到颜色/覆盖多路复用器545。

[0078] 颜色/覆盖多路复用器545基于控制信号562的状态选择传达到CROP单元550

的来自片段着色器530的颜色信息582或者来自晚ZROP单元540的覆盖信息572。控制/覆盖多路复用器545从SM310中的机构(未示出)接收控制信号562作为状态属性。然而，图形处理管线400中的任何技术上可行的单元可以提供控制信号562。

[0079] CROP单元550对从片段着色器530所接收的像素信息和存储在一个或多个着色器目标中的像素信息实施各种混合或合成操作。CROP单元550基于从片段着色器所接收的pre-z或post-z覆盖信息将这类混合或合成操作的结果存储在一个或多个渲染目标中。如下文所进一步描述的，补充或者取代存储混合或合成操作的结果，CROP单元550将post-z覆盖信息存储在一个或多个渲染目标中。CROP单元550经由分区单元215将信息存储在渲染目标中。

[0080] 图6示出了根据本发明的一个实施例的、如经由图2的分区单元215中的一个或多个所存储的渲染目标的集合600。如所示，渲染目标的集合600包括8个分开的渲染目标610(0)-610(7)。

[0081] 第一渲染目标610(0)包括表示与相应样本或片段相关联的透明度信息和颜色的4个字段。如所示，4个字段包括红值615、绿值620、蓝值625以及阿尔法(alpha)或透明度值630。

[0082] 第二渲染目标610(0)包括表示与相应样本或片段相关联的深度和模板信息的2个字段。如所示，2个字段包括z或者深度值635和模板掩码640。如所示，z值635比模板掩码640包括更多的位。

[0083] 第三渲染目标610(2)包括表示与相应样本或片段相关联的表面法向量信息的4个字段。如所示，4个字段包括x轴法向量645、y轴法向量650、z轴法向量655。在用于第三渲染目标610(2)的该特定配置中，第四字段是未使用的660。

[0084] 第四渲染目标610(3)包括表示与相应样本或片段相关联的post-z覆盖信息665的单个字段。如本文所描述的，CROP单元550将post-z覆盖信息存储在渲染目标中。在一个实施例中，CROP单元可将这类post-z覆盖信息存储在第四渲染目标610(3)的post-z覆盖信息665中。

[0085] 剩余的渲染目标610(4)-610(7)配置为存储与相应样本或片段相关联的附加的信息。这类配置(未示出)包括用于各种信息的存储，包括但不限于3D位置数据、漫射照明信息以及反射照明信息。

[0086] 应该理解，本文所描述的架构仅是示例性的并且变化和修改是可能的。在一个示例中，本文在给定配置中的8个渲染目标610的上下文中描述了技术。然而，所描述的技术可使用任何数目的渲染目标610加以采用。每个渲染目标可独立于其他渲染目标配置为包括任何数目的字段。渲染目标内的每个字段可独立于其他字段配置为包括任何数目的位。在另一个示例中，渲染目标的集合600包括具有用于post-z覆盖信息665的存储的单个字段的渲染目标610(3)。然而，只要字段包括足够数量的位以存储post-z覆盖信息，post-z覆盖信息就可以存储在任何渲染目标610中的任何技术上可行的字段中。具体地，post-z覆盖信息可以存储在第三渲染目标610(2)的第四未使用字段660中。可替代地，当具体应用不使用模板掩码时，post-z覆盖信息可以存储在第二渲染目标610(1)的目标掩码字段640中。

[0087] 在另一个示例中，CROP单元550可尝试将post-z覆盖信息存储在被认为无资格存

储这类信息的字段中。在这种情况下，CROP 单元 550 可实施任何合适的操作，包括但不限于，将 post-z 覆盖信息存储在无资格字段中，丢弃存储操作使得存储在字段中的数据不被覆写，或者基于由当前应用所指示的偏好实施存储或丢弃。在又一个示例中，如上文所描述的，CROP 单元 550 将 post-z 覆盖信息存储在渲染目标 610 中。然而，图形处理管线 400 内的任何技术上可行的单元可将 post-z 覆盖数据存储在渲染目标 610 中，包括但不限于片段着色器 530、早 ZROP 单元 510 以及晚 ZROP 单元 540。在另一个示例中，本文所描述的系统包括用于 z- 光栅操作的两个分开的单元，具体地早 ZROP 单元 510 和晚 ZROP 单元 540。然而，本文所描述的技术可以结合具有可配置为实施早 z- 光栅操作或晚 z- 光栅操作的单个 ZROP 单元（未示出）的技术来使用。这类共享 ZROP 单元经由通信路径、多路复用器等等将可配置为在片段着色器 530 之前实施 z 测试（早 z 测试）或者在片段着色器 530 之后实施 z 测试（晚 z 测试）。

[0088] 图 7A-7D 示出了根据本发明的一个实施例的、相交多个图形基元 730735740 的像素 710。如图 7A 所示，每个像素 710 再分成 16 个样本 720（0）-720（15）。如本文所进一步描述的，post-z 覆盖数据基于由图形基元 730735740 所覆盖的样本 720 来针对图形基元 730735740 中的每一个进行计算。在一个实施例中，post-z 覆盖数据可由位掩码表示，其中图中的每个位包括用于给定样本 720 的 post-z 覆盖数据。这类位掩码可以是 16 位宽，以包括像素 710 的 16 个样本 720（0）-720（15）的 post-z 覆盖数据。位掩码的最高有效位可以与样本 720（15）相应，而最低有效位可以与样本 720（0）相应。位掩码中的位位置（bit position）中的“1”值可以指示图形基元 730735740 覆盖相应的样本 720，而位掩码的位位置中的“0”可以指示图形基元 730735740 不覆盖相应的样本 720。

[0089] 如图 7B 所示，第一图形基元 730 与像素 710 相交。图形基元 730 覆盖阴影样本 720（0）、720（1）、720（4）、720（5）、720（8）、720（9）、720（12）以及 720（13）。相应的位掩码可以是 0x3333，反映由图形基元 730 覆盖的像素 710 的 8 个样本。CROP 单元 550 将渲染目标 610 中的该 post-z 覆盖位掩码存储在与像素 710 相应并且与图形基元 730 相关联的位置处。post-z 覆盖位掩码可以存储在渲染目标中的多个位置中，其中每个位置表示被覆盖在相同像素 710 内的样本 720。

[0090] 如图 7C 所示，第二图形基元 735 与像素 710 相交。图形基元 735 覆盖阴影样本 720（2）、720（3）、720（6）、720（7）、720（10）、720（11）、720（14）以及 720（15）。相应的位掩码可以是 0xCCCC，反映由图形基元 735 覆盖的像素 710 的 8 个样本。CROP 单元 550 将渲染目标 610 中的该 post-z 覆盖位掩码存储在与像素 710 相应并且与图形基元 735 相关联的位置处。post-z 覆盖位掩码可以存储在渲染目标中的多个位置中，其中每个位置表示被覆盖在相同像素 710 内的样本 720。

[0091] 如图 7D 所示，第三图形基元 740 与像素 710 相交。图形基元 740 在图形基元 730 前面并且在图形基元 735 后面。在 z 测试之前，图形基元 740 覆盖 6 个样本 720（9）、720（10）、720（11）、720（13）、720（14）以及 720（15）。相应的 pre-z 覆盖位掩码可以是 0xEE00，反映 z 测试之前由图形基元 740 覆盖的像素 710 的 6 个样本。在 z 测试之后，图形基元 740 覆盖阴影样本 720（9）和 720（13）。相应的 post-z 覆盖位掩码可以是 0x2200，反映 z 测试之后由图形基元 740 覆盖的像素 710 的 2 个样本。CROP 单元 550 将渲染目标 610 中的该 post-z 覆盖位掩码存储在与像素 710 相应并且与图形基元 740 相关联的位置

处。post-z 覆盖位掩码可以存储在渲染目标中的多个位置中，其中每个位置表示被覆盖在相同像素 710 内的样本 720。

[0092] 在渲染图形基元 740 之后，图形基元 730 仅覆盖 6 个样本 720（0）、720（1）、720（4）、720（5）、720（8）以及 720（12）。然而，如上文结合图 7A 所描述的，图形基元 730 在图形基元 730 原始被渲染时覆盖 8 个样本。在一个实施例中，截至每个图形基元被首次渲染时，post-z 覆盖掩码可以表示 post-z 覆盖。在这种情况下，用于图形基元 730 的 post-z 覆盖位掩码可以保持为 0x3333，反映在渲染时由图形基元 730 覆盖的像素 710 的 8 个样本。在另一个实施例中，在场景中的所有图形基元被渲染之后，post-z 覆盖掩码可以表示 post-z 覆盖。在这种情况下，用于图形基元 730 的 post-z 覆盖位掩码可以修改到 0x1133，反映渲染图形基元 740 之后由图形基元 730 覆盖的像素 710 的 6 个样本。

[0093] 在一些实施例中，像素 710 包括已通过可见性测试的样本。这类可见性测试可以包括确定样本是否在最后经渲染图像中可见的任何一个或多个测试，包括但不限于以任何组合的深度测试、模板测试、阿尔法测试以及深度限度测试。可见性测试可以在样本已由片段着色器 530 处理之后实施。在可替代实施例中，样本可以不由片段着色器 530 处理。在这种情况下，片段着色器 530 可被断电或处于低功率状态中。

[0094] post-z 覆盖信息可以使用在除典型 3D 渲染技术之外的各种应用中。在一个示例中，post-z 覆盖信息可以用来支持延迟渲染（deferred rendering）。利用延迟渲染，片段处理单元 460 或 ROP 单元 465 将片段数据直接存储在一个或多个渲染目标 610 中而不对片段数据进行渲染。在渲染目标 610 积累与多个图形基元相关联的片段数据之后，片段处理单元 460 或 ROP 单元 465 从渲染目标 610 检索片段数据，对所检索的片段数据实施一个或多个混合或合成操作，以及将产生的颜色值或其他像素信息存储在渲染目标中。post-z 覆盖数据可以用来确定由所存储片段中的每一个所影响的样本的集合。

[0095] 在另一个示例中，post-z 覆盖信息可以用来支持路径渲染。路径渲染是 2D 图形渲染技术，其中场景指定为分辨率独立的轮廓线（outline）的序列，轮廓线被称为可以填充或划掉的（stroked）路径。这些路径，也称为轮廓线，指定对象依照用于画连接线、曲线和弧线的命令的序列进行渲染。这些路径可以是凹的，可以自相交，可以包含孔并且可以是任意复杂的。这类路径可以利用常量颜色、线性或径向梯度、或者概述为形成（所划掉的）路径的相反图样的（所填充的）图像。随着特定路径被遍历，post-z 覆盖信息可以用来确定样本是在由路径所划定的空间之内还是之外。用于样本的集合的 post-z 覆盖信息可以初始化到 0。随着片段被处理，着色器程序 530 可以实施对用于当前片段的 post-z 覆盖信息与存储在渲染目标中的 post-z 覆盖信息的 XOR 操作。样本第一次与片段相交时，post-z 覆盖信息可以设定到 1。post-z 覆盖信息可在样本第二次与片段相交时重新设定到 0，在样本第三次与片段相交时返回设定到 1 等等。一旦遍历完成，则“1”值可以指示样本在由路径所划定的空间内部，而“0”值可以指示样本在由路径所划定的空间外部。

[0096] 可替代地，由路径所划定的空间可以随着路径被遍历而进行填充。着色器程序 530 可以实施对用于当前片段的 post-z 覆盖信息与存储在渲染目标中的 post-z 覆盖信息的 OR 操作。一旦遍历完成，则“1”值可以指示样本在遍历期间被填充，而“0”值可以指示样本在遍历期间未被填充。

[0097] 在又一个实施例中，post-z 覆盖信息可以用来支持目标独立光栅化。利用目标独

立光栅化，每像素样本的数目可独立于针对存储经渲染图像所分配的存储器来进行指定。例如，图形处理管线可配置为对每像素 8 个样本进行光栅化，而 CROP 单元 550 配置为写到可配置为存储每像素 1 个样本的单个渲染目标。这类配置可以叫做 8:1 模式。在这种情况下，光栅化可以以与渲染目标的配置无关的速率行进。post-z 覆盖可以用来确定由一个或多个图形基元覆盖的给定像素中样本的数目。如果 16 个中的 8 个样本被覆盖，那么像素具有 50% 覆盖率，如果 16 个中的 12 个样本被覆盖，那么像素具有 75% 覆盖率等等。CROP 单元 550 然后可以计算用于如由覆盖率的百分比所加权的像素的单个颜色值。

[0098] 最后，post-z 覆盖信息可以用来确定片段的 pre-z 或 post-z 覆盖质心 (centroid)。返回参考图 7D，像素 710 的质心是处于由像素 710 覆盖的区域的中心处的点。然而，该点不是由图形基元 740 覆盖的像素 710 的一部分的质心。如上文所描述的，图形基元 740 的 pre-z 覆盖包括 6 个样本 720 (9)、720 (10)、720 (11)、720 (13)、720 (14)以及 720 (15)。通过检查 pre-z 覆盖数据，像素 710 与图形基元 740 的相交的 pre-z 质心可以确定为邻近样本 720 (10) 和 720 (14) 的线段的中点。图形基元 740 的 post-z 覆盖包括 2 个样本 720 (9)和 720 (13)。通过检查 post-z 覆盖数据，像素 710 与图形基元 740 的相交的 post-z 质心可以确定为邻近样本 720 (9) 和 720 (13) 的线段的中点。

[0099] 图 8 阐述了根据本发明的一个实施例的、用于存储 post-z 覆盖数据的方法步骤的流程图。尽管结合图 1-6 的系统描述了方法步骤，但是本领域普通技术人员将理解的是，配置为以任何次序实施方法步骤的任何系统均在本发明的范围内。

[0100] 如所示，方法 800 开始于步骤 802，其中 CROP 单元 550 接收与包括一个或多个样本的片段相关联的 post-z 覆盖信息。在步骤 804，CROP 单元 550 接收与片段相关联的颜色或其他信息。在步骤 806，CROP 单元 550 确定应用是否配置为对 post-z 覆盖信息实施逻辑操作。例如，CROP 单元 550 可以对当前所接收 post-z 覆盖信息与从渲染目标 610 所检索的、先前存储的 post-z 覆盖信息实施逻辑 OR 或 XOR 操作。如果应用配置为实施逻辑操作，那么方法 800 行进到步骤 808，其中 CROP 单元 550 从渲染目标 610 检索现存的 post-z 覆盖信息。

[0101] 在步骤 810，CROP 单元 550 基于所接收的 post-z 覆盖信息和所检索的 post-z 覆盖信息实施逻辑操作。在步骤 812，CROP 单元 550 确定 post-z 覆盖信息的存储操作是否针对有资格渲染目标 610。如果存储操作针对有资格渲染目标，那么方法 800 行进到步骤 814，其中 CROP 单元 550 将 post-z 覆盖信息存储在渲染目标 610 中。在步骤 816，CROP 单元 550 确定附加的片段是否可用于处理。如果附加的片段可用于处理，那么方法 800 返回到步骤 802，如上文所描述的。如果没有附加的片段可用于处理，那么方法 800 终止。

[0102] 返回到步骤 812，如果存储操作针对无资格渲染目标，那么方法 800 行进到上文所描述的步骤 816。

[0103] 返回到步骤 806，如果应用不配置为实施逻辑操作，那么方法 800 行进到上文所描述的步骤 812。

[0104] 总而言之，光栅操作单元将 post-z 覆盖数据存储到渲染目标。post-z 覆盖数据可由早 z 测试单元、晚 z 测试单元或片段着色单元来计算。光栅操作单元将 post-z 覆盖数据存储到单个分量渲染目标。可替代地，光栅操作单元将 post-z 覆盖数据存储到多个分量渲染目标的一个分量。GPU 然后基于 post-z 覆盖数据计算用于样本的颜色和其他像素信息。

post-z 覆盖数据也可以用来支持其他渲染技术诸如延迟渲染、路径渲染，以及计算片段的 post-z 覆盖质心。

[0105] 所公开技术的一个优势在于，GPU 仅计算用于如由 post-z 覆盖数据所确定的可见片段的颜色和其他像素信息。GPU 不计算用于遮蔽片段的颜色和其他像素信息，从而降低总功耗并且改进总渲染性能。

[0106] 本发明的一个实施例可被实施为与计算机系统一起使用的程序产品。该程序产品的程序定义实施例的各功能(包括本文中描述的方法)并且可以被包含在各种计算机可读存储介质上。示例性计算机可读存储介质包括但不限于：(i)不可写的存储介质(例如，计算机内的只读存储器设备，诸如可由 CD-ROM 驱动器读取的光盘只读存储器(CD-ROM)盘、闪存、只读存储器(ROM)芯片或任何类型的固态非易失性半导体存储器)，在其上存储永久性信息；和(ii)可写的存储介质(例如，磁盘驱动器或硬盘驱动器内的软盘或者任何类型的固态随机存取半导体存储器)，在其上存储可更改的信息。

[0107] 以上已参照特定实施例对本发明进行了描述。然而，本领域普通技术人员将理解的是，可对此做出各种修改和变化而不脱离如随附权利要求书中所阐述的本发明的较宽精神和范围。因此，前面的描述以及附图应被视为是例示性而非限制性的意义。

[0108] 因此，本发明的实施例的范围由下面的权利要求书进行阐述。

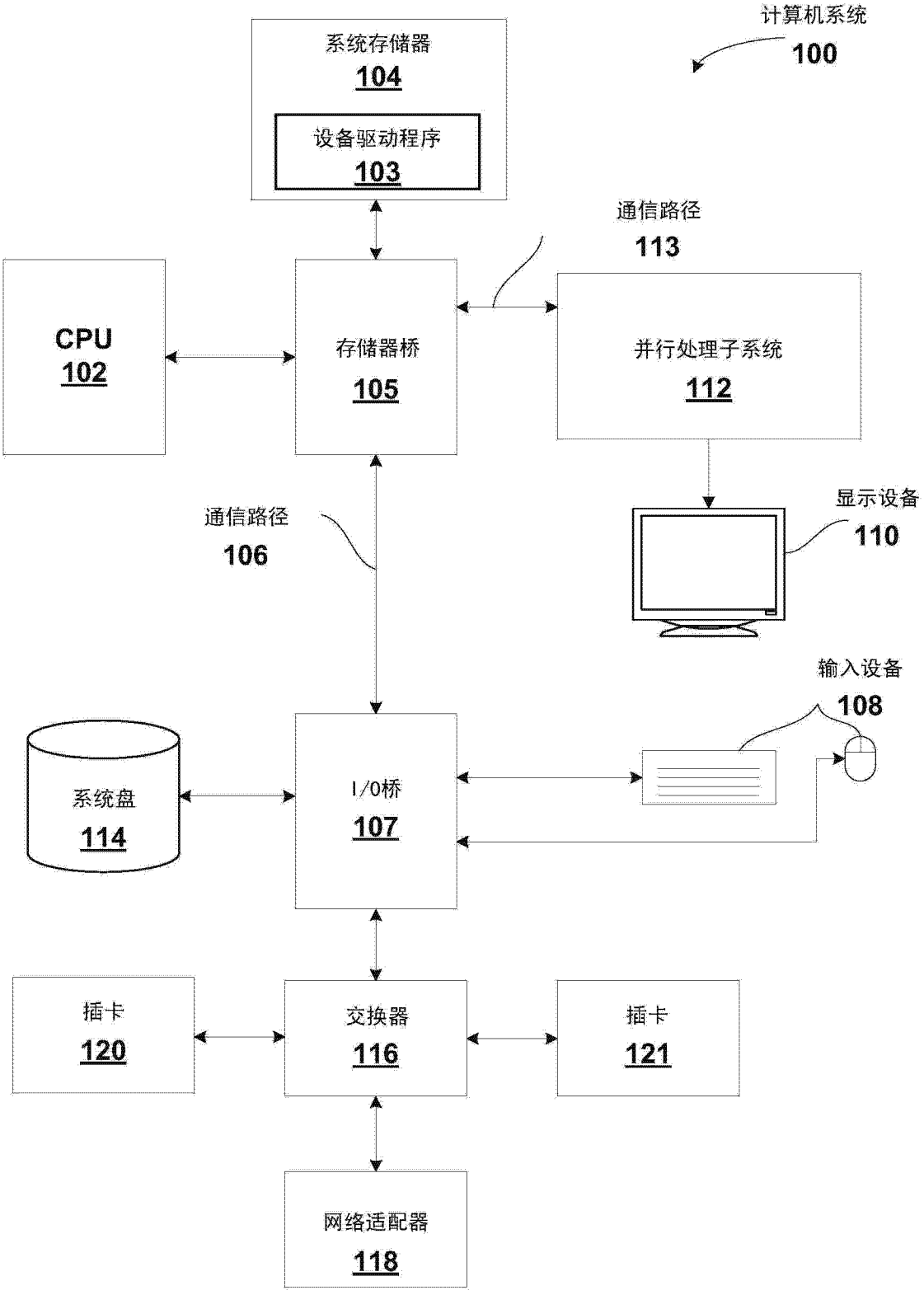

计算机系统
100
系统存储器
104
设备驱动程序
103
通信路径
113
CPU
102
存储器桥
105
并行处理子系统
112
显示设备
110
通信路径
106
输入设备
108
系统盘
114
I/O桥
107
插卡
120
交换器
116
插卡
121
网络适配器
118

图 1

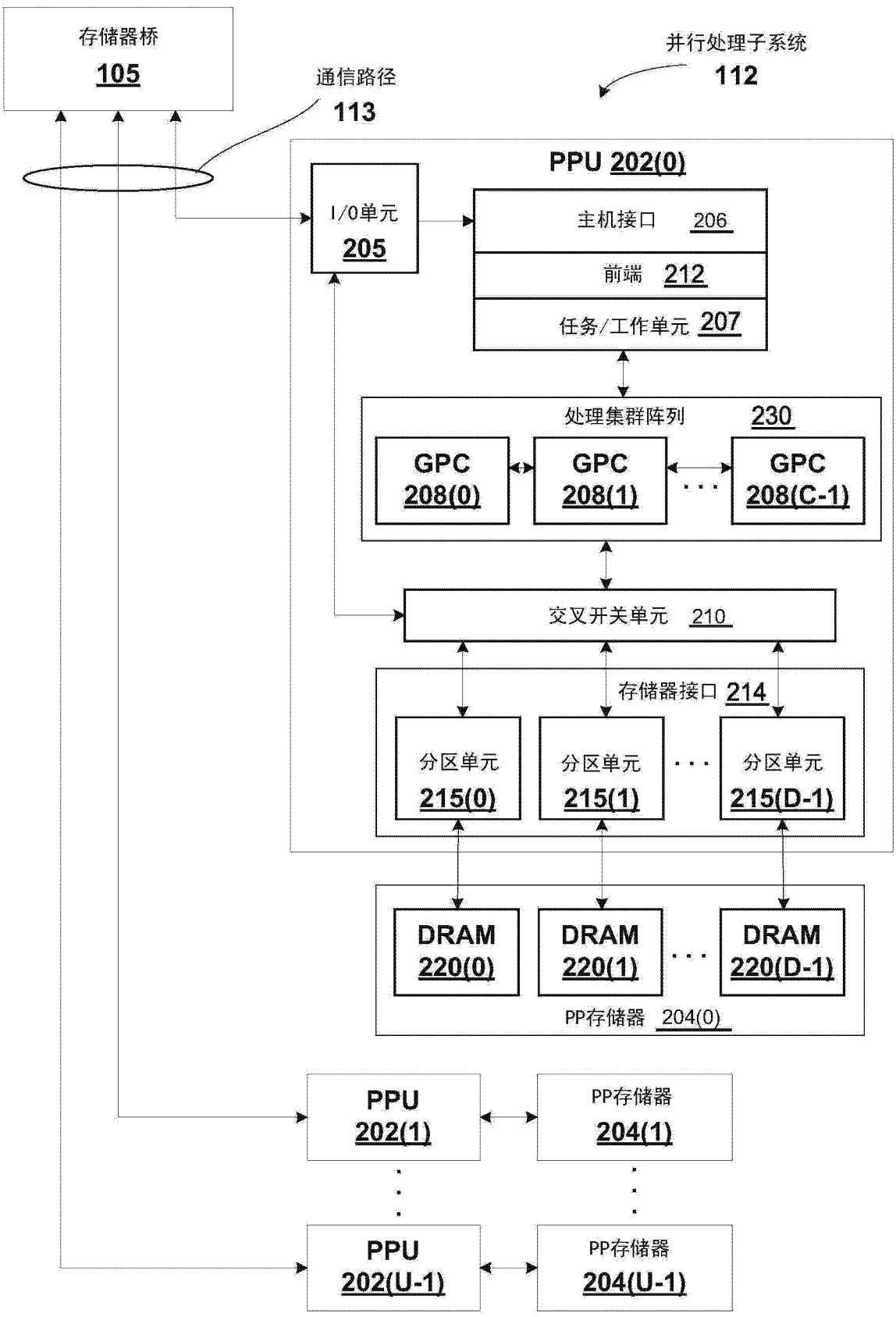

存储器桥
105
通信路径
113
并行处理子系统
112
PPU 202(0)
I/O单元
205
主机接口 206
前端 212
任务/工作单元 207
处理集群阵列 230
GPC
208(0)
GPC
208(1)
GPC
208(C-1)
交叉开关单元 210
存储器接口 214
分区单元
215(0)
分区单元
215(1)
分区单元
215(D-1)
DRAM
220(0)
DRAM
220(1)
DRAM
220(D-1)
PP存储器 204(0)
PPU
202(1)
PP存储器
204(1)
PPU
202(U-1)
PP存储器
204(U-1)

图 2

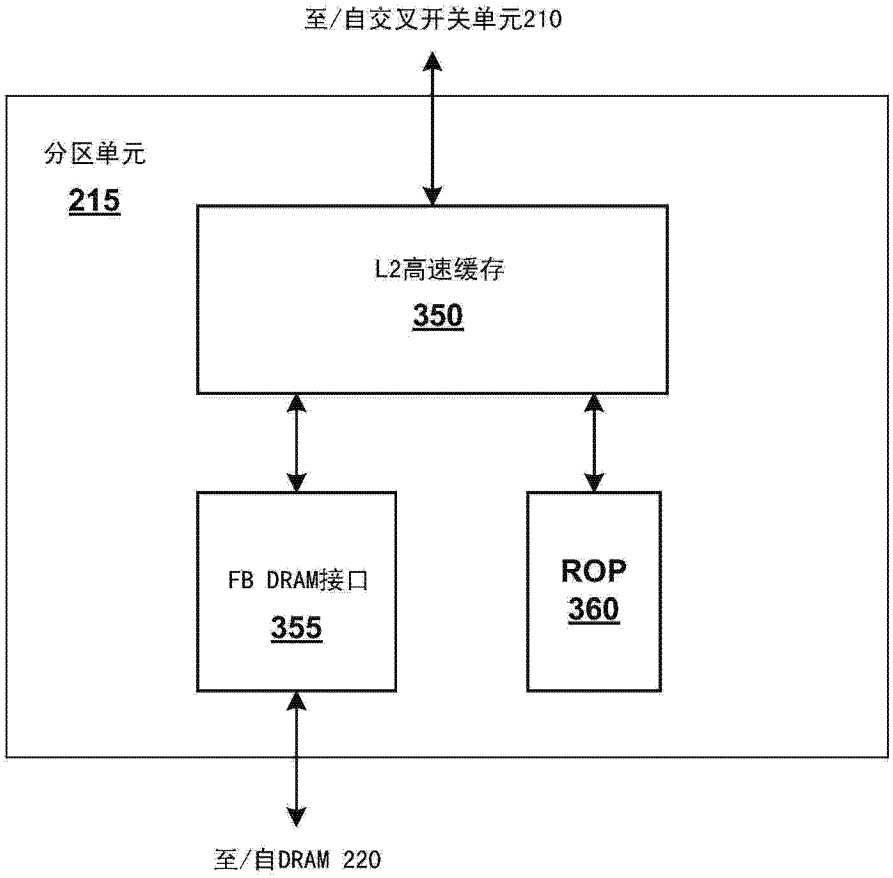

至/自交叉开关单元210
分区单元
215
L2高速缓存
350
FB DRAM接口
355
ROP
360
至/自DRAM 220

图 3A

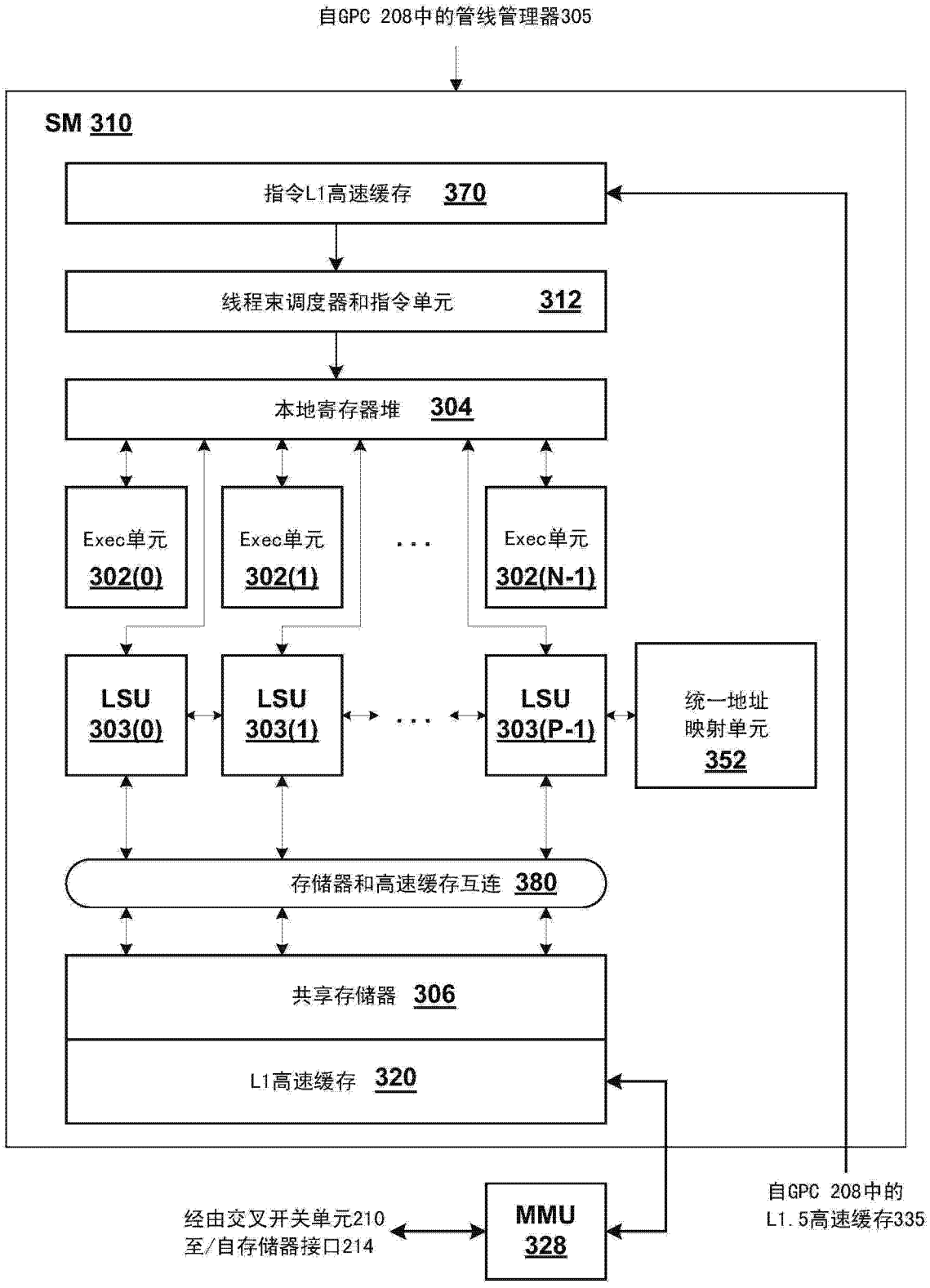

自GPC 208中的管线管理器305
SM 310
指令L1高速缓存 370
线程束调度器和指令单元 312
本地寄存器堆 304
Exec单元 302(0)
Exec单元 302(1)
Exec单元 302(N-1)
LSU 303(0)
LSU 303(1)
LSU 303(P-1)
统一地址映射单元 352
存储器和高速缓存互连 380
共享存储器 306
L1高速缓存 320
MMU 328
经由交叉开关单元210至/自存储器接口214
自GPC 208中的L1.5高速缓存335

图 3B

示意图

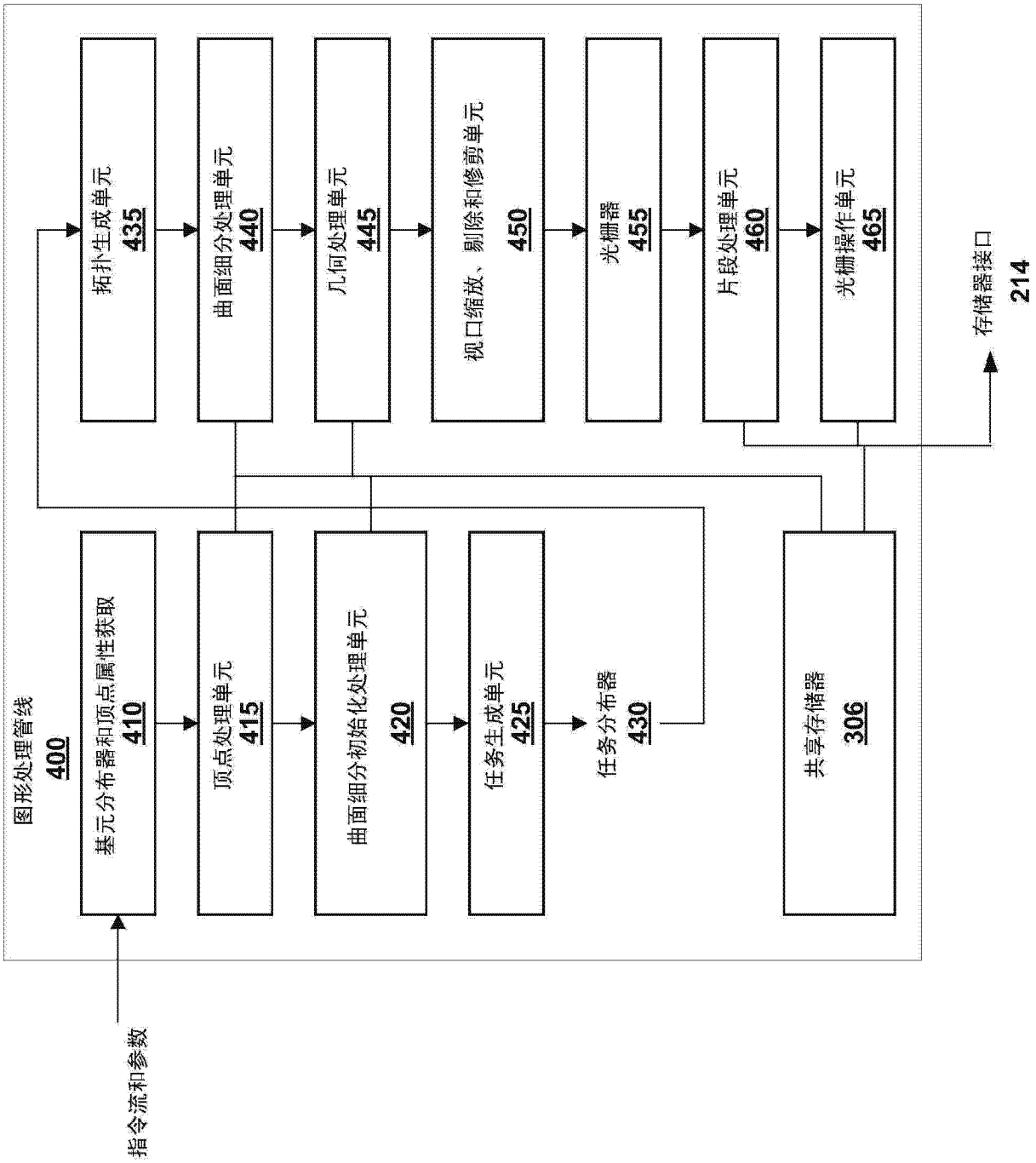

图形处理管线
400
基元分布器和顶点属性获取
410
顶点处理单元
415
曲面细分初始化处理单元
420
任务生成单元
425
任务分布器
430
拓扑生成单元
435
曲面细分处理单元
440
几何处理单元
445
视口缩放、剔除和修剪单元
450
光栅器
455
片段处理单元
460
光栅操作单元
465
共享存储器
306
存储器接口
214
指令流和参数

图 4

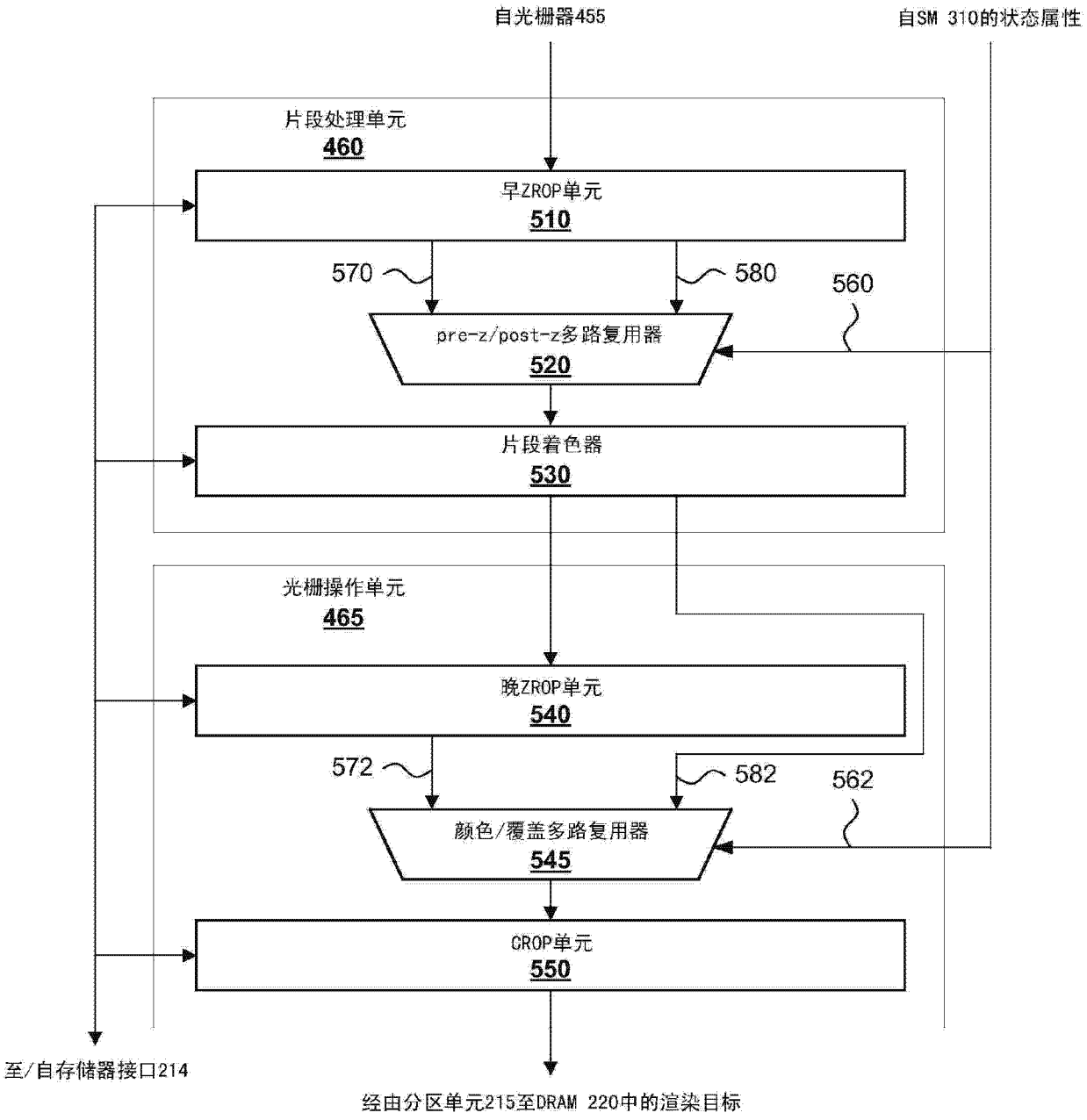

自光栅器455
自SM 310的状态属性
片段处理单元
460
早ZROP单元
510
570
580
560
pre-z/post-z多路复用器
520
片段着色器
530
光栅操作单元
465
晚ZROP单元
540
572
582
562
颜色/覆盖多路复用器
545
CROP单元
550
至/自存储器接口214
经由分区单元215至DRAM 220中的渲染目标

图 5

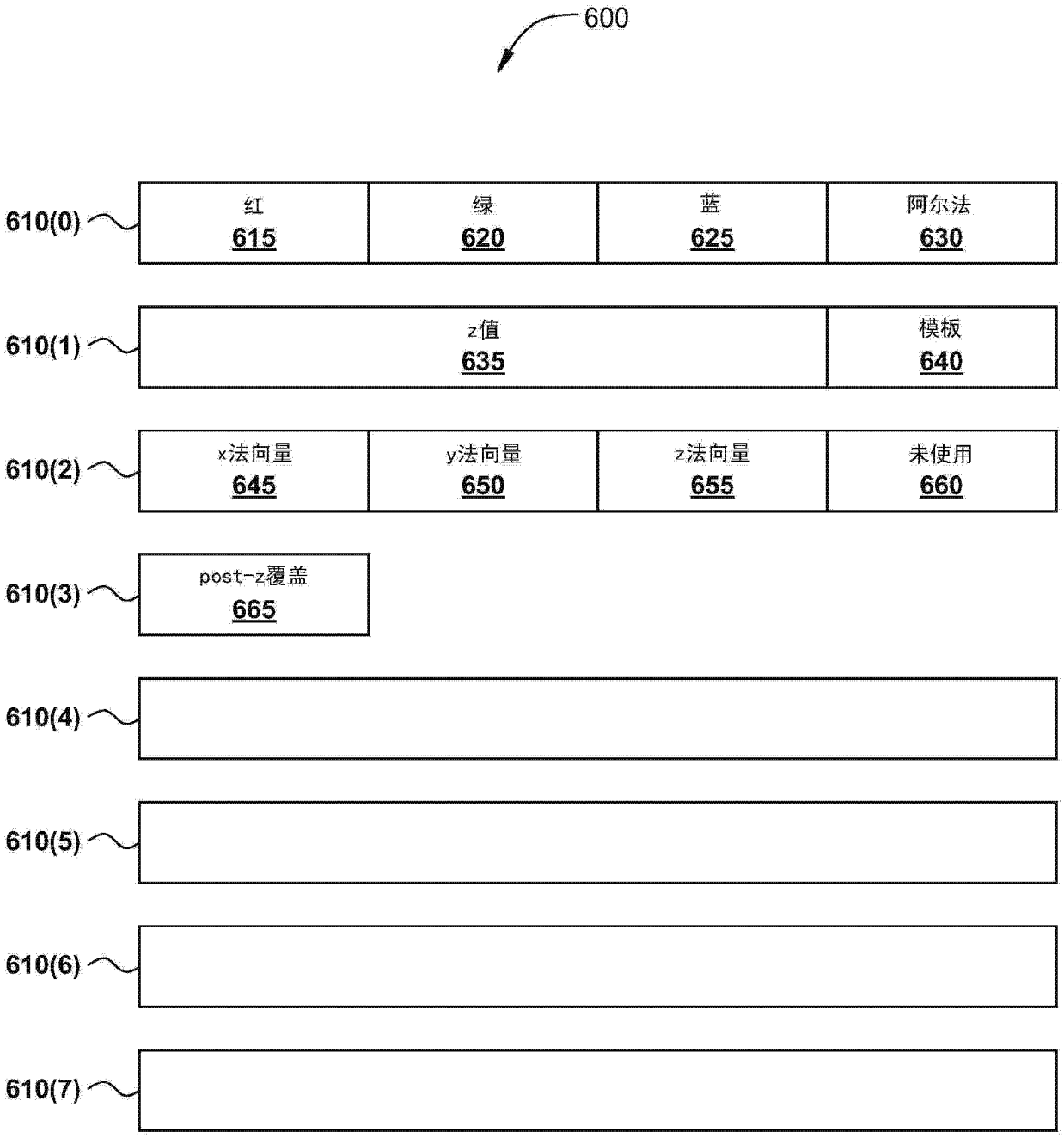

600
610(0)
红
615
绿
620
蓝
625
阿尔法
630
610(1)
z值
635
模板
640
610(2)
x法向量
645
y法向量
650
z法向量
655
未使用
660
610(3)
post-z覆盖
665
610(4)
610(5)
610(6)
610(7)

图 6

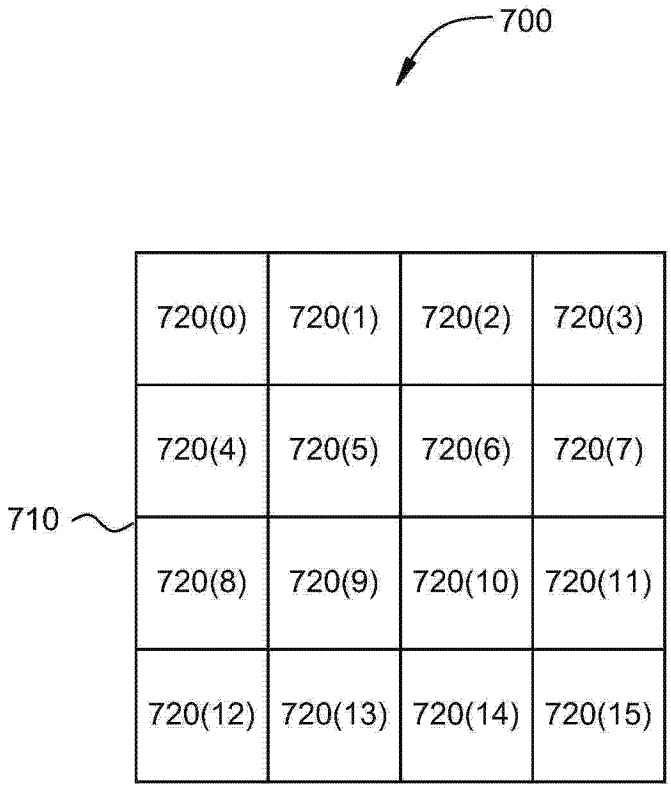

700
710
720(0)
720(1)
720(2)
720(3)
720(4)
720(5)
720(6)
720(7)
720(8)
720(9)
720(10)
720(11)
720(12)
720(13)
720(14)
720(15)

图 7A

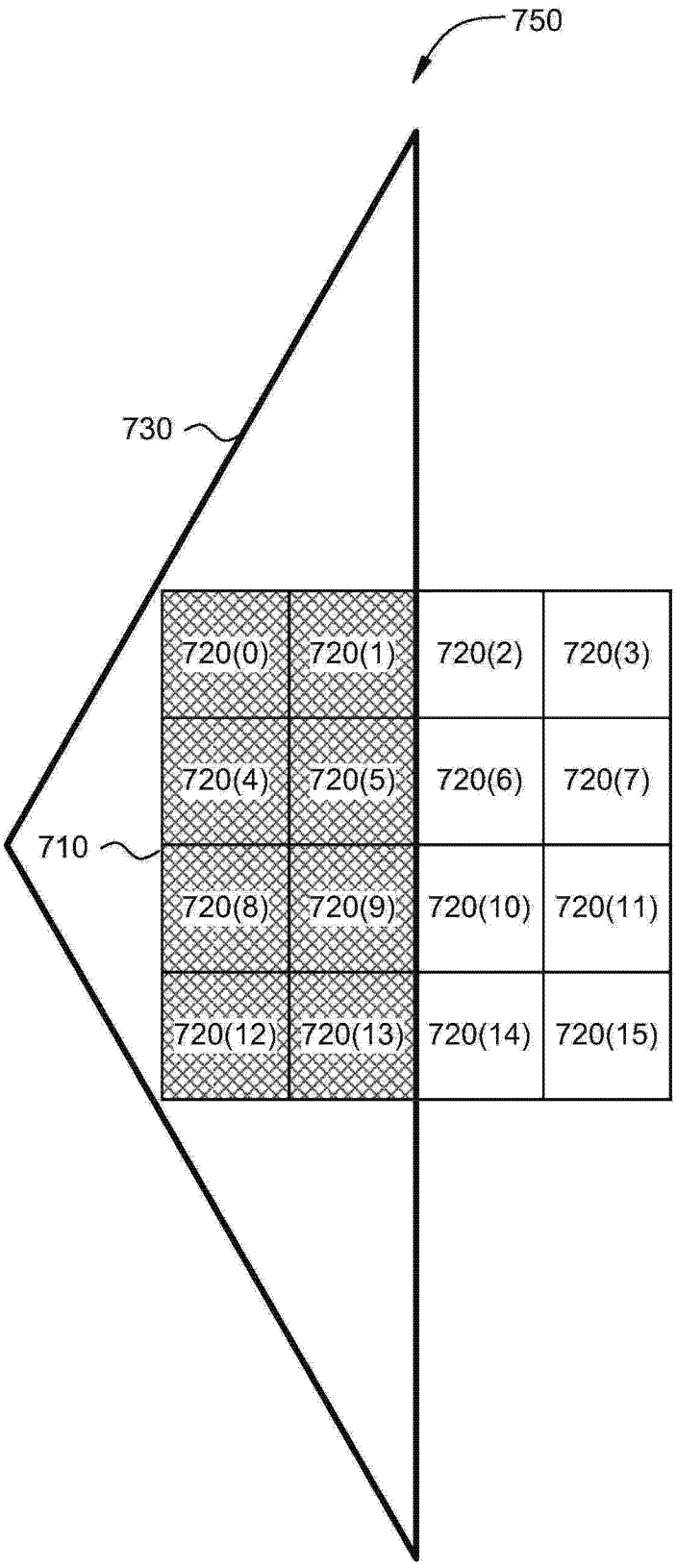

750
730
710
720(0)
720(1)
720(2)
720(3)
720(4)
720(5)
720(6)
720(7)
720(8)
720(9)
720(10)
720(11)
720(12)
720(13)
720(14)
720(15)

图 7B

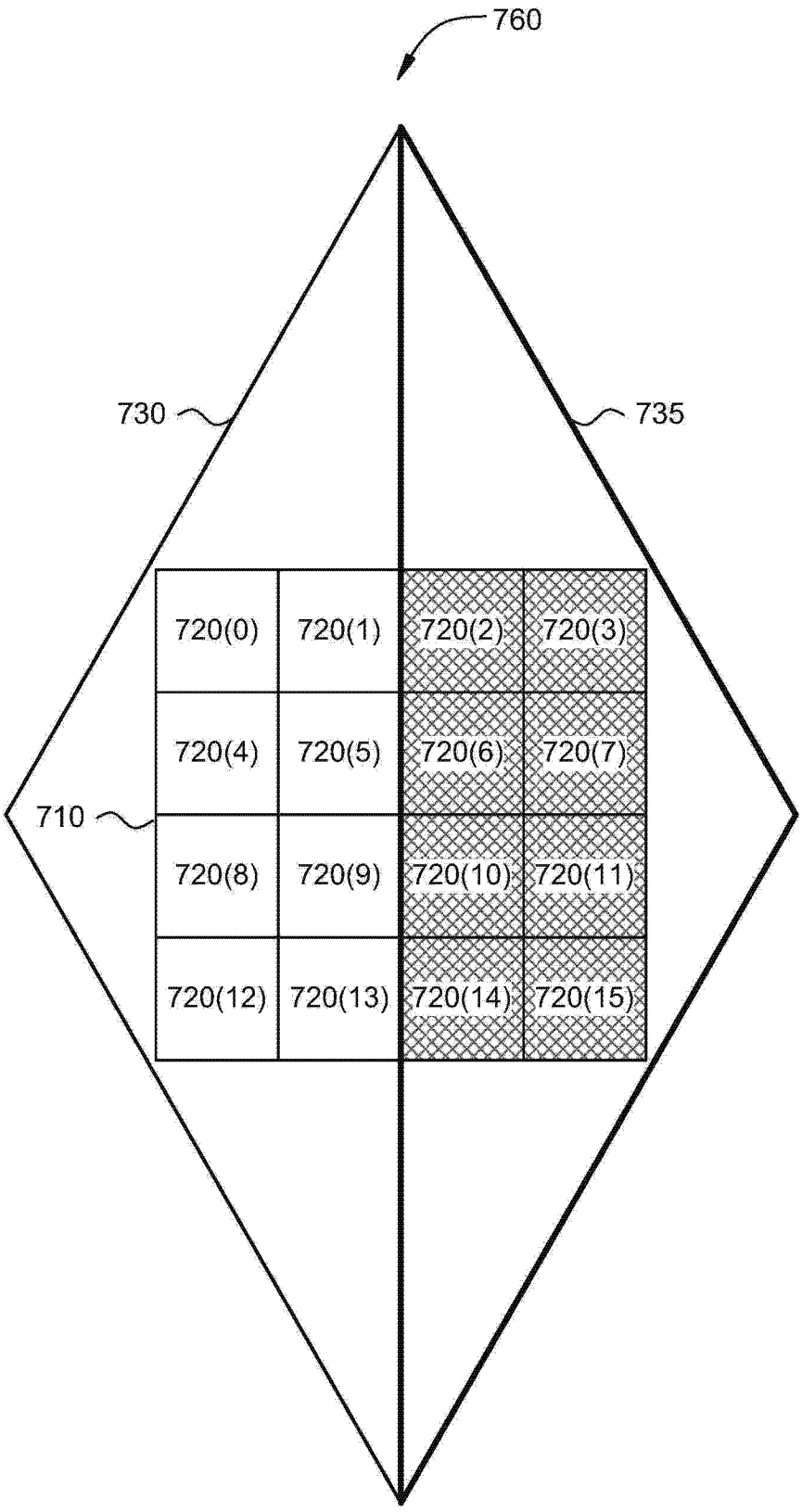

760
730
735
710
720(0)
720(1)
720(2)
720(3)
720(4)
720(5)
720(6)
720(7)
720(8)
720(9)
720(10)
720(11)
720(12)
720(13)
720(14)
720(15)

图 7C

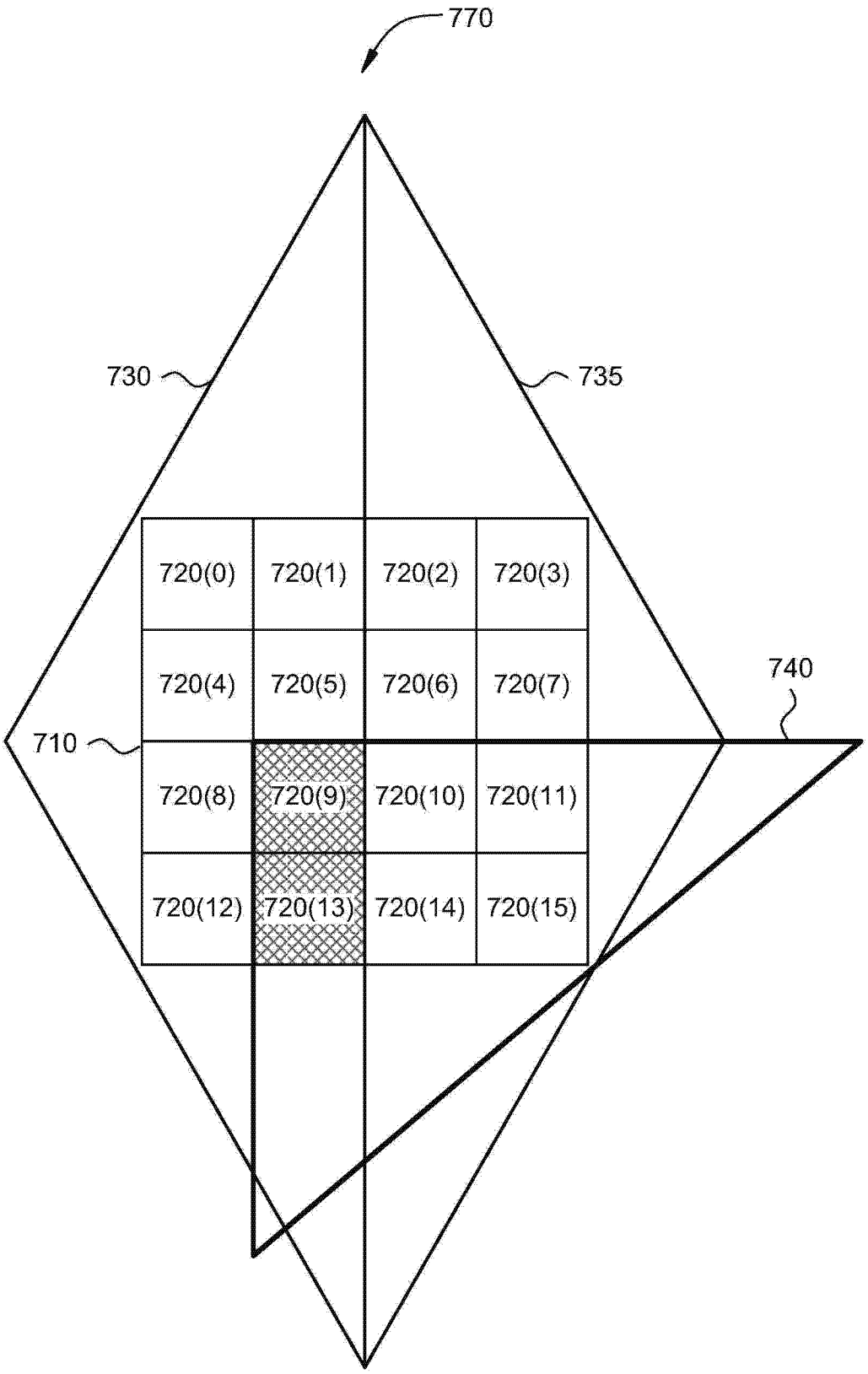

770
730
735
740
710
720(0)
720(1)
720(2)
720(3)
720(4)
720(5)
720(6)
720(7)
720(8)
720(9)
720(10)
720(11)
720(12)
720(13)
720(14)
720(15)

图 7D

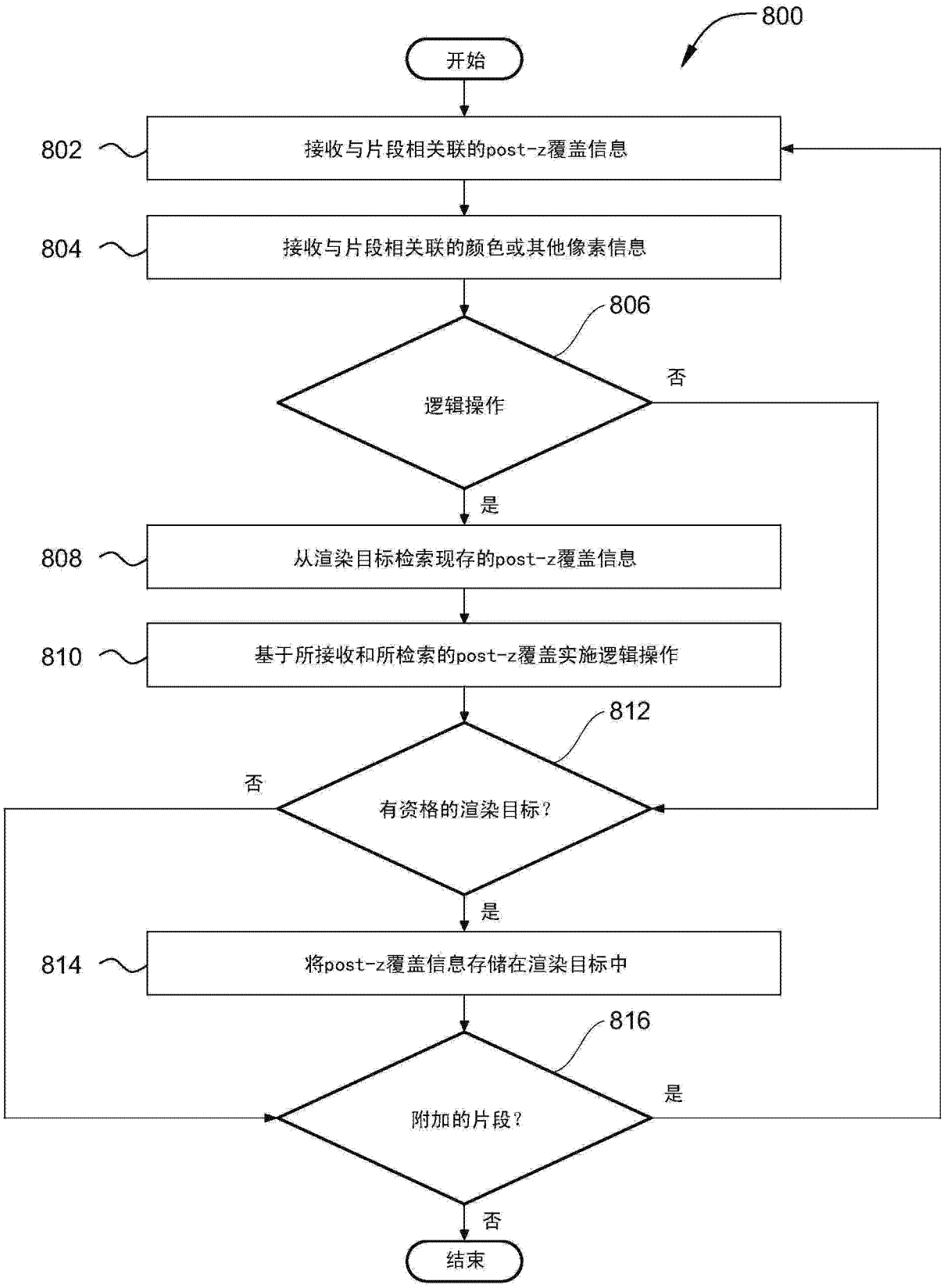

800
开始
802
接收与片段相关联的post-z覆盖信息
804
接收与片段相关联的颜色或其他像素信息
806
逻辑操作
否
是
808
从渲染目标检索现存的post-z覆盖信息
810
基于所接收和所检索的post-z覆盖实施逻辑操作
812
有资格的渲染目标？
否
是
814
将post-z覆盖信息存储在渲染目标中
816
附加的片段？
是
否
结束

图 8