



(12) 发明专利

(10) 授权公告号 CN 108732754 B

(45) 授权公告日 2025. 06. 06

(21) 申请号 201810336612.6
(22) 申请日 2018.04.16
(65) 同一申请的已公布的文献号
申请公布号 CN 108732754 A

(43) 申请公布日 2018.11.02
(30) 优先权数据
15/489,096 2017.04.17 US

(73) 专利权人 英特尔公司
地址 美国加利福尼亚州

(72) 发明人 D·S·范巴 A·桑原
C·萨科斯维尔 R·文卡特拉曼
B·E·英斯科 A·S·卡尔拉
H·拉韦 A·考克 M·阿波达卡
K·肖 J·S·波尔斯
A·T·莱克 D·M·西米尼

B·韦布 E·乌尔德-阿迈德-瓦尔
J·奎亚特科维斯基 P·R·劳斯
A·N·沙阿 A·R·阿普 J·雷
W·付 N·卡布拉斯欧斯
P·萨蒂 B·M·波罗尔

(74) 专利代理机构 上海专利商标事务所有限公
司 31100
专利代理师 李炜 黄嵩泉

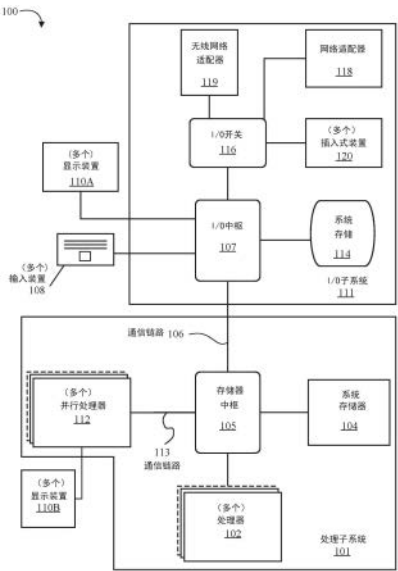
(51) Int.Cl.
G02B 27/01 (2006.01)
G06T 19/00 (2011.01)

(56) 对比文件
CN 105393158 A, 2016.03.09
US 9342929 B2, 2016.05.17

审查员 闫西章
权利要求书3页 说明书57页 附图44页

(54) 发明名称
协作式多用户虚拟现实

(57) 摘要
本申请公开了协作式多用户虚拟现实。图形设备的实施例可以包括:处理器;存储器,通信地耦合至所述处理器;以及协作引擎,通信地耦合至所述处理器以用于识别环境中的两个或更多个用户之间的共享图形分量,并且与所述环境中的所述两个或更多个用户共享所述共享图形分量。所述协作引擎的实施例可以包括集中式共享器、深度共享器、共享预处理器、多端口图形子系统、以及解码共享器中的一个或多个。公开并要求保护了其他实施例。



1. 一种图形设备,包括:
处理器;
存储器,通信地耦合至所述处理器;以及
协作引擎,通信地耦合至所述处理器以用于:
确定第一视点与第二视点之间的视点差,其中,所述第一视点与两个或更多个用户中的第一用户相关联,并且所述第二视点与所述两个或更多个用户中的第二用户相关联;
响应于确定了所述视点差低于阈值,识别可显示对象要作为背景的部分,其中所述可显示对象用于以相同定向且在相同位置处为所述第一用户和所述第二用户渲染;
生成所述背景;以及
与所述第一用户和所述第二用户共享所述背景。
2. 如权利要求1所述的设备,其中,所述协作引擎进一步包括集中式共享器,所述集中式共享器用于:
基于目标虚拟现实设备来分割所述共享图形分量的工作负荷。
3. 如权利要求1所述的设备,其中,所述协作引擎进一步包括深度共享器,所述深度共享器用于:
从独立于所述第一用户的独立源收集深度信息;以及
与所述第一用户共享所述深度信息。
4. 如权利要求3所述的设备,其中,所述深度共享器进一步用于:
从所述第一用户的视场之外的所述独立源收集可见性信息;以及
与所述第一用户共享所述可见性信息。
5. 如权利要求1所述的设备,其中,所述协作引擎进一步包括共享预处理器,所述共享预处理器用于:
预计算与所述共享图形分量有关的信息;以及
与所述两个或更多个用户共享预计算的信息。
6. 如权利要求5所述的设备,其中,所述预计算的信息包括几何信息、图元信息、和物理信息中的一项或多项。
7. 如权利要求1所述的设备,其中,所述协作引擎进一步包括多端口图形子系统,所述多端口图形子系统用于:
在所述多端口图形子系统的每个端口上支持不同的用户。
8. 如权利要求7所述的设备,其中,所述多端口图形子系统进一步用于:
当仅第一虚拟现实设备连接至所述多端口图形子系统时,支持所述第一虚拟现实设备的左眼显示器和右眼显示器;
当所述第一虚拟现实设备和第二虚拟现实设备两者均连接至所述多端口图形子系统时,支持所述第一虚拟现实设备和所述第二虚拟现实设备;并且
当所述第一虚拟现实设备和所述第二虚拟现实设备两者均连接至所述多端口图形子系统时,在所述第一虚拟现实设备与所述第二虚拟现实设备之间共享图形图元。
9. 如权利要求1所述的设备,其中,所述协作引擎进一步包括解码共享器,所述解码共享器用于:
识别所述第一用户的第一视图区与所述第二用户的第二视图区之间的共享场景的重

叠区；

对所述重叠区进行解码；以及

与所述第一用户和所述第二用户两者共享经解码的重叠区。

10. 如权利要求9所述的设备，其中，所述解码共享器进一步用于：

基于所述重叠区对所述解码进行优先级排序。

11. 一种图形协作的方法，所述方法包括：

确定第一视点与第二视点之间的视点差，其中，所述第一视点与两个或更多个用户中的第一用户相关联，并且所述第二视点与所述两个或更多个用户中的第二用户相关联；

响应于确定了所述视点差低于阈值，识别可显示对象要作为要作为背景的部分，其中所述可显示对象用于以相同定向且在相同位置处为所述第一用户和所述第二用户渲染；

生成所述背景；以及

与所述第一用户和所述第二用户共享所述背景。

12. 如权利要求11所述的方法，进一步包括：

基于目标虚拟现实设备来分割所述共享图形分量的工作负荷。

13. 如权利要求11所述的方法，进一步包括：

从独立于所述第一用户的独立源收集深度信息；以及

与所述第一用户共享所述深度信息。

14. 如权利要求13所述的方法，进一步包括：

从所述第一用户的视场之外的所述独立源收集可见性信息；以及

与所述第一用户共享所述可见性信息。

15. 如权利要求11所述的方法，进一步包括：

预计算与所述共享图形分量有关的信息；以及

与所述两个或更多个用户共享预计算的信息。

16. 如权利要求15所述的方法，其中，所述预计算的信息包括几何信息、图元信息、和物理信息中的一项或多项。

17. 如权利要求11所述的方法，进一步包括：

在多端口图形子系统的每个端口上支持不同的用户。

18. 如权利要求17所述的方法，进一步包括：

当仅第一虚拟现实设备连接至所述多端口图形子系统时，支持所述第一虚拟现实设备的左眼显示器和右眼显示器；

当所述第一虚拟现实设备和第二虚拟现实设备两者均连接至所述多端口图形子系统时，支持所述第一虚拟现实设备和所述第二虚拟现实设备；以及

当所述第一虚拟现实设备和所述第二虚拟现实设备两者均连接至所述多端口图形子系统时，在所述第一虚拟现实设备与所述第二虚拟现实设备之间共享图形图元。

19. 如权利要求11所述的方法，进一步包括：

识别所述第一用户的第一视图区与所述第二用户的第二视图区之间的共享场景的重叠区；

对所述重叠区进行解码；以及

与所述第一用户和所述第二用户两者共享经解码的重叠区。

20. 如权利要求19所述的方法,进一步包括:

基于所述重叠区对所述解码进行优先级排序。

21. 一种图形设备,包括:

用于确定第一视点与第二视点之间的视点差的装置,其中,所述第一视点与两个或更多个用户中的第一用户相关联,并且所述第二视点与所述两个或更多个用户中的第二用户相关联;

用于响应于确定了所述视点差低于阈值而识别可显示对象要作为背景的部分的装置,其中所述可显示对象用于以相同定向且在相同位置处为所述第一用户和所述第二用户渲染;

用于将所述背景广播至所有所述两个或更多个用户的装置;以及

用于与所述第一用户和所述第二用户共享所述背景的装置。

22. 如权利要求21所述的设备,进一步包括:

用于基于目标虚拟现实设备来分割所述共享图形分量的工作负荷的装置。

协作式多用户虚拟现实

技术领域

[0001] 实施例总体上涉及数据处理并涉及经由图形处理单元进行的图形处理。更具体地,实施例涉及协作式多用户虚拟现实 (VR)。

背景技术

[0002] 当前的并行图形数据处理包括被开发用于对图形数据执行如例如线性内插、曲面细分、光栅化、纹理映射、深度测试等特定操作的系统和方法。传统上,图形处理器使用固定功能计算单元来处理图形数据;然而,最近,图形处理器的一部分已经变得可编程,从而使这种处理器能够支持用于处理顶点和片段数据的各种各样的操作。图形处理器可以用于各种VR应用中。

附图说明

[0003] 实施例的各种优点将通过阅读以下说明和所附权利要求以及通过参考以下附图而变得为本领域技术人员所显而易见,在附图中:

[0004] 图1是展示计算机系统的框图,所述计算机系统被配置成实现本文描述的实施例的一个或多个方面;

[0005] 图2A到图2D展示了根据实施例的并行处理器部件;

[0006] 图3A到图3B是根据实施例的图形多处理器的框图;

[0007] 图4A到图4F展示了示例性架构,其中多个GPU通信地耦合至多个多核处理器;

[0008] 图5展示根据实施例的图形处理流水线;

[0009] 图6A是根据实施例的电子处理系统的示例的框图;

[0010] 图6B是根据实施例的感测引擎的示例的框图;

[0011] 图6C是根据实施例的聚焦引擎的示例的框图;

[0012] 图6D是根据实施例的运动引擎的示例的框图;

[0013] 图6E是根据实施例的协作引擎的示例的框图;

[0014] 图6F是根据实施例的图形设备的示例的框图;

[0015] 图6G至图6L是根据实施例的图形协作的方法的示例的流程图;

[0016] 图6M是根据实施例的图形设备的另一个示例的框图;

[0017] 图6N是根据实施例的图形协作的方法的另一个示例的流程图;

[0018] 图7A是根据实施例的集中式共享器的示例的框图;

[0019] 图8A是根据实施例的深度共享器的示例的框图;

[0020] 图8B是根据实施例的在AR/VR环境中的用户的示例的示意图;

[0021] 图8C是根据实施例的共享预处理器的示例的框图;

[0022] 图8D是根据实施例的在AR/VR环境中的用户的另一个示例的示意图;

[0023] 图9A是根据实施例的多端口图形子系统的示例的框图;

[0024] 图9B是根据实施例的多端口图形子系统的另一个示例的框图;

- [0025] 图9C是根据实施例的多端口图形子系统的另一个示例的框图；
- [0026] 图10A是根据实施例的共享解码器的示例的框图；
- [0027] 图10B是根据实施例的多端口图形子系统的另一个示例的框图；
- [0028] 图10C是根据实施例的360帧的示例的示意图；
- [0029] 图10D是根据实施例的360帧的示例的另一个示意图；
- [0030] 图11是根据实施例的头戴式显示器 (HMD) 系统的示例的图；
- [0031] 图12是根据实施例包括于图11的HMD系统中的功能部件的示例的框图；
- [0032] 图13是根据实施例包括于并行处理单元中的通用处理集群的示例的框图；
- [0033] 图14是根据实施例可在并行处理单元内实现的图形处理流水线的示例的概念图；
- [0034] 图15是根据实施例的串流多处理器的示例的框图；
- [0035] 图16到图18是根据实施例的数据处理系统的概述的示例的框图；
- [0036] 图19是根据实施例的图形处理引擎的示例的框图；
- [0037] 图20到图22是根据实施例的执行单元的示例的框图；
- [0038] 图23是根据实施例的图形流水线的示例的框图；
- [0039] 图24A到图24B是根据实施例的图形流水线的示例的框图；
- [0040] 图25是根据实施例的图形软件架构的示例的框图；
- [0041] 图26是根据实施例的知识产权 (IP) 核开发系统的示例的框图；以及
- [0042] 图27是根据实施例的片上系统集成电路的示例的框图。

具体实施方式

[0043] 在以下描述中,阐述了许多细节以提供对本公开的更全面的理解。然而对于本领域技术人员显然的是,没有一个或多个这些具体细节也可实践本发明。在其他实例中,未描述公知的特征以便不阻碍本发明。

[0044] 系统概述

[0045] 图1是展示计算系统100的框图,所述计算系统被配置成实现本文描述的实施例的一个或多个方面。计算系统100包括处理子系统101,所述处理子系统具有一个或多个处理器102和系统存储器104,所述处理器与所述系统存储器经由可包括存储器中枢105的互连路径来通信。存储器中枢105可以是芯片组部件内的单独的部件,或可以集成在一个或多个处理器102内。存储器中枢105经由通信链路106与I/O子系统111耦合。I/O子系统111包括I/O中枢107,所述I/O中枢可以使得计算系统100能够从一个或多个输入装置108接收输入。另外,I/O中枢107可以使得显示控制器能够将输出提供给一个或多个显示装置110A,所述显示控制器可被包括在一个或多个处理器102中。在一个实施例中,与I/O中枢107耦合的一个或多个显示装置110A可以包括本地、内部或嵌入式显示装置。

[0046] 在一个实施例中,处理子系统101包括一个或多个并行处理器112,所述并行处理器经由总线或其他通信链路113耦合至存储器中枢105。通信链路113可以是任何数目的基于标准的通信链路技术或协议中的一者(比如但不限于,PCI快速总线),或可以是供应方特定的通信接口或通信结构。在一个实施例中,一个或多个并行处理器112形成计算上集中的并行或向量处理系统,所述系统包括大量处理核和/或处理集群(比如,集成众核(MIC)处理器)。在一个实施例中,一个或多个并行处理器112形成图形处理子系统,所述图形处理子系

统可以将像素输出到经由I/O中枢107耦合的一个或多个显示装置110A中的一者。一个或多个并行处理器112还可以包括显示控制器和显示接口(未示出)以使得能够直接连接到一个或多个显示装置110B。

[0047] 在I/O子系统111内,系统存储单元114可以连接到I/O中枢107以提供用于计算系统100的存储机制。I/O开关116可以用于提供接口机制以实现I/O中枢107与其他部件(比如,可集成到平台中的网络适配器118和/或无线网络适配器119,以及可以经由一个或多个插入式装置120添加的各种其他装置)之间的连接。网络适配器118可以是以以太网适配器或另一有线网络适配器。无线网络适配器119可以包括以下各者中的一者或多者:Wi-Fi、蓝牙、近场通信(NFC)、或包括一个或多个无线电装置的其他网络装置。

[0048] 计算系统100可以包括未明确示出的其他部件,包括USB或其他端口连接件、光学存储驱动器、视频捕获装置等等,它们也可连接到I/O中枢107。使图1中的各种部件互连的通信路径可使用任何合适的协议来实现,比如基于PCI(外围部件互连)的协议(例如,PCI快速总线)或任何其他总线或点对点通信接口和/或协议(比如,NV-Link高速互连、或本领域中已知的互连协议)。

[0049] 在一个实施例中,一个或多个并行处理器112包括针对图形和视频处理进行优化的电路(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,一个或多个并行处理器112包括针对通用处理进行优化的电路,同时保持本文更详细地描述的底层计算架构。在又一个实施例中,计算系统100的部件可与一个或多个其他系统元件一起集成在单个集成电路上。例如,一个或多个并行处理器112、存储器中枢105、处理器102和I/O中枢107可以集成到芯片上系统(SoC)集成电路中。可替代地,计算系统100的部件可以集成到单个封装中以形成封装中系统(SIP)配置。在一个实施例中,计算系统100的部件的至少一部分可以集成到多芯片模块(MCM)中,所述MCM可以与其他多芯片模块一起互连到模块化计算系统中。

[0050] 将认识到的是,本文示出的计算系统100是展示性的,并且各种变型和修改是有可能的。可根据需要修改连接拓扑,包括桥的数目和排、(多个)处理器102的数目和(多个)并行处理器112的数目。例如,在一些实施例中,系统存储器104直接而非通过桥连接到(多个)处理器102,而其他装置经由存储器中枢105和(多个)处理器102与系统存储器104通信。在其他替代性拓扑中,(多个)并行处理器112连接到I/O中枢107或直接连接到一个或多个处理器102中的一者,而非连接到存储器中枢105。在其他实施例中,I/O中枢107和存储器中枢105可集成到单个芯片中。一些实施例可包括经由多个插口附接的两组或更多组处理器102,它们可以与(多个)并行处理器112的两个或更多个实例耦合。

[0051] 本文中示出的一些特定部件是可选的,并且可以不被包括在计算系统100的所有实现方式中。例如,可支持任何数量的插入式卡或外设,或可消除一些部件。此外,一些架构可对于与图1中展示的那些部件类似的部件使用不同的术语。例如,在一些架构中,存储器中枢105可称为北桥,而I/O中枢107可称为南桥。

[0052] 图2A展示了根据实施例的并行处理器200。并行处理器200的各种部件可使用一个或多个集成电路装置来实现,比如可编程处理器、专用集成电路(ASIC)或现场可编程门阵列(FPGA)。根据实施例,所展示的并行处理器200是图1中所示的一个或多个并行处理器112的变体。

[0053] 在一个实施例中,并行处理器200包括并行处理单元202。所述并行处理单元包括I/O单元204,所述I/O单元实现与其他装置(包括并行处理单元202的其他实例)的通信。I/O单元204可直接连接到其他装置。在一个实施例中,I/O单元204经由使用中枢或开关接口(比如,存储器中枢105)来与其他装置连接。存储器中枢105与I/O单元204之间的连接形成通信链路113。在并行处理单元202内,I/O单元204与主机接口206和存储器交叉开关(memory crossbar)216连接,其中,主机接口206接收涉及执行处理操作的命令,并且存储器交叉开关216接收涉及执行存储器操作的命令。

[0054] 当主机接口206经由I/O单元204接收命令缓冲时,主机接口206可以将用于执行那些命令的工作操作导引至前端208。在一个实施例中,前端208与调度器210耦合,该调度器210被配置成将命令或其他工作项目分布至处理集群阵列212。在一个实施例中,调度器210确保在任务被分布至处理集群阵列212的处理集群之前,处理集群阵列212被适当地配置且处于有效状态。在一个实施例中,调度器210是经由在微控制器上执行的固件逻辑实现的。微控制器实现的调度器210可配置用于以粗粒度和细粒度执行复杂的调度和工作分布操作,从而实现在处理阵列212上执行的线程的快速抢占和上下文切换。在一个实施例中,主机软件可以经由多个图像处理门铃中的一个来证明工作负荷以用于在处理阵列212上调度。随后工作负荷可以由调度器微控制器内的调度器210逻辑跨处理阵列212自动地分布。

[0055] 处理集群阵列212可以包括多达“N”个处理集群(例如,集群214A、集群214B、直到集群214N)。处理集群阵列212的每一个集群214A-214N都可以执行大量的并发线程。调度器210可以使用各种调度和/或工作分布算法来将工作分配给处理集群阵列212的集群214A-214N,各种调度和/或工作分布算法可取决于为每一种类型的程序或计算而产生的工作负荷而变化。调度可以由调度器210动态地处置,或者可以在配置用于由处理集群阵列212执行的程序逻辑的编译期间部分地由编译器逻辑辅助。在一个实施例中,可以将处理集群阵列212的不同集群214A-214N分配用于处理不同类型的程序,或用于执行不同类型的计算。

[0056] 可以将处理集群阵列212配置成执行各种类型的并行处理操作。在一个实施例中,将处理集群阵列212配置成执行通用并行计算操作。例如,处理集群阵列212可以包括用于执行处理任务的逻辑,处理任务包括过滤视频和/或音频数据、执行建模操作(包括物理操作)、以及执行数据变换。

[0057] 在一个实施例中,处理集群阵列212被配置成执行并行的图形处理操作。在其中并行处理器200被配置成执行图形处理操作的实施例中,处理集群阵列212可以包括用于支持执行这样的图形处理操作的附加逻辑,包括但不限于用于执行纹理操作的纹理采样逻辑、以及曲面细分逻辑和其他顶点处理逻辑。另外,处理集群阵列212可以被配置成执行与图形处理有关的着色器程序,比如但不限于顶点着色器、曲面细分着色器、几何着色器和像素着色器。并行处理单元202可以经由I/O单元204传递来自系统存储器的数据以供处理。在处理期间,可以将所传递的数据在处理期间存储到芯片上存储器(例如,并行处理器存储器222),然后将其写回到系统存储器。

[0058] 在一个实施例中,当并行处理单元202用于执行图形处理时,调度器210可以被配置成将处理工作负荷划分成近似等规模的任务,以更好地使得能够将图形处理操作分布到处理集群阵列212中的多个集群214A至214N。在一些实施例中,处理集群阵列212的多个部分可以被配置成执行不同类型的处理。例如,第一部分可被配置成执行顶点着色和拓扑生

成,第二部分可被配置成执行曲面细分和几何着色,并且第三部分可被配置成执行像素着色或其他屏幕空间操作,以产生供显示的渲染图像。由集群214A至214N中的一者或多者产生的中间数据可存储在缓冲器中以允许在集群214A至214N之间传输所述中间数据以供进一步处理。

[0059] 在操作期间,处理集群阵列212可以经由调度器210来接收待执行的处理任务,所述调度器从前端208接收定义处理任务的命令。针对图形处理操作,处理任务可以包括待处理的数据(例如,表面(补片(patch))数据、图元数据(primitive data)、顶点数据和/或像素数据)的索引以及状态参数和定义要如何处理数据(例如,要执行什么程序)的命令。调度器210可被配置成获取与任务相对应的索引,或可从前端208接收这些索引。前端208可以被配置成确保在发起由进入的命令缓冲(例如,分批缓冲、推动缓冲等)指定的工作负荷之前处理集群阵列212被配置成有效状态。

[0060] 并行处理单元202的一个或多个实例中的每一个都可以与并行处理器存储器222耦合。并行处理器存储器222可以经由存储器交叉开关216来访问,存储器交叉开关216可以从处理集群阵列212以及I/O单元204接收存储器请求。存储器交叉开关216可以经由存储器接口218访问并行处理器存储器222。存储器接口218可以包括多个分区单元(例如,分区单元220A、分区单元220B、直到分区单元220N),每一个分区单元都可以耦合至并行处理器存储器222的一部分(例如,存储器单元)。在一种实现方式中,将分区单元220A-220N的数量配置成等于存储器单元的数量,使得第一分区单元220A具有对应的第一存储器单元224A,第二分区单元220B具有对应的存储器单元224B,并且第N分区单元220N具有对应的第N存储器单元224N。在其他实施例中,分区单元220A-220N的数量可以不等于存储器装置的数量。

[0061] 在各种实施例中,存储器单元224A至224N可以包括各种类型的存储器装置,包括动态随机存取存储器(DRAM)或图形随机存取存储器(比如,同步图形随机存取存储器(SGRAM),包括图形双数据速率(GDDR)存储器)。在一个实施例中,存储器单元224A至224N还可包括3D堆叠式存储器,包括但不限于高带宽存储器(HBM)。本领域技术人员将认识到,存储器单元224A至224N的具体的实现方式可以变化,并且可以选自各种常规设计中的一者。渲染目标(比如,帧缓冲器或纹理映射(texture map))可跨越存储器单元224A至224N存储,从而允许分区单元220A至220N并行写入每个渲染目标的多个部分以高效地使用并行处理器存储器222的可用带宽。在一些实施例中,可排除并行处理器存储器222的本地实例,以有利于结合本地高速缓存存储器来利用系统存储器的统一的存储器设计。

[0062] 在一个实施例中,处理集群阵列212的集群214A-214N中的任何一个都可以处理将被写入并行处理器存储器222内的存储器单元224A-224N中的任何一个的数据。可以将存储器交叉开关216配置成将每一个集群214A-214N的输出传递到可以对输出执行附加的处理操作的任何分区单元220A-220N或另一集群214A-214N。每一个集群214A-214N都可以通过存储器交叉开关216与存储器接口218通信,以便从各种外部存储器装置读取或向各种外部存储器装置写入。在一个实施例中,存储器交叉开关216具有到存储器接口218的连接以及与I/O单元204通信,并具有到并行处理器存储器222的本地实例的连接,从而使不同的处理集群214A-214N内的处理单元能够与系统存储器或不在并行处理单元202本地的其他存储器通信。在一个实施例中,存储器交叉开关216可以使用虚拟通道以分离集群214A-214N与分区单元220A-220N之间的业务流。

[0063] 虽然在并行处理器200内展示了并行处理单元202的单个实例,但是可以包括并行处理单元202的任何数目的实例。例如,可以在单个插入式卡上提供并行处理单元202的多个实例,或可以将多个插入式卡互连。并行处理单元202的不同实例可以被配置成即使这些不同实例具有不同数目的处理核、不同量的本地并行处理器存储器和/或其他配置差异而仍互操作。例如且在一个实施例中,并行处理单元202的一些实例可以相对于其他实例包括更高精度浮点单元。包括并行处理单元202或并行处理器200的一个或多个实例的系统可以以多种配置和形状因数来实现,包括但不限于台式、膝上型或手持式个人计算机、服务器、工作站、游戏控制台和/或嵌入式系统。

[0064] 图2B是根据实施例的分区单元220的框图。在一个实施例中,分区单元220是图2A的分区单元220A至220N中的一者的实例。如所展示,分区单元220包括L2高速缓存221、帧缓冲器接口225和ROP 226(光栅操作单元)。L2高速缓存221是读/写高速缓存,其被配置成执行从存储器交叉开关216和ROP 226接收的加载和存储操作。由L2高速缓存221将读未命中(read miss)和紧急回写请求输出到帧缓冲器接口225以供处理。也可以经由帧缓冲器接口225将更新发送到帧缓冲器以供处理。在一个实施例中,帧缓冲器接口225与并行处理器存储器中的存储器单元(比如,图2的存储器单元224A至224N(例如,在并行处理器存储器222内))中的一者交界。

[0065] 在图形应用中,ROP 226是执行诸如模板印刷(stencil)、z测试、混合等等的光栅操作的处理单元。随后ROP 226输出存储在图形存储器中的处理过的图形数据。在一些实施例中,ROP 226包括压缩逻辑,该压缩逻辑用于压缩写入到存储器的深度或颜色数据,并且解压缩从存储器读取的深度或颜色数据。压缩逻辑可以是利用多种压缩算法的一种或多种的无损压缩逻辑。由ROP 226执行的压缩的类型可以基于待压缩的数据的统计特性而变化。例如,在一个实施例中, Δ 颜色压缩逐图块地对深度和颜色数据执行。

[0066] 在一些实施例中,ROP 226被包括在每个处理集群(例如,图2的集群214A至214N)内而非包括在分区单元220内。在这样的实施例中,经由存储器交叉开关216来传输针对像素数据的读和写请求而非像素片段数据。已处理的图形数据可在显示装置(比如,图1的一个或多个显示装置110中的一者)上显示、被路由以供由(多个)处理器102进一步处理、或被路由以供由图2A的并行处理器200内的处理实体中的一者进一步处理。

[0067] 图2C是根据实施例的并行处理单元内的处理集群214的框图。在一个实施例中,处理集群是图2的处理集群214A至214N中的一者的实例。处理集群214可以被配置成并行执行许多线程,其中,术语“线程”是指在一组特定的输入数据上执行的特定程序的实例。在一些实施例中,在不提供多个独立的指令单元的情况下,使用单指令多数据(SIMD)指令发布技术以支持对大量线程的并行执行。在其他实施例中,在使用共同指令单元的情况下,使用单指令多线程(SIMT)技术以支持对大量一般为同步的线程的并行执行,所述共同指令单元被配置成将指令发布到处理集群中的每一者内的一组处理引擎。不同于SIMD执行制度(其中,所有处理引擎通常执行相同的指令),SIMT执行允许不同的线程更容易沿着分歧的执行路径通过给定的线程程序。本领域技术人员将理解,SIMD处理制度表示SIMT处理制度的功能性子集。

[0068] 可以经由流水线管理器232来控制处理集群214的操作,所述流水线管理器将处理任务分布到SIMT并行处理器。流水线管理器232从图2的调度器210接收指令,并且经由图形

多处理器234和/或纹理单元236来管理对那些指令的执行。所展示的图形多处理器234是SIMT并行处理器的示例性实例。然而,具有不同架构的各种类型的SIMT并行处理器可被包括在处理集群214内。图形多处理器234的一个或多个实例可以被包括在处理集群214内。图形多处理器234可以处理数据,并且数据交叉开关240可以用于将已处理的数据分布到多个可能的目的地(包括其他着色器单元)中的一者。流水线管理器232可以通过指定待经由数据交叉开关240分布的已处理的数据的目的地来促进已处理的数据的分布。

[0069] 处理集群214内的每一个图形多处理器234都可以包括完全相同的一组功能执行逻辑(例如,算术逻辑单元、加载-存储单元等)。能以流水线方式配置功能执行逻辑,在流水线方式中,在先前的指令完成之前,可发布新指令。功能执行逻辑支持多种多样的操作,包括整数和浮点算术、比较操作、布尔操作、位移位和各种代数函数的计算。在一个实施例中,可以利用同一功能性单元硬件来执行不同的操作,并且可以存在功能单元的任何组合。

[0070] 传输至处理集群214的指令构成线程。跨一组并行处理引擎而执行的一组线程是线程组。线程组对不同的输入数据执行同一程序。可以将线程组内的每一个线程分配给图形多处理器234内的不同的处理引擎。线程组可包括比图形多处理器234内的处理引擎的数量少的线程。当线程组包括比处理引擎的数量少的线程时,处理引擎中的一个或多个在线程组正在被处理的周期期间可以是空闲的。线程组也可包括比图形多处理器234内的处理引擎的数量多的线程。当线程组包括比图形多处理器234内的处理引擎的数量多的线程时,处理可以在连续的时钟周期上执行。在一个实施例中,可在图形多处理器234上并发地执行多个线程组。

[0071] 在一个实施例中,图形多处理器234包括内部高速缓存存储器以执行加载和存储操作。在一个实施例中,图形多处理器234可以放弃内部高速缓存,并且使用处理集群214内的高速缓存存储器(例如,L1高速缓存308)。每个图形多处理器234还有权访问在所有处理集群214当中共享并且可用于在线程之间转移数据的分区单元(例如,图2的分区单元220A至220N)内的L2高速缓存。图形多处理器234还可访问芯片外全局存储器,所述芯片外全局存储器可以包括本地并行处理器存储器和/或系统存储器中的一者或多者。可将在并行处理单元202外部的任何存储器用作全局存储器。多个实施例(其中处理集群214包括图形多处理器234的多个实例)可以共享共同的指令和数据,这些指令和数据可存储在L1高速缓存308中。

[0072] 每个处理集群214可包括MMU 245(存储器管理单元),所述MMU被配置成将虚拟地址映射到物理地址中。在其他实施例中,MMU 245的一个或多个实例可驻留在图2的存储器接口218内。MMU 245包括:一组页表条目(PTE),用于将图块(tile)(更多地讨论分块(tiling))的虚拟地址映射到物理地址;以及可选地高速缓存行索引。MMU 245可包括可驻留在图形多处理器234或L1高速缓存或处理集群214内的地址转换后备缓冲器(TLB)或高速缓存。物理地址经处理以分布表面数据存取局部性,从而允许在分区单元当中实现高效的请求交错。高速缓存行索引可用于确定针对高速缓存行的请求是命中还是未命中。

[0073] 在图形和计算应用中,处理集群214可被配置成使得每个图形多处理器234耦合至纹理单元236以用于执行纹理映射操作,例如确定纹理样本位置、读取纹理数据和过滤纹理数据。根据需要,从内部纹理L1高速缓存(未示出)或在一些实施例中从图形多处理器234内的L1高速缓存读取纹理数据,并且从L2高速缓存、本地并行处理器存储器或系统存储器获

取所述纹理数据。每个图形多处理器234将已处理的任务输出到数据交叉开关240以将已处理的任务提供给另一个处理集群214,以供进一步处理或以经由存储器交叉开关216将已处理的任务存储在L2高速缓存、本地并行处理器存储器或系统存储器中。preROP 242(例如,预光栅操作单元)被配置成从图形多处理器234接收数据、将数据导引到ROP单元,这些ROP单元可与如本文描述的分区单元(例如,图2的分区单元220A至220N)位于一起。preROP 242单元可以执行针对颜色混合的优化、组织像素颜色数据和执行地址转换。

[0074] 将认识到的是,本文描述的核架构是展示性的,并且各种变型和修改是有可能的。任何数目的处理单元(例如,图形多处理器234、纹理单元236、preROP 242等)可被包括在处理集群214内。此外,虽然仅示出了一个处理集群214,但是如本文描述的并行处理单元可以包括处理集群214的任何数目的实例。在一个实施例中,每个处理集群214可以被配置成使用单独的且截然不同的处理单元、L1高速缓存等独立于其他处理集群214来操作。

[0075] 图2D示出了根据一个实施例的图形多处理器234。在这样的实施例中,图形多处理器234与处理集群214的流水线管理器232耦合。图形多处理器234具有执行流水线,包括但不限于:指令高速缓存252、指令单元254、地址映射单元256、寄存器堆258、一个或多个通用图形处理单元(GPGPU)核262和一个或多个加载/存储单元266。GPGPU核262和加载/存储单元266经由存储器和高速缓存互连268与高速缓存存储器272和共享存储器270耦合。

[0076] 在一个实施例中,指令高速缓存252从流水线管理器232接收待执行的指令流。这些指令被高速缓存在指令高速缓存252中,并且由指令单元254分派以供执行。指令单元254可以将指令分派为线程组(例如,线程束),其中线程组的每个线程被指派给GPGPU核262内的一不同执行单元。指令可以通过指定统一地址空间内的地址来访问本地、共享或全局地址空间中的任一者。地址映射单元256可以用于将统一地址空间中的地址转换成可以由加载/存储单元266访问的截然不同的存储器地址。

[0077] 寄存器堆258为图形多处理器324的功能单元提供一组寄存器。寄存器堆258为连接到图形多处理器324的功能单元(例如,GPGPU核262、加载/存储单元266)的数据路径的操作数提供临时存储。在一个实施例中,在这些功能单元中的每一者之间划分寄存器堆258,使得每个功能单元分配有寄存器堆258的专用部分。在一个实施例中,在由图形多处理器324执行的不同线程束之间划分寄存器堆258。

[0078] GPGPU核262可以各自包括浮点单元(FPU)和/或整数算术逻辑单元(ALU),这些FPU和整数ALU用于执行图形多处理器324的指令。根据实施例,GPGPU核262可以在架构上是类似的,或可以在架构上是不同的。例如且在一个实施例中,GPGPU核262的第一部分包括单精度FPU和整数ALU,而GPGPU核的第二部分包括双精度FPU。在一个实施例中,FPU可以针对浮点算术实现IEEE 754-2008标准,或可以实现可变精度浮点算术。图形多处理器324可以另外包括一个或多个固定功能或特殊功能单元以执行特定的功能(比如,复制矩形或像素混合操作)。在一个实施例中,GPGPU核中的一者或多者也可以包括固定或特殊功能逻辑。

[0079] 在一个实施例中,GPGPU核262包括能够对多组数据执行单条指令的SIMD逻辑。在一个实施例中,GPGPU核262可以物理地执行SIMD4、SIMD8和SIMD16指令,并且逻辑地执行SIMD1、SIMD2和SIMD32指令。用于GPGPU核的SIMD指令可以由着色器编译器在编译时生成,或者可以在执行为单程序多数据(SPMD)或SIMT架构编写和编译的程序时自动生成。为SIMT执行模型而配置的程序的多个线程可以经由单条SIMD指令而执行。例如,在一个实施例中,

执行相同或类似操作的八个SIMT线程可以经由单个SIMD8逻辑单元并行地执行。

[0080] 存储器和高速缓存互连268是互连网络,该互连网络将图形多处理器234的功能单元中的每一个连接到寄存器堆258,并连接到共享存储器270。在一个实施例中,存储器和高速缓存互连268是交叉开关互连,该交叉开关互连允许加载/存储单元266在共享存储器270与寄存器堆258之间实现加载和存储操作。寄存器堆258能以与GPGPU核262相同的频率操作,由此在GPGPU核262与寄存器堆258之间的数据传递是非常低等待时间的。共享存储器270可以用来实现在图形多处理器234内的功能单元上执行的线程之间的通信。高速缓存存储器272可以用作例如数据高速缓存,以便对功能单元与纹理单元236之间通信的纹理数据进行高速缓存。共享存储器270也可以用作程序管理的高速缓存。在GPGPU核262上执行的线程能以程序方式还将除了存储在高速缓存存储器272内的经自动高速缓存的数据之外的数据存储在共享存储器内。

[0081] 图3A到图3B展示了根据实施例的附加图形多处理器。所展示的图形多处理器325、350是图2C的图形多处理器234的变体。所展示的图形多处理器325、350可以被配置为能够同时执行大量执行线程的流传送多处理器(SM)。

[0082] 图3A示出了根据附加实施例的图形多处理器325。图形多处理器325相对于图2D的图形多处理器234包括执行资源单元的多个附加实例。例如,图形多处理器325可以包括指令单元332A至332B、寄存器堆334A-334B和纹理单元344A-344B的多个实例。图形多处理器325还包括多组图形或计算执行单元(例如,GPGPU核336A至336B、GPGPU核337A至337B、GPGPU核338A至338B)和多组加载/存储单元340A至340B。在一个实施例中,执行资源单元具有共同的指令高速缓存330、纹理和/或数据高速缓存存储器342以及共享存储器346。

[0083] 各种部件可以经由互连结构327通信。在一个实施例中,互连结构327包括一个或多个交叉开关以启用图形多处理器325的各种部件之间的通信。在一个实施例中,互连结构327是分开的高速网络结构层,图形多处理器325的每一个部件堆叠在该高速网络结构层上。图形多处理器325的部件经由互连结构327与远程部件通信。例如,GPGPU核336A-336B、337A-337B以及338A-338B可以各自经由互连结构327与共享存储器346通信。互连结构327可以仲裁图形多处理器325内的通信以确保部件之间的公平的带宽分配。

[0084] 图3B示出了根据附加实施例的图形多处理器350。图形处理器包括多组执行资源356A至356D,其中,每一组执行资源包括多个指令单元、寄存器堆、GPGPU核和加载存储单元,如图2D和图3A中所展示。执行资源356A至356D可以与纹理单元360A至360D一致地工作以进行纹理操作,同时共享指令高速缓存354和共享存储器362。在一个实施例中,执行资源356A至356D可以共享指令高速缓存354和共享存储器362以及纹理和/或数据高速缓存存储器358A至358B的多个实例。各种部件可以经由类似于图3A的互连结构327的互连结构352来通信。

[0085] 本领域技术人员将理解,图1、图2A至图2D以及图3A至图3B中所描述的架构就本实施例的范围而言是描述性的和非限制性的。因此,在不背离本文描述的实施例的范围的情况下,本文描述的技术可在任何正确配置的处理单元上实现,所述处理单元包括但不限于一个或多个移动应用处理器、一个或多个台式计算机或服务器中央处理单元(CPU)(包括多核CPU)、一个或多个并行处理单元(比如,图2的并行处理单元202)以及一个或多个图形处理器或专用处理单元。

[0086] 在一些实施例中,如本文描述的并行处理器或GPGPU通信地耦合至主机/处理器核以加速图形操作、机器学习操作、模式分析操作和各种通用GPU(GPGPU)功能。GPU可经由总线或其他互连(例如,比如PCIe或NVLink的高速互连)通信地耦合至主机处理器/核。在其他实施例中,GPU可集成在与这些核相同的封装或芯片上,并且经由内部的处理器总线/互连(即,在所述封装或芯片的内部)通信地耦合至这些核。不管连接GPU的方式如何,处理器核都可用工作描述符中所包含的命令/指令序列的形式将工作分配给GPU。GPU然后使用专用的电路/逻辑来高效地处理这些命令/指令。

[0087] 用于GPU至主机处理器互连的技术

[0088] 图4A展示了示例性架构,其中多个GPU 410至413经由高速链路440至443(例如,总线、点对点互连等)通信地耦合至多个多核处理器405至406。在一个实施例中,取决于实现方式,高速链路440至443支持4GB/s、30GB/s、80GB/s或更高的通信吞吐量。可使用各种互连协议,包括但不限于PCIe 4.0或5.0以及NVLink 2.0。然而,本发明的基本原理并不限于任何特定的通信协议或吞吐量。

[0089] 另外,在一个实施例中GPU 410至413中的两者或更多者经由高速链路444至445互连,这些高速链路可使用与用于高速链路440至443的协议/链路相同或不同的协议/链路来实现。类似地,多核处理器405至406中的两者或更多者可经由高速链路433相连接,所述高速链路可以是以20GB/s、30GB/s、120GB/s或更高操作的对称多处理器(SMP)总线。可替代地,图4A中所示的各种系统部件之间的所有通信可使用相同的协议/链路(例如,经由共同的互连结构)来实现。然而,如所提到,本发明的基本原理并不限于任何特定类型的互连技术。

[0090] 在一个实施例中,每个多核处理器405至406分别经由存储器互连430至431通信地耦合至处理器存储器401至402,并且每个GPU 410至413分别经由GPU存储器互连450至453通信地耦合至GPU存储器420至423。存储器互连430至431以及450至453可利用相同或不同的存储器访问技术。通过示例的方式且不受限制地,处理器存储器401至402和GPU存储器420至423可以是易失性存储器,比如动态随机存取存储器(DRAM)(包括堆叠式DRAM)、图形DDR SDRAM(GDDR)(例如,GDDR5、GDDR6)或高带宽存储器(HBM),和/或可以是非易失性存储器,比如3D XPoint或纳米随机存取存储器。在一个实施例中,存储器的某一部分可以是易失性存储器,并且另一部分可以是非易失性存储器(例如,使用两级存储器(2LM)层级结构)。

[0091] 如下文所描述,虽然各种处理器405至406和GPU 410至413可以分别物理地耦合至特定的存储器401至402、420至423,但是可实现统一存储器架构,其中相同的虚拟系统地址空间(也称为“有效地址”空间)被分布在所有各个物理存储器当中。例如,处理器存储器401至402可各自包括64GB的系统存储器地址空间,并且GPU存储器420至423可各自包括32GB的系统存储器地址空间(在本示例中产生总共256GB的可寻址存储器)。

[0092] 图4B展示了根据一个实施例的针对多核处理器407与图形加速模块446之间的互连的附加细节。图形加速模块446可包括集成在线卡上的一个或多个GPU芯片,所述线卡经由高速链路440耦合至处理器407。可替代地,图形加速模块446可集成在与处理器407相同的封装或芯片上。

[0093] 所展示的处理器407包括多个核460A至460D,每个核具有转换后备缓冲器

(translation lookaside buffer) 461A至461D和一个或多个高速缓存462A至462D。这些核可包括用于执行指令和处理数据的各种其他部件,未展示这些部件以避免使本发明的基本原理模糊(例如,指令获取单元、分支预测单元、解码器、执行单元、重排序缓冲器等)。高速缓存462A至462D可包括1级(L1)和2级(L2)高速缓存。另外,一个或多个共享的高速缓存426可被包括在缓存层级结构中并且由多组核460A至460D共享。例如,处理器407的一个实施例包括24个核,每个核具有其自身的L1高速缓存、12个共享的L2高速缓存和12个共享的L3高速缓存。在本实施例中,L2和L3高速缓存中的一者由两个邻近的核共享。处理器407和图形加速器集成模块446与系统存储器441连接,所述系统存储器可包括处理器存储器401至402。

[0094] 经由通过一致性总线464上的核间通信来保持存储在各种高速缓存462A至462D、456和系统存储器441中的数据和指令的一致性。例如,每个高速缓存可具有与其相关联的高速缓存一致性逻辑/电路以响应于检测到的至特定高速缓存行的读或写来经由一致性总线464进行通信。在一个实现方式中,经由一致性总线464来实现高速缓存监听协议,以监听高速缓存访问。高速缓存监听/一致性技术是为本领域技术人员所充分理解的,并且此处将不加以详细描述以避免使本发明的基本原理模糊。

[0095] 在一个实施例中,代理电路425将图形加速模块446通信地耦合至一致性总线464,从而允许图形加速模块446作为核的对等物来参与到高速缓存一致性协议中。特定地,接口435经由高速链路440(例如,PCIe总线、NVLink等)提供至代理电路425的连接性,并且接口437将图形加速模块446连接到链路440。

[0096] 在一个实现方式中,加速器集成电路436代表图形加速模块446的多个图形处理引擎431、432、N来提供高速缓存管理、存储器访问、上下文管理和中断管理服务。图形处理引擎431、432、N可各自包括单独的图形处理单元(GPU)。可替代地,图形处理引擎431、432、N可包括GPU内的不同类型的图形处理引擎,比如图形执行单元、媒体处理引擎(例如,视频编码器/解码器)、采样器和位块传输(blit)引擎。换句话说,图形加速模块可以是具有多个图形处理引擎431至432、N的GPU,或图形处理引擎431至432、N可以是集成在共同的封装、线卡或芯片上的个体GPU。

[0097] 在一个实施例中,加速器集成电路436包括存储器管理单元(MMU) 439以用于执行各种存储器管理功能,比如虚拟至物理存储器转换(也称为有效至真实存储器转换)和用于访问系统存储器441的存储器访问协议。MMU 439还可包括转换后备缓冲器(TLB)(未示出)以用于将虚拟/有效缓存到物理/真实地址转换。在一个实现方式中,高速缓存438存储命令和数据以供由图形处理引擎431至432、N进行有效访问。在一个实施例中,存储在高速缓存438和图形存储器433至434、N中的数据与核高速缓存462A至462D、456以及系统存储器411保持一致。如所提到,这可经由代理电路425来实现,所述代理电路代表高速缓存438和存储器433至434、N来参与到高速缓存一致性机制中(例如,将与对处理器高速缓存462A至462D、456上的高速缓存行的修改/访问有关的更新发送到高速缓存438,和从高速缓存438接收更新)。

[0098] 一组寄存器445存储用于由图形处理引擎431至432、N执行的线程的上下文数据,并且上下文管理电路448管理线程上下文。例如,上下文管理电路448可在上下文切换期间执行保存和恢复操作以保存和恢复各种线程的上下文(例如,其中,第一线程被保存并且第

二线程被存储,使得可以由图形处理引擎执行第二线程)。例如,在上下文切换时,上下文管理电路448可将当前寄存器值存储到存储器中的指定的区域(例如,由上下文指针标识)。然后,其可在返回到上下文时恢复这些寄存器值。在一个实施例中,中断管理电路447接收并处理从系统装置接收到的中断。

[0099] 在一个实现方式中,由MMU 439将来自图形处理引擎431的虚拟/有效地址转换成系统存储器411中的真实/物理地址。加速器集成电路436的一个实施例支持多个(例如,4、8、16个)图形加速器模块446和/或其他加速器装置。图形加速器模块446可专用于在处理器407上执行的单个应用,或可在多个应用之间共享。在一个实施例中,呈现虚拟化图形执行环境,其中多个应用或虚拟机器 (VM) 共享图形处理引擎431至432、N的资源。这些资源可被再分为“切片(slice)”,这些切片基于与不同的VM和/或应用相关联的处理要求和优先权来被分配给这些VM和/或应用。

[0100] 因此,加速器集成电路起至图形加速模块446的系统的桥的作用,并且提供地址转换和系统存储器高速缓存服务。另外,加速器集成电路436可为主机处理器提供虚拟化设施,以管理图形处理引擎的虚拟化、中断和存储器管理。

[0101] 由于图形处理引擎431至432、N的硬件资源被显式地映射到由主机处理器407所见的真实地址空间,所以任何主机处理器都可以使用有效地址值来直接寻址这些资源。在一个实施例中,加速器集成电路436的一个功能是物理地分离图形处理引擎431至432、N,使得它们向系统呈现为独立的单元。

[0102] 如所提到,在所展示的实施例中,一个或多个图形存储器433至434、M分别耦合至图形处理引擎431至432、N中的每一者。图形存储器433至434、M存储由图形处理引擎431至432、N中的每一者处理的指令和数据。图形存储器433至434、M可以是易失性存储器,比如DRAM(包括堆叠式DRAM)、GDDR存储器(例如,GDDR5、GDDR6)或HBM,和/或可以是非易失性存储器,比如3D XPoint或纳米随机存取存储器(Nano-Ram)。

[0103] 在一个实施例中,为减少链路440上的数据业务,使用偏置技术以确保存储在图形存储器433至434、M中的数据是将被图形处理引擎431至432、N使用最频繁并且优选地不被核460A至460D使用(至少不是频繁地)的数据。类似地,偏置机制试图将由核(且优选地不是由图形处理引擎431至432、N)所需的数据保存在这些核的高速缓存462A至462D、456和系统存储器411内。

[0104] 图4C展示了另一个实施例,其中加速器集成电路436被集成在处理器407内。在本实施例中,图形处理引擎431至432、N经由接口437和接口435(再次,这些接口可利用任何形式的总线或接口协议)通过高速链路440来直接通信到加速器集成电路436。加速器集成电路436可执行与关于图4B所描述的操作相同的操作,但考虑到其与一致性总线462和高速缓存的462A至462D、426极为接近而潜在地处于更高的吞吐量。

[0105] 一个实施例支持不同的编程模型,包括专用进程编程模型(没有图形加速模块虚拟化)和共享的编程模型(具有虚拟化)。后者可包括由加速器集成电路436控制的编程模型和由图形加速模块446控制的编程模型。

[0106] 在专用进程模型的一个实施例中,图形处理引擎431至432、N在单一操作系统下专用于单个应用或进程。单个应用可以将其他应用请求汇集(funnel)到图形引擎431至432、N,从而在VM/分区内提供虚拟化。

[0107] 在专用进程编程模型中,可由多个VM/应用分区共享图形处理引擎431至432、N。共享的模型需要系统管理程序虚拟化图形处理引擎431至432、N,以允许由每个操作系统进行访问。针对无管理程序的单分区系统,图形处理引擎431至432、N由操作系统所拥有。在两种情况下,操作系统可以虚拟化图形处理引擎431至432、N以提供对每个进程或应用的访问。

[0108] 针对共享的编程模型,图形加速模块446或个别图形处理引擎431至432、N使用进程句柄来选择进程要素。在一个实施例中,进程要素存储在系统存储器411中,并且可使用本文描述的有效地址至真实地址转换技术来寻址。进程句柄可以是在向图形处理引擎431至432、N登记其上下文(也就是说,调用系统软件以将进程要素添加到进程要素链表)时被提供给主机进程的特定于实现方式的值。进程句柄的较低的16位可以是进程要素链表内的进程要素的偏移。

[0109] 图4D展示了示例性加速器集成切片490。如本文所使用,“切片”包括加速器集成电路436的处理资源的指定部分。系统存储器411内的应用有效地址空间482存储进程要素483。在一个实施例中,响应于来自处理器407上执行的应用480的GPU调用481来存储进程要素483。进程要素483包含对应的应用480的进程状态。包含在进程要素483中的工作描述符(WD) 484可以是由应用请求的单作业,或可包含指向作业队列的指针。在后一种情况下,WD 484是指向应用的地址空间482中的作业请求队列的指针。

[0110] 图形加速模块446和/或个别图形处理引擎431至432、N可以由系统中的所有进程或进程子集共享。本发明的实施例包括用于设置进程状态并将WD 484发送到图形加速模块446以在虚拟化环境中开始作业的基础设施。

[0111] 在一个实现方式中,专用进程编程模型是特定于实现方式的。在这个模型中,单进程拥有图形加速模块446或个别图形处理引擎431。由于图形加速模块446由单进程所拥有,所以在指派图形加速模块446的时候,管理程序针对拥有的分区初始化加速器集成电路436并且操作系统针对拥有的进程初始化加速器集成电路436。

[0112] 在操作中,加速器集成切片490中的WD获取单元491获取下一个WD 484,所述下一个WD包括待由图形加速模块446的图形处理引擎中的一者完成的工作的指示。来自WD 484的数据可存储在寄存器445中,并且由如所展示的MMU 439、中断管理电路447和/或上下文管理电路446使用。例如,MMU 439的一个实施例包括用于访问OS虚拟地址空间485内的段/页表486的段/页行走电路。中断管理电路447可处理从图形加速模块446接收的中断事件492。当执行图形操作时,由MMU 439将由图形处理引擎431至432、N产生的有效地址493转换为真实地址。

[0113] 在一个实施例中,针对每个图形处理引擎431至432、N和/或图形加速模块446复制一组相同的寄存器445,并且可由管理程序或操作系统来初始化这些寄存器。这些所复制的寄存器中的每一者可被包括在加速器集成切片490中。表1中示出了可由管理程序初始化的示例性寄存器。

[0114] 表1-管理程序初始化的寄存器

[0115]

1	切片控制寄存器
2	真实地址 (RA) 调度的进程区域指针
3	权限掩码覆盖寄存器 (Authority Mask Override Register)
4	中断向量表条目偏移

5	中断向量表条目限制
6	状态寄存器
7	逻辑分区ID
8	真实地址 (RA) 管理程序加速器利用记录指针
9	存储描述寄存器

[0116] 表2中示出了可由操作系统初始化的示例性寄存器。

[0117] 表2-操作系统初始化的寄存器

1	进程和线程标识
2	有效地址 (EA) 上下文保存/恢复指针
3	虚拟地址 (VA) 加速器利用记录指针
4	虚拟地址 (VA) 存储段表指针
5	权限掩码
6	工作描述符

[0119] 在一个实施例中,每个WD 484是特定于特定的图形加速模块446和/或图形处理引擎431至432、N的。其包含图形处理引擎431至432、N完成其工作所需的全部信息,或其可以是指向存储器位置(在所述存储器位置处,应用已设置了待完成的工作的命令队列)的指针。

[0120] 图4E展示了共享模型的一个实施例的附加细节。本实施例包括其中存储有进程要素列表499的管理程序真实地址空间498。管理程序真实地址空间498可经由管理程序496来访问,所述管理程序虚拟化用于操作系统495的图形加速模块引擎。

[0121] 共享的编程模型允许来自系统中的所有分区或分区子集的所有进程或进程子集使用图形加速模块446。存在两个编程模型,其中,图形加速模块446由多个进程和分区共享:时间切片共享和图形定向共享(graphics directed shared)。

[0122] 在这个模型中,系统管理程序496拥有图形加速模块446,并且使其功能可用于所有操作系统495。为使图形加速模块446支持由系统管理程序496进行的虚拟化,图形加速模块446可遵循以下要求:1) 应用的作业请求必须是自主的(即,无需在作业之间保持状态),或图形加速模块446必须提供上下文保存和恢复机制。2) 由图形加速模块446保证在指定的时间量内完成应用的作业请求(包括任何转换故障),或图形加速模块446提供抢占作业的处理的能力。3) 当在定向共享的编程模型中操作时,必须保证图形加速模块446在进程之间的公平性。

[0123] 在一个实施例中,针对共享模型,需要应用480利用图形加速模块446类型、工作描述符(WD)、权限掩码寄存器(AMR)值和上下文保存/恢复区域指针(CSRP)来进行操作系统495系统调用。图形加速模块446类型描述了用于系统调用的目标加速度函数。图形加速模块446类型可以是特定于系统的值。WD专门针对图形加速模块446被格式化,并且可以呈图形加速模块446命令、指向用户定义的结构的有效地址指针、指向命令队列的有效地址指针或用于描述待由图形加速模块446完成的工作的任何其他数据结构的形式。在一个实施例中,AMR值是待用于当前进程的AMR状态。被传递到操作系统的值类似于设定AMR的应用。如果加速器集成电路436和图形加速模块446实现方式不支持用户权限掩码覆盖寄存器(UAMOR),那么操作系统可将当前UAMOR值应用于AMR值,之后在管理程序调用中传递AMR。可

选地,管理程序496可应用当前权限掩码覆盖寄存器 (AMOR) 值,之后将AMR放到进程要素483中。在一个实施例中,CSRP是寄存器445中的一者,其包含在应用的地址空间482中的区域的有效地址以用于使图形加速模块446保存和恢复上下文状态。如果不需要在作业之间保存状态或当作业被抢占时,这个指针是可选的。上下文保存/恢复区域可以是固定的 (pinned) 系统存储器。

[0124] 在接收到系统调用时,操作系统495可验证应用480已注册并且已被给予使用图形加速模块446的权限。然后,操作系统495利用表3中所示的信息来调用管理程序496。

[0125] 表3-OS至管理程序调用参数

[0126]	1	工作描述符 (WD)
	2	权限掩码寄存器 (AMR) 值 (潜在地被掩码)
	3	有效地址 (EA) 上下文保存/恢复区域指针 (CSRP)
	4	进程ID (PID) 和可选线程ID (TID)
	5	虚拟地址 (VA) 加速器利用记录指针 (AURP)
	6	存储段表指针 (SSTP) 的虚拟地址
	7	逻辑中断服务号 (LISN)

[0127] 在接收到管理程序调用时,管理程序496验证操作系统495已注册并且已被给予使用图形加速模块446的权限。然后,管理程序496将进程要素483放入对应的图形加速模块446类型的进程要素链表中。进程要素可包括表4中所示的信息。

[0128] 表4-进程要素信息

[0129]	1	工作描述符 (WD)
	2	权限掩码寄存器 (AMR) 值 (潜在地被掩码)
	3	有效地址 (EA) 上下文保存/恢复区域指针 (CSRP)
	4	进程ID (PID) 和可选线程ID (TID)
	5	虚拟地址 (VA) 加速器利用记录指针 (AURP)
	6	存储段表指针 (SSTP) 的虚拟地址
	7	逻辑中断服务号 (LISN)
	8	从管理程序调用参数导出的中断向量表
	9	状态寄存器 (SR) 值
	10	逻辑分区ID (LPID)
	11	真实地址 (RA) 管理程序加速器利用记录指针
	12	存储器描述符寄存器 (SDR)

[0130] 在一个实施例中,管理程序初始化多个加速器集成切片490寄存器445。

[0131] 如图4F中所展示,本发明的一个实施例采用可经由共同的虚拟存储器地址空间寻址的统一存储器,所述共同的虚拟存储器地址空间用于访问物理处理器存储器401至402和GPU存储器420至423。在这种实现方式中,在GPU 410至413上执行的操作利用相同的虚拟/有效存储器地址空间来访问处理器存储器401至402且反之亦然,由此简化可编程性。在一个实施例中,虚拟/有效地址空间的第一部分被分配给处理器存储器401,第二部分被分配给第二处理器存储器402,第三部分被分配GPU存储器420,等。由此得以跨越处理器存储器401至402和GPU存储器420至423中的每一者来分布整个虚拟/有效存储器空间 (有时称为有

效地址空间),从而允许任何处理器或GPU利用被映射到所述存储器的虚拟地址来访问任何物理存储器。

[0132] 在一个实施例中,在MMU 439A至439E中的一者或多者内的偏置/一致性管理电路494A至494E确保主机处理器(例如,405)与GPU 410至413的高速缓存器之间的高速缓存一致性,并且实现指示其中应存储有某些类型的数据的物理存储器的偏置技术。虽然图4F中展示了偏置/一致性管理电路494A至494E的多个实例,但是可在一个或多个主机处理器405的MMU内和/或在加速器集成电路436内实现偏置/一致性电路。

[0133] 一个实施例允许GPU附加存储器420至423被映射为系统存储器的一部分并且使用共享虚拟存储器(SVM)技术来访问,但经受与完全系统高速缓存一致性相关联的典型性能缺陷。GPU附加存储器420至423被作为系统存储器来访问而无繁重的高速缓存一致性开销的这种能力为GPU卸载提供了有益的操作环境。这种安排允许主机处理器405软件设置操作数和访问计算结果,而没有传统I/O DMA数据复制的开销。这样的传统的复制涉及驱动器调用、中断和存储器映射I/O(MMIO)访问,它们相对于简单的存储器访问来说全部都是低效率的。同时,访问GPU附加存储器420至423而无高速缓存一致性开销的能力对于已卸载的计算的执行时间来说可以是关键的。在具有实质流传送写存储器业务的情况下,例如,高速缓存一致性开销可以显著减少由GPU 410至413所见的有效写带宽。操作数设置的效率、结果访问的效率和GPU计算的效率在确定GPU卸载的有效性中全部都起到一定的作用。

[0134] 在一个实现方式中,由偏置跟踪器数据结构来驱动在GPU偏置与主机处理器偏置之间的选择。可使用偏置表,例如,其可以是每GPU附加存储器页包括1或2个位的页粒度结构(即,被控制在存储器页的粒度下)。可以采用一个或多个GPU附加存储器420至423的被偷取的存储器范围来实现偏置表,其中在GPU 410至413中具有或不具有偏置高速缓存(例如,用于缓存偏置表的频繁使用/最近使用的条目)。可替代地,可将整个偏置表保持在GPU内。

[0135] 在一个实现方式中,在实际访问GPU存储器之前访问与每一次访问GPU附加存储器420至423相关联的偏置表条目,从而引起以下操作。首先,来自GPU 410至413的在GPU偏置中寻找其页的本地请求(这些本地请求发现它们的页处于GPU偏置)被直接转发到对应的GPU存储器420至423。来自GPU的本地请求(这些本地请求发现它们的页处于主机偏置)被转发到处理器405(例如,经由如上文所讨论的高速链路)。在一个实施例中,来自处理器405的在主机处理器偏置中寻找所请求的页的请求完成类似于正常存储器读取的请求。可替代地,可将针对GPU偏置页的请求转发到GPU 410至413。然后,如果GPU当前不使用页,那么其可将所述页转变到主机处理器偏置。

[0136] 可以由基于软件的机制、硬件辅助的基于软件的机制抑或针对一组有限的情况由纯粹基于硬件的机制来改变页的偏置状态。

[0137] 用于改变偏置状态的一个机制采用API调用(例如,OpenCL),所述API调用转而调用GPU的装置驱动器,所述装置驱动器转而发送消息(或为命令描述符排队)到GPU,从而指导其改变偏置状态并且针对一些转变在主机中执行高速缓存转储清除(cache flushing)操作。高速缓存转储清除操作对于从主机处理器405偏置转变到GPU偏置来说是需要的,但对于反向转变来说是不需要的。

[0138] 在一个实施例中,通过暂时渲染不可由主机处理器405缓存的GPU偏置页来保持高速缓存一致性。为了访问这些页,处理器405可请求来自GPU410的访问,这取决于实现方式

可立即授予访问权或可不立即授予访问权。因此,为减少处理器405与GPU 410之间的通信,确保GPU偏置页是为GPU所需但非为主机处理器405所需(且反之亦然)的那些页是有利的。

[0139] 图形处理流水线

[0140] 图5展示根据实施例的图形处理流水线500。在一个实施例中,图形处理器可以实现所展示的图形处理流水线500。所述图形处理器可以被包括在如本文描述的并行处理子系统内,所述并行处理子系统为比如图2的并行处理器200,在一个实施例中,其是图1的(多个)并行处理器112的变体。各种并行处理系统可以经由如本文描述的并行处理单元(例如,图2的并行处理单元202)的一个或多个实例来实现图形处理流水线500。例如,着色器单元(例如,图3的图形多处理器234)可被配置成执行顶点处理单元504、曲面细分控制处理单元508、曲面细分评估处理单元512、几何处理单元516和片段/像素处理单元524中的一者或多者的功能。数据组装器502、图元组装器506、514、518、曲面细分单元510、光栅化器522和光栅操作单元526的功能也可由处理集群(例如,图3的处理集群214)内的其他处理引擎和对应的分区单元(例如,图2的分区单元220A至220N)执行。还可使用用于一个或多个功能的专用处理单元来实现图形处理流水线500。在一个实施例中,可以由通用处理器(例如,CPU)内的并行处理逻辑来执行图形处理流水线500的一个或多个部分。在一个实施例中,图形处理流水线500的一个或多个部分可以经由存储器接口528来访问芯片上存储器(例如,如图2中的并行处理器存储器222),所述存储器接口可以是图2的存储器接口218的实例。

[0141] 在一个实施例中,数据组装器502是收集表面和图元的顶点数据的处理单元。数据组装器502随后将包括顶点属性的顶点数据输出至顶点处理单元504。顶点处理单元504是可编程执行单元,该可编程执行单元执行顶点着色器程序,按顶点着色器程序所指定来照亮并变换顶点数据。顶点处理单元504读取存储在高速缓存、本地或系统存储器中的数据以供在处理顶点数据时使用,并且顶点处理单元504可被编程为将顶点数据从基于物体的坐标表示变换到世界空间坐标空间或归一化装置坐标空间。

[0142] 图元组装器506的第一实例从顶点处理单元504接收顶点属性。图元组装器506按照需要读取所存储的顶点属性,并且构建图形图元以供由曲面细分控制处理单元508处理。图形图元包括由各种图形处理应用编程接口(API)所支持的三角、线段、点、补片(patch)等等。

[0143] 曲面细分控制处理单元508将输入顶点视为用于几何补片的控制点。控制点是来自补片的输入表示(例如,补片的基底)转换到适于由曲面细分评估处理单元512在表面评估中使用的表示。曲面细分控制处理单元508也可以计算几何补片的边缘的曲面细分因数。曲面细分因数应用于单个边缘,并且对与该边缘相关联的依赖于视图的细节等级进行量化。将曲面细分单元510配置成接收补片的边缘的曲面细分因数,并且将补片曲面细分为诸如线、三角或四边形图元的多个几何图元,多个几何图元被传输到曲面细分评估处理单元512。曲面细分评估处理单元512对经再分的补片的参数化坐标操作以生成与几何图元相关联的每一个顶点的表面表示和顶点属性。

[0144] 图元组装器514的第二实例从曲面细分评估处理单元512接收顶点属性,按照需要读取所存储的顶点属性,并且构建图形图元以供由几何处理单元516处理。几何处理单元516是可编程执行单元,该可编程执行单元执行几何着色器程序以便按几何着色器程序所指定来变换从图元组装器514接收的图形图元。在一个实施例中,将几何处理单元516编程

为将图形图元再分成一个或多个新图形图元,并且计算用于对新图形图元进行光栅化的参数。

[0145] 在一些实施例中,几何处理单元516可在几何流中增加或删除元素。几何处理单元516将指定新图形图元的参数和顶点输出到图元组装器518。图元组装器518从几何处理单元516接收参数和顶点,并且构建用于由视口缩放、拣选和剪辑单元520处理的图形图元。几何处理单元516读取存储在并行处理器存储器或系统存储器中的数据以供在处理几何数据时使用。视口缩放、拣选和剪辑单元520执行剪辑、拣选和视口缩放,并将经处理的图形图元输出到光栅化器522。

[0146] 光栅化器522可以执行深度拣选和其他基于深度的优化。光栅化器522也执行对新图形图元的扫描转换以生成片段,并且将那些片段和相关联的覆盖数据输出到片段/像素处理单元524。片段/像素处理单元524是被配置成执行片段着色器程序或像素着色器程序的可编程执行单元。片段/像素处理单元524按片段或像素着色器程序所指定来变换从光栅化器522接收的片段或像素。例如,可将片段/像素处理单元524编程为执行操作产生输出到光栅操作单元526的经着色的片段或像素,这些操作包括但不限于纹理映射、着色、混合、纹理校正和透视校正。片段/像素处理单元524可以读取存储在并行处理器存储器或系统存储器中的数据以供在处理片段数据时使用。可将片段或像素着色器程序配置成以取决于为处理单元配置的采样率的样本、像素、图块或其他粒度来着色。

[0147] 光栅操作单元526是处理单元,该处理单元执行包括但不限于模板印刷、z测试、混合等的光栅操作,并将像素数据作为经处理的图形数据输出,以存储在图形存储器(例如,如图2中的并行处理器存储器222和/或图1中的系统存储器104)中,显示在一个或多个显示装置110上,或者供由一个或多个处理器102或(多个)并行处理器112中的一个来进一步处理。在一些实施例中,将光栅操作单元526配置成压缩被写入到存储器的z或颜色数据,并且解压缩从存储器读取的z或颜色数据。

[0148] 协作式多用户虚拟现实示例

[0149] 现在转向图6A,电子处理系统600的实施例可以包括:应用处理器601;永久性存储介质602,通信地耦合至应用处理器601;以及图形子系统603,通信地耦合至应用处理器601。系统600可以进一步包括:感测引擎604,通信地耦合至图形子系统603以提供所感测信息;聚焦引擎605,通信地耦合至感测引擎604和图形子系统603以提供聚焦信息;运动引擎606,通信地耦合至感测引擎604、聚焦引擎605和图形子系统603以提供运动信息;以及协作引擎607,通信地耦合至运动引擎606、聚焦引擎605和感测引擎604中的一个或多个以识别环境中的两个或更多个用户之间的共享图形分量并且与所述环境中的所述两个或更多个用户共享所述共享图形分量。

[0150] 上述应用处理器601、永久性存储介质602、图形子系统603、感测引擎604、聚焦引擎605、运动引擎606、协作引擎607、以及其他系统部件中的每一个的实施例可以采用硬件、软件或其任何适当的组合的方式来实现。例如,硬件实现方式可以包括可配置逻辑,如例如可编程逻辑阵列(PLA)、FPGA、复杂可编程逻辑器件(CPLD),或者采用使用如例如ASIC、互补金属氧化物半导体(CMOS)或晶体管-晶体管逻辑(TTL)技术等电路技术的固定功能逻辑硬件,或其任意组合。替代性地或另外地,这些部件可以在作为存储在诸如随机存取存储器(RAM)、只读存储器(ROM)、可编程ROM(PROM)、固件、闪存等机器或计算机可读存储介质中的

有待由处理器或计算装置执行的一组逻辑指令的一个或多个模块中实现。例如,用于执行这些部件的操作的计算机程序代码可以以一种或多种操作系统适用/适合的编程语言的任何组合来编写,包括如PYTHON、PERL、JAVA、SMALLTALK、C++、C#等面向对象的编程语言、以及如“C”编程语言或类似的编程语言等常规的程序化编程语言。

[0151] 例如,系统600可以包括与系统100类似的部件和/或特征,进一步被配置有如本文中所描述的协作引擎。另外或可替代地,图形子系统603可以包括与并行处理器200类似的部件和/或特征,进一步被配置有如本文中所描述的协作引擎。系统600还可以被适配成与立体声头戴式系统一起工作,如例如以下结合图11至图15所描述的系统。

[0152] 感测引擎示例

[0153] 现在转向图6B,感测引擎612可以从传感器、内容、服务、和/或其他源得到信息以提供所感测信息。所感测信息可以包括例如图像信息、音频信息、运动信息、深度信息、温度信息、生物特征信息、GPU信息等。在较高层次上,一些实施例可以使用所感测信息来降低用户的图形系统的工作负荷或增强所述系统的性能。

[0154] 例如,感测引擎可以包括传感器中枢,所述传感器中枢通信地耦合至二维(2D)相机、三维(3D)相机、深度相机、陀螺仪、加速度计、惯性测量单元(IMU)、定位服务、麦克风、接近度传感器、温度计、生物特征传感器等、和/或向聚焦引擎和/或运动引擎提供信息的多个源的组合。传感器中枢可以跨多个装置分布。来自所述传感器中枢的信息可以包括来自用户装置的输入数据(例如,触摸数据)或与所述输入数据进行组合。

[0155] 例如,用户的(多个)装置可以包括一个或多个2D相机、3D相机、和/或深度相机。用户的(多个)装置还可以包括陀螺仪、加速度计、IMU、定位服务、温度计、生物特征传感器等。例如,用户可以携带智能电话(例如,在用户的口袋中)和/或可以穿戴可穿戴装置(例如,诸如智能手表、活动监视器、和/或健康跟踪器)。用户的(多个)装置还可以包括麦克风,所述麦克风可以用于检测到用户是否正在说话、打电话、跟另一个附近的人说话等。传感器中枢可以包括用户的能够采集与用户的动作或活动有关信息的各种装置(例如,包括可以采集键盘/鼠标/触摸活动的用户装置的I/O接口)中的一些或全部。传感器中枢可以直接从用户的装置的采集装置中得到信息(例如,有线或无线地),或者传感器中枢可以能够根据服务器或服务整合来自装置的信息(例如,传感器中枢可以下载的信息可以从健康跟踪器上传到云服务)。

[0156] 聚焦引擎示例

[0157] 现在转向图6C,聚焦引擎614可以从感测引擎和/或运动引擎以及其他源得到信息以提供聚焦信息。聚焦信息可以包括例如焦点、聚焦区域、眼睛位置、眼睛运动、瞳孔大小、瞳孔扩张、焦深(DOF)、内容焦点、内容聚焦对象、内容聚焦区域等。聚焦信息还可以包括之前的聚焦信息、所确定的未来聚焦信息、和/或所预测聚焦信息(例如,所预测焦点、所预测聚焦区域、所预测眼睛位置、所预测眼睛运动、所预测瞳孔大小、所预测的瞳孔扩张、所预测DOF、所确定的未来内容焦点、所确定的未来内容聚焦对象、所确定的未来内容聚焦区域、所预测内容焦点、所预测内容聚焦对象、所预测内容聚焦区域等)。

[0158] 在较高层次上,一些实施例可以使用聚焦信息基于以下各项来降低用户的图形系统的工作负荷或增强所述系统的性能:1)假设用户要看何处、2)确定用户要看何处、3)应用想要用户看何处、和/或4)预测用户将来要看何处。一些聚焦提示可能在用户正看的聚焦区

中更强。如果用户正直视前方,则他们可能看到处于锐聚焦的事物。对于朝向外围的场景或对象,用户可以注意到运动,但不会注意到锐聚焦中的细节。

[0159] 例如,如果图形系统的所感测信息或处理能力有限(例如,所附接头戴式显示器(HMD)或主机不能提供或使用信息),则聚焦信息可以是静态的和/或基于假设的(例如,假设用户正以固定的眼睛位置、DOF等看着屏幕的中央)。聚焦信息还可以基于诸如运动信息(例如,来自虚拟现实(VR)耳机)、运动预测信息、内容信息(例如,场景中的运动)等而动态地变化。更优选地,可以使用包括眼睛跟踪(有时也被称为视线跟踪)的富传感器组来识别聚焦区并提供聚焦信息从而提供更好的用户体验。例如,一些实施例可以包括眼睛跟踪器或者从眼睛跟踪器中得到眼睛信息来跟踪用户的眼睛。眼睛信息可以包括眼睛位置、眼睛运动、瞳孔大小/扩张、焦深等。眼睛跟踪器可以采集用户的眼睛包括瞳孔的图像。可以基于眼睛位置和瞳孔扩张来确定、推测、和/或估计用户的焦点和/或DOF。用户可以经受校准过程,这可以帮助眼睛跟踪器提供更准确的聚焦和/或DOF信息。

[0160] 例如,当用户正戴着VR耳机时,相机可以采集瞳孔的图像,并且系统可以确定用户正在看何处(例如,聚焦区域、深度和/或方向)。相机可以采集瞳孔扩张信息,并且系统可以基于此信息来推断用户的聚焦区域在何处。例如,人眼具有一定的DOF,从而使得如果人聚焦于附近的某物,则远处的事物可能会模糊。聚焦信息可以包括在聚焦距离X处的焦点以及 $\Delta(X)$ 的DOF信息,因此,聚焦区域可以与定位在用户的焦点周围的 $X \pm \Delta(X)$ 相对应。DOF的大小可以随距离X(例如,不同聚焦距离处的不同 Δ)而变化。例如,用户的DOF可以被校准并且可以在每个方向(例如,x、y和z)上变化,从而使得函数 $\Delta(X)$ 可以不必是球形的。

[0161] 在一些实施例中,所述聚焦信息可以包括基于内容的聚焦信息。例如,在3D、VR、和/或增强现实(AR)环境中,可以从应用中提供深度和/或距离信息(例如,用户处于虚拟环境中何处、对象所处何处、和/或对象距离用户有多远等)。基于内容的聚焦信息还可以包括应用想要用户聚焦在内容中的点、对象或区域,比如应用想要用户关注的正发生的更有趣的某事。应用还可以能够提供未来内容聚焦信息,因为所述应用可能知道内容的运动信息和/或下一帧或场景中的哪些对象/区域可能对用户而言更有趣(例如,即将从屏幕边缘进入场景的对象)。

[0162] 运动引擎示例

[0163] 现在转向图6D,运动引擎616可以从感测引擎和/或聚焦引擎以及其他源得到信息以提供运动信息。运动信息可以包括例如头部位置、头部速度、头部加速度、头部运动方向、眼睛速度、眼睛加速度、眼睛运动方向、对象位置、对象速度、对象加速度、对象运动方向等。运动信息还可以包括之前的运动信息、所确定的未来运动信息、和/或所预测运动信息(例如,所预测头部速度、所预测头部加速度、所预测头部位置、所预测头部运动方向、所预测眼睛速度、所预测眼睛加速度、所预测眼睛运动方向、所确定的未来内容位置、所确定的未来内容对象速度、所确定的未来内容对象加速度、所预测对象位置、所预测对象速度、所预测对象加速度等)。

[0164] 在较高层次上,一些实施例可以使用运动信息基于以下各项来降低用户的图形系统的工作负荷或增强所述系统的性能:1) 用户移动他们的头部,2) 用户移动他们的眼睛,3) 用户移动他们的身体,4) 应用想要用户转动他们的头部/眼睛和/或身体,和/或4) 预测用户

未来会转动他们的头部、眼睛和/或身体。可以容易地从所感测信息中确定一些运动信息。例如,可以从加速度计中确定头部位置、速度、加速度、运动方向等。可以通过随时间变化跟踪眼睛位置信息来确定眼睛运动信息(例如,如果眼睛跟踪器仅提供眼睛位置信息)。

[0165] 一些运动信息可以是基于内容的。例如,在游戏或即时3D内容中,应用可以知道对象正移动的有多快以及正在何处移动。所述应用可以向运动引擎(例如,通过API调用)提供信息。还可以将下一帧/场景的未来基于内容的对象运动信息馈送到运动引擎中以供作出决策。一些基于内容的运动信息可以通过对所述内容进行图像处理或机器视觉处理来确定。

[0166] 例如,机器视觉系统的一些实施例可以对由相机采集的图像进行分析和/或执行特征/对象辨识。例如,机器视觉和/或图像处理可以识别和/或辨识场景中的对象(例如,边缘属于椅子的前部)。机器视觉系统还可以被配置成执行面部辨识、视线跟踪、面部表情辨识、和/或姿势辨识,所述姿势包括身体级别的姿势、手臂/腿部级别的姿势、手部级别的姿势、和/或手指级别的姿势。机器视觉系统可以被配置成对用户的动作进行分类。在一些实施例中,适当配置的机器视觉系统可以能够判定用户是否位于计算机处存在、在键盘上打字、使用鼠标、使用触控板、使用触摸屏、使用HMD、使用VR系统、坐着、站立、和/或以其他方式采取某个其他动作或活动。

[0167] 例如,运动引擎可以得到与场景中的真实对象有关的相机数据,并且可以使用此信息来识别所述真实对象的运动和定向。运动引擎可以从图形处理器得到延迟信息。运动引擎然后可以预测此真实对象的下一帧的定向。延迟量可以基于用于渲染场景的时间、场景中虚拟对象的数量、以及场景的复杂度等中的一项或多项。例如,场景引擎可以包括用于采集真实场景的一个或多个相机。例如,所述一个或多个相机可以包括一个或多个2D相机、3D相机、深度相机、高速相机或其他图像采集装置。真实场景可以包括在场景中移动的对象。相机可以耦合至图像处理器,所述图像处理器用于处理来自相机的数据以便识别场景中的对象(例如,包括移动对象)并且识别所述对象的运动(例如,包括定向信息)。运动引擎可以基于跟踪对象的运动来确定所预测运动信息,并且基于所测量或所估计延迟(例如,从采集时间到渲染时间)来预测对象的未来位置。根据一些实施例,可以使用光流技术和其他真实运动估计技术来增强各种运动跟踪和/或运动预测技术以确定真实对象的下一个位置。例如,一些实施例可以使用扩展公共过滤和/或透视处理(例如,来自自动驾驶应用)来预测对象的运动。

[0168] 协作引擎示例

[0169] 现在转向图6E,协作引擎618可以从感测引擎、聚焦引擎、运动引擎、内容、多个用户(例如,HMD#1、HMD#2、至HMD#N)、和/或其他源得到信息以提供共享信息和可选地单独信息(例如,用户特定信息)。共享信息可以包括例如图像信息、音频信息、运动信息、深度信息、温度信息、生物特征信息、GPU信息、对象空间信息、几何信息、图元信息、物理信息、人工智能(AI)信息等。在较高层次上,一些实施例可以使用共享信息来降低用户的(多个)图形系统的工作负荷或增强所述系统的性能。单独信息(如果提供的话)可以包括特定于每个用户的相同类型的信息,并且用户的系统可以根据需要合并共享信息与特定信息。

[0170] 引擎重叠示例

[0171] 本领域技术人员将理解,本文中所描述的各种引擎的各方面可以与其他引擎重

叠,并且每个引擎的各部分可以贯穿电子处理系统的各个部分来实现或分布。例如,聚焦引擎可以使用运动信息来提供所预测的未来聚焦区域,并且运动引擎可以使用聚焦信息来预测未来运动。眼睛运动信息可以直接来自感测引擎,可由聚焦引擎来确定/预测,和/或可由运动引擎来确定/预测。就具体实现方式而言,本文中的示例应当被认为是说明性的而非限制性的。

[0172] 现在转向图6F,图形设备620的实施例可以包括:处理器621;存储器622,通信地耦合至处理器621;以及协作引擎623,通信地耦合至所述处理器以用于识别环境中的两个或更多个用户之间的共享图形分量并且与所述环境中的所述两个或更多个用户共享所述共享图形分量。

[0173] 设备620的一些实施例可以包括用于支持协作式图形处理的任何数量的附加模块。例如,协作引擎623可以包括集中式共享器624、深度共享器625、共享预处理器626、多端口图形子系统627、以及解码共享器628中的一个或多个。

[0174] 在设备620的一些实施例中,例如,协作引擎623可以进一步包括集中式共享器624(例如,如以下更详细描述)。例如,集中式共享器可以被配置成将共享图形分量广播至所有所述两个或更多个用户,并且将单独的图形分量单独地分配至所述两个或更多个用户中的一个用户。集中式共享器624可以进一步被配置成基于目标虚拟现实设备来分割所述共享图形分量的工作负荷。

[0175] 在设备620的一些实施例中,例如,协作引擎623可以进一步包括深度共享器625(例如,如以下更详细描述)。例如,深度共享器625可以被配置成从独立于所述两个或更多个用户中的第一用户的源处收集深度信息,并且与所述第一用户共享所述深度信息。深度共享器625可以进一步被配置成从所述第一用户的视场之外的所述独立源处收集可见性信息,并且与所述第一用户共享所述可见性信息。

[0176] 在设备620的一些实施例中,例如,协作引擎623可以进一步包括共享预处理器626(例如,如以下更详细描述)。例如,共享预处理器626可以被配置成预计算与所述共享图形分量有关的信息,并且与所述两个或更多个用户共享预计算的信息。预计算的信息可以包括几何信息、图元信息、和物理信息中的一项或多项。

[0177] 在设备620的一些实施例中,例如,协作引擎623可以进一步包括多端口图形子系统627(例如,如以下更详细描述)。多端口图形子系统627可以被配置成在多端口图形子系统627的每个端口上支持不同的用户。例如,多端口图形子系统627可以进一步被配置成:当仅第一虚拟现实设备连接至多端口图形子系统627时,支持所述第一虚拟现实设备的左眼显示器和右眼显示器;当所述第一虚拟现实设备和第二虚拟现实设备两者均连接至多端口图形子系统627时,支持所述第一虚拟现实设备和所述第二虚拟现实设备;并且当所述第一虚拟现实设备和所述第二虚拟现实设备两者均连接至多端口图形子系统627时,在所述第一虚拟现实设备与所述第二虚拟现实设备之间共享图形图元。

[0178] 在设备620的一些实施例中,例如,协作引擎623可以进一步包括解码共享器628(例如,如以下更详细描述)。解码共享器628可以被配置成:识别第一用户的第一视图区与第二用户的第二视图区之间的共享场景的重叠区;对所述重叠区进行解码;并且与所述第一用户和所述第二用户两者共享经解码的重叠区。解码共享器628还可以被配置成基于所述重叠区对所述解码进行优先级排序。

[0179] 上述处理器621、存储器622、协作引擎623、集中式共享器624、深度共享器625、共享预处理器626、多端口图形子系统627、解码共享器628、以及其他系统部件中的每一个的实施例可以采用硬件、软件或其任何适当的组合的方式来实现。例如,硬件实现方式可以包括例如PLA、FPGA、CPLD等可配置逻辑,或者采用使用例如ASIC、CMOS或TTL技术等电路技术的固定功能逻辑硬件,或其任意组合。替代性地或另外地,这些部件可以在作为存储在如RAM、ROM、PROM、固件、闪存等机器或计算机可读存储介质中的有待由处理器或计算装置执行的一组逻辑指令的一个或多个模块中实现。例如,用于执行这些部件的操作的计算机程序代码可以以一种或多种操作系统适用/适合的编程语言的任何组合来编写,包括如PYTHON、PERL、JAVA、SMALLTALK、C++、C#等面向对象的编程语言、以及如“C”编程语言或类似的编程语言等常规的程序化编程语言。

[0180] 例如,设备620可以包括与系统100类似的部件和/或特征,进一步被配置有协作引擎。例如,设备620可以另外地或可替代地包括与并行处理器200类似的部件和/或特征,进一步被配置有如本文中所描述的协作引擎。设备620还可以被适配成与立体声头戴式系统一起工作,如例如以下结合图11至图15所描述的系统。

[0181] 现在转向图6G至图6L,图形协作的方法630的实施例可以包括:在框631处,识别环境中的两个或更多个用户之间的共享图形分量;以及在框632处,与所述环境中的所述两个或更多个用户共享所述共享图形分量。

[0182] 在一些实施例中,方法630可以进一步包括:在框633处,将共享图形分量广播至所有所述两个或更多个用户;以及在框634处,将单独的图形分量单独地分配至所述两个或更多个用户中的一个用户。一些实施例还可以包括:在框635处,基于目标虚拟现实设备来分割所述共享图形分量的工作负荷。

[0183] 在一些实施例中,方法630可以进一步包括:在框636处,从独立于所述两个或更多个用户中的第一用户的源处收集深度信息;以及在框637处,与所述第一用户共享所述深度信息。所述方法还可以包括:在框638处,从所述第一用户的视场之外的所述独立源处收集可见性信息;以及在框639处,与所述第一用户共享所述可见性信息。

[0184] 在一些实施例中,方法630可以进一步包括:在框640处,预计算与所述共享图形分量有关的信息;以及在框641处,与所述两个或更多个用户共享预计算的信息。例如,在框642处,预计算的信息可以包括几何信息、图元信息、和物理信息中的一项或多项。

[0185] 在一些实施例中,方法630可以进一步包括:在框643处,在多端口图形子系统的每个端口上支持不同的用户。例如,方法630可以包括:在框644处,当仅第一虚拟现实设备连接至多端口图形子系统时,支持所述第一虚拟现实设备的左眼显示器和右眼显示器;在框645处,当所述第一虚拟现实设备和第二虚拟现实设备两者均连接至多端口图形子系统时,支持所述第一虚拟现实设备和所述第二虚拟现实设备;以及在框646处,当所述第一虚拟现实设备和所述第二虚拟现实设备两者均连接至多端口图形子系统时,在所述第一虚拟现实设备与所述第二虚拟现实设备之间共享图形图元。

[0186] 在一些实施例中,方法630可以进一步包括:在框647处,识别第一用户的第一视图区与第二用户的第二视图区之间的共享场景的重叠区;在框648处,对所述重叠区进行解码;以及在框649处,与所述第一用户和所述第二用户两者共享经解码的重叠区。在框650处,一些实施例还可以包括基于所述重叠区对所述解码进行优先级排序。

[0187] 方法630的实施例可以在系统、设备、GPU、并行处理单元 (PPU)、或如例如本文中所描述的这些图形处理器流水线设备中来实现。更具体地,方法630的硬件实现方式可以包括可配置逻辑(如例如PLA、FPGA、CPLD),或者采用使用电路技术(如例如ASIC、CMOS或TTL技术)的固定功能逻辑硬件,或其任意组合。替代性地或另外地,方法630可以在作为存储在如RAM、ROM、PROM、固件、闪存等机器或计算机可读存储介质中的有待由处理器或计算装置执行的一组逻辑指令的一个或多个模块中实现。例如,用于执行这些部件的操作的计算机程序代码可以以一种或多种操作系统适用/适合的编程语言的任何组合来编写,包括如PYTHON、PERL、JAVA、SMALLTALK、C++、C#等面向对象的编程语言、以及如“C”编程语言或类似的编程语言等常规的程序化编程语言。例如,方法630可以在如以下结合示例23至示例33描述的计算机可读介质上来实现。

[0188] 例如,方法630的实施例或一部分可以采用应用(例如,通过API)或驱动程序软件来实现。方法630的其他实施例或一部分可以采用有待在GPU上执行的专用代码(例如,着色器)来实现。方法630的其他实施例或一部分可以采用固定功能逻辑或专用硬件(例如,在GPU中)来实现。

[0189] 协作式多用户主机/客户端示例

[0190] 本文中所描述的多用户协作式VR系统的各个实施例可以在各种装置包括如云服务器等服务器系统/主机系统上来实现。其他主机系统可以包括PC,诸如台式计算机、膝上型计算机、可转换计算机、平板机等。在VR情境中,客户端系统可以包括用户的HMD或用于消费内容的其他VR装备。概括地说,来自一个源(例如,中央源或其他用户)的信息可以与用户共享(例如,以便降低用户的工作负荷和/或增强用户的体验)。用户的系统因此可以被配置成接收并使用共享信息和/或合并共享信息与本地和/或用户特定信息。

[0191] 现在转向图6M,图形设备670的实施例可以包括:处理器671;存储器672,通信地耦合至处理器671;协作引擎673,通信地耦合至处理器671以接收共享图形分量;以及合成器674,通信地耦合至处理器671以合并共享图形分量与单独的图形分量。设备670的一些实施例可以进一步包括用于在每个端口上支持不同用户的多端口图形子系统675。

[0192] 设备670的一些实施例可以进一步包括:待由用户穿戴的可穿戴外壳676;以及通信地耦合至处理器671的显示器677,其中,显示器677、处理器671、存储器672、协作接口673以及合成器674可全部由可穿戴外壳676支持。例如,可穿戴外壳676可穿戴在用户的头部、用户的颈部、用户的肩部等上,其中,显示器677定位在或能够定位在用户的眼睛附近。

[0193] 在一些实施例中,协作接口673可以被配置成广播包括共享图形分量的信息。例如,共享图形分量可以包括共享深度信息、共享几何信息和共享物理信息中的一项或多项。在一些实施例中,共享图形分量可以包括360视频内容。

[0194] 上述处理器671、存储器672、协作接口673、合成器674、多端口图形子系统675、显示器677、以及其他系统部件中的每一个的实施例可以采用硬件、软件或其任何适当的组合的方式来实现。例如,硬件实现方式可以包括如例如PLA、FPGA、CPLD等可配置逻辑,或者采用使用如例如ASIC、CMOS或TTL技术等电路技术的固定功能逻辑硬件,或其任意组合。替代性地或另外地,这些部件可以在作为存储在如RAM、ROM、PROM、固件、闪存等机器或计算机可读存储介质中的有待由处理器或计算装置执行的一组逻辑指令的一个或多个模块中实现。例如,用于执行这些部件的操作的计算机程序代码可以以一种或多种操作系统适用/适合

的编程语言的任何组合来编写,包括如PYTHON、PERL、JAVA、SMALLTALK、C++、C#等面向对象的编程语言、以及如“C”编程语言或类似的编程语言等常规的程序化编程语言。

[0195] 例如,设备670可以包括与系统100类似的部件和/或特征,进一步被配置有协作接口以及合成器。例如,设备670可以另外地或可替代地包括与并行处理器200类似的部件和/或特征,进一步被配置有如本文中所描述的协作接口以及合成器。设备670还可以被适配成与立体声头戴式系统一起工作,如例如以下结合图11至图15所描述的系统。

[0196] 现在转向图6N,图形协作的方法680可以包括:在框681处,接收共享图形分量;以及在框682处,合并共享图形分量与单独的图形分量。方法680可以进一步包括:在框683处,提供待由用户穿戴的可穿戴外壳;以及在框684处,使用可穿戴外壳来支持显示器、处理器、存储器、协作接口以及合成器。例如,在框685处,共享图形分量可以包括共享深度信息、共享几何信息和共享物理信息中的一项或多项,和/或在框686处,共享图形分量可以包括360视频内容。方法680还可以包括:在框687处,接收包括共享图形分量的广播信息。一些实施例还可以包括:在框688处,在多端口图形子系统的每个端口上支持不同的用户。

[0197] 方法680的实施例可以在系统、设备、GPU、PPU、或如例如本文中所描述的这些图形处理器流水线设备中来实现。更具体地,方法680的硬件实现方式可以包括可配置逻辑(如例如PLA、FPGA、CPLD),或者采用使用电路技术(如例如ASIC、CMOS或TTL技术)的固定功能逻辑硬件,或其任意组合。替代性地或另外地,方法680可以在作为存储在如RAM、ROM、PROM、固件、闪存等机器或计算机可读存储介质中的有待由处理器或计算装置执行的一组逻辑指令的一个或多个模块中实现。例如,用于执行这些部件的操作的计算机程序代码可以以一种或多种操作系统适用/适合的编程语言的任何组合来编写,包括如PYTHON、PERL、JAVA、SMALLTALK、C++、C#等面向对象的编程语言、以及如“C”编程语言或类似的编程语言等常规的程序化编程语言。例如,方法680可以在如以下结合示例57至示例61描述的计算机可读介质上来实现。

[0198] 例如,方法680的实施例或一部分可以采用应用(例如,通过API)或驱动程序软件来实现。方法680的其他实施例或一部分可以采用有待在GPU上执行的专用代码(例如,着色器)来实现。方法680的其他实施例或一部分可以采用固定功能逻辑或专用硬件(例如,在GPU中)来实现。

[0199] 集中式共享器示例

[0200] 现在转向图7A,集中式共享器设备700可以包括:广播器721,用于将共享图形分量广播至所有的两个或更多个用户;以及分配器722,通信地耦合至广播器721以便将单独的图形分量单独地分配至所述两个或更多个用户中的一个用户。设备700的一些实施例可以进一步包括:工作分割器723,用于基于目标虚拟现实设备来分割所述共享图形分量的工作负荷。设备700的一些实施例可以实现方法630和/或方法680(例如,参见图6H和6N)的一部分。

[0201] 上述广播器721、分配器722、工作分割器723以及设备700的其他部件中的每一个的实施例都可以采用硬件、软件或其组合的方式来实现。例如,设备700的一部分或全部可以被实现为并行处理器200的一部分,进一步被配置有如本文中所描述的广播器、分配器和/或工作分割器。设备700还可以被适配成与立体声头戴式系统一起工作,如例如以下结合图11至图15所描述的系统。例如,硬件实现方式可以包括如例如PLA、FPGA、CPLD等可配置

逻辑,或者采用使用如例如ASIC、CMOS或TTL技术等电路技术的固定功能逻辑硬件,或其任意组合。替代性地或另外地,这些部件可以在作为存储在如RAM、ROM、PROM、固件、闪存等机器或计算机可读存储介质中的有待由处理器或计算装置执行的一组逻辑指令的一个或多个模块中实现。例如,用于执行这些部件的操作的计算机程序代码可以以一种或多种操作系统适用/适合的编程语言的任何组合来编写,包括如PYTHON、PERL、JAVA、SMALLTALK、C++、C#等面向对象的编程语言、以及如“C”编程语言或类似的编程语言等常规的程序化编程语言。

[0202] 一些实施例可以有利地提供了通过无线显示器在多个用户之中进行广播的集中式VR渲染框架。例如,当多个用户在共享VR环境中交互时,它们可以在中央机器上执行VR渲染并且将结果广播至其无线显示器上。中央机器可以处理某些共同工作,并且将用户特定的差异留给每个用户的末端单元以完成最终渲染。

[0203] 例如,在主题公园骑行的用户大多数情况下可以观看相同的内容,并且可以在集中式单元处完成这种共同工作。在用户具有不同观看角度时的情况下,集中式机器可以处理任何所识别的共同工作并且将一些用户特定的差异留给每个用户的末端单元以完成最终渲染。根据一些实施例,可以通过使用用于广播信息的公共频率来增大网络带宽。

[0204] 一些实施例可以针对图形处理器与一个或多个目标HMD之间的交互提供更高效率的分配模型。例如,代替与每个HMD的点对点连接,一些实施例可以将一些要素广播至多个用户。与点对点模型相比,广播共享内容可以节省带宽。即使每个用户具有不同的视点,仍然可以共享一些有用信息。例如,一些实施例可以使用集中式机器来广播共享信息。每个用户还可以针对其特定信息/内容得到单独的流。这些信息可以在本地被合并在一起,以用于用户的完整、特定视图。

[0205] 一些实施例可以识别共同工作与用户特定工作。对于共同的用户体验(例如,主题公园骑行),通过设计可能存在许多共同的要素。取决于用户在乘坐汽车中的位置,乘坐汽车中的四名用户可能仅仅以略微不同的角度看到大部分相同的场景。相同内容可以是可广播至用户的共享分量。取决于用户的确切位置,这些信息可以根据需要被内插和/或重新定向到用户的视点。在一些环境中,共同工作/共享分量可能提前已知。共享分量可以集中渲染,而用户特定部分可以在本地渲染。在一些实施例中,特定工作还可以在集中位置处进行渲染,但是可以被单独地发送而不是广播。用户的装置可以被配置成合并共享分量和用户特定部分。

[0206] 在一些实施例中,可以在共同链路(例如,同一频率)上发送共享信息以用于广播。一般而言,广播可以指只发送一次信息并且由所有用户接收信息。与针对多个用户发送多次相同或类似信息相比,识别共享分量并针对所有用户发送一次共享分量可以节省带宽。

[0207] 对于骑行的示例,用户可以从略微不同的角度看到背景。但是出于提高通用性和节省带宽的目的,可以忽视轻微差异。一些实施例可以将背景渲染为不会随视点/定向的微小变化而精确地倾斜的静态图像。相对静态的背景可以在节省大量带宽的同时不对用户体验造成可感知的影响。在一些实施例中,视点的变化量可以是基于阈值的。例如,如果第二个用户的视点在第一个用户的视点的五度范围内,则可以提供相同的背景。通过设计,一些实施例可以并不坚持对每个用户的视点的理想渲染以节省处理和/或网络带宽。例如,对于增强现实(AR)体验,即使用户的定向/位置不完全相同,也可以针对多于一个用户以相同的

定向/位置来渲染AR对象。

[0208] 对于已安装的AR/VR体验,众多用户可能会定期经历这种体验。尽管从特定用户的角度来看,当他们经历骑行时,体验会发生变化,但还有另一组用户很快进入场景,他们将与之前的用户有非常相似的体验。可以针对各个位置和定向来预先渲染共同信息。然后,用户的HMD可以传输其各自的位置和定向信息,并且集中式机器可以传输用于此位置和定向的适当的场景信息。

[0209] 对于共同分量和特定分量,一些实施例可以在两个阶段中实现。一些实施例可以在第一阶段中一起渲染和/或广播共同分量并且在第二阶段中根据需要分配特定分量。一些实施例可以有利地通过广播共同分量来节省带宽。集中式共享处理器(例如,GPU或AR/VR处理器)可以识别场景的共同分量和场景的特定分量,在适用时处理共同分量,并广播所处理的共同分量。集中式共享处理器还可以分配特定分量。用户的AR/VR装置(例如,HMD)可以包括用于合并共同分量和特定分量的合成器。

[0210] 集中式共享处理器可以不同程度地处理共享分量。在一些情况下,集中式共享处理器可以渲染至最终显示像素。共同分量可以不被渲染至最终显示像素。HMD仍然可以对共同分量执行某种渲染或扭曲处理。例如,一些共同分量可能仅与共同的模型和/或几何图形相对应。集中式处理器可以将一些处理推迟到HMD,所述HMD可以具有更好的本地信息来完成处理。

[0211] 一些实施例可以包括用于基于HMD的处理能力来分割共同工作的工作分割器。例如,不同的HMD可能具有不同的处理能力和/或部件来处理不同的图形工作负荷。简单的HMD可能需要集中式共享处理器来渲染至最终显示像素,而更复杂的HMD可以能够处理更多的工作负荷。集中式处理器可以被配置成识别目标HMD的类型并相应地调整工作负荷。

[0212] 深度共享器示例

[0213] 现在转向图8A,深度共享设备800的实施例可以包括:深度信息收集器821,用于从独立于两个或更多个用户的第一用户的源处收集深度信息;以及深度信息共享器822,通信地耦合至深度信息收集器821以便与第一用户共享深度信息。设备800的一些实施例可以进一步包括:可见性信息收集器823,用于从第一用户的视场之外的独立源处收集可见性信息;以及可见性信息共享器824,通信地耦合至可见性信息收集器823以便与第一用户共享可见性信息。设备800的一些实施例可以实现方法630和/或方法680(例如,参见图6I和6N)的一部分。

[0214] 上述深度信息收集器821、深度信息共享器822、可见性信息收集器823、可见性信息共享器824、以及设备800的其他部件中的每一个的实施例都可以采用硬件、软件或其组合的方式来实现。例如,设备800的一部分或全部可以被实现为并行处理器200的一部分,进一步被配置有如本文中所描述的深度信息收集器、深度信息共享器、可见性信息收集器和/或可见性信息共享器。设备800还可以被适配成与立体声头戴式系统一起工作,如例如以下结合图11至图15所描述的系统。例如,硬件实现方式可以包括如例如PLA、FPGA、CPLD等可配置逻辑,或者采用使用如例如ASIC、CMOS或TTL技术等电路技术的固定功能逻辑硬件,或其任意组合。替代性地或另外地,这些部件可以在作为存储在如RAM、ROM、PROM、固件、闪存等机器或计算机可读存储介质中的有待由处理器或计算装置执行的一组逻辑指令的一个或多个模块中实现。例如,用于执行这些部件的操作的计算机程序代码可以以一种或多种操

作系统适用/适合的编程语言的任何组合来编写,包括如PYTHON、PERL、JAVA、SMALLTALK、C++、C#等面向对象的编程语言、以及如“C”编程语言或类似的编程语言等常规的程序化编程语言。

[0215] 现在转向图8B,多个用户 U_1 、 U_2 、至 U_N 可以正在环境中操作AR/VR装备(例如,HMD₁、HMD₂、至HMD_N) (例如,其中, $N>2$)。多个用户可以或可以不正在使用同一个应用(例如,玩同一个游戏)和/或可以或可以不具有相同的AR/VR装备。根据一些实施例,AR/VR装备可以被配置成共享可能对其他用户有用的信息和/或接收可能对AR/VR装备有用的共享信息。

[0216] 一些实施例可以有利地提供对来自多个用户的深度相机数据的分布式处理。例如,如果多个AR用户处于一个环境中,则其AR装置可以共享深度相机信息。有利地,每个AR/VR系统可能不需要单独地开发整个场景。例如,如果在密集区域中(例如,城市环境)许多用户正使用具有深度相机的AR,则每个AR装置可以发射和接收(例如,高度压缩的)相机数据以获得更高的准确度和可能更低的处理能力。

[0217] 一些实施例可以提高定位数据的精度。例如,用户的AR装置可以将深度信息传输至附近的相邻AR装置。信息通信可以有助于准确度,因为附近的相邻装置可能具有比用户的AR装置更好的对对象的深度读数。例如,深度相机可以在一定范围内具有更高准确度。一些实施例可以有利地使用来自附加用户的附加深度信息来扩展用户的AR/VR系统的范围。一些实施例可以能够基于共享深度数据的能力来提供不太强大的相机和/或关闭相机电源(例如,功率越低,成本越低)。与本地地开发信息相比,一些实施例还可以通过从其他源得到此信息来节省功率。

[0218] 其他源还可以将深度信息传递至用户的AR/VR装置。其他源可以包括例如环境中的静态源(例如,固定相机位置)、环境中的自主源(例如,无人机或自主车辆上的相机)、和/或环境之外的源,如云服务。一些实施例可以利用对等共享、集中式共享、或两者。例如,关于移动对象的信息可以主要从对等源中基于实时感知的对象来提供,而关于静态对象的信息可以从云中提供。一些实施例可以被配置成接收来自其他源(例如,其他用户/云)的深度信息,并且将所接收深度信息与本地深度信息进行整合。

[0219] 例如,深度信息可以是源自群集的并且存储在云中。用户的AR装置然后可以基于用户的位置和定向来下载深度信息(例如,3D映射)。可以基于可用带宽和用户的AR装置使用所下载信息的能力而为所述信息提供不同程度的分辨率。基于云的深度信息共享可以对于可能相对稳定的对象是有用的。例如,用户围绕环境行走可以生成深度信息,所述深度信息可以上传至云并保存。所上传的深度信息可以用于针对在环境中处于静态的对象开发3D映射。例如,一些实施例可以使用数据分析来确定静态对象与动态对象(例如,在接收到来自100个不同用户的显示对象尚未发生变化的相同数据之后,对象可以被认为是静态的)。如果之前的静态对象似乎已经移动或者已不再被证实存在,则所述对象可以从云映射中移除。

[0220] 除了或替代增强距离/深度信息,一些实施例可以能够提供用户的视图之外的信息来增强此用户看到其视图或其相机的视图之外的能力。例如,用户可以能够看到在角落周围或其他对象后面的遮挡对象、穿过墙壁等。如果在用户前面的人阻挡用户对静态对象(例如,消防栓)的视图,则可以提醒用户静态对象存在。用户可以能够看到人的背后的事物(例如,他们对用户隐藏某物,但是其他人看到它或看到过它)。例如,穿戴AR眼睛的警察可

以看到隐藏在角落的人或隐藏的武器。一些实施例还可以提供之前在所述场景的上下文记录(例如,两分钟之前,巷子里有五个人)。

[0221] 一些实施例可以有利地提供多个VR耳机之间的协作。例如,在一个环境中操作的多个VR系统可以共享信息。在VR环境中,很有可能具有VR耳机的多个用户正在看相同的场景或几何图形。例如,可能存在同一场景的多个视图。一些实施例可以在用户可能会看不透明对象后面的事物或具有对其可用的不同视角的情况下使用VR耳机之间的协作/影像来提供增强的用户体验。

[0222] 一些实施例可以提供实体对象后面的可见性。例如,在使用共享同一个场景的VR相机的游戏环境中,即使存在遮挡视点的实体对象,第一用户也可以从其视角得到被遮挡的视点。其他用户可以提供实体对象后面的视图信息以便与第一用户共享。例如,其他用户可以正在从不受遮挡的不同角度看所述区域。其他用户可以提供可见性信息并且此可见性信息可以反射到第一用户的HMD中,因此第一用户可以看到实体对象之外的事物。其他用户的视角可以外推并重新定向到第一用户的视点。一些实施例可以包括从一个HMD中接收与场景有关的信息,将所述信息外推至其他HMD的定向,以及在其他HMD上显示重新定向信息。

[0223] 取决于可用资源(例如,存储设备、存储器、和/或处理能力),可以基于各个用户围绕区域的移动来映射所述区域,从而使得即使没有其他用户当前正看所述对象后面的区域,第一用户也可以能够看到对象后面的事物(例如,可能是时间延迟的视图而不是实时视图)。一些实施例可以对于VR应用和/或AR应用是有用的。一些实施例可以适用于自动驾驶应用来扩展驾驶员的视图。例如,多个车辆可以共享信息并将所述视图提供给超越汽车、汽车一侧、或汽车后面的驾驶员。可以向驾驶员提供可能在驾驶员的当前视图之外的危险。危险图像可以连同关于距危险的距离的信息一起投影在挡风玻璃上。

[0224] 共享预处理器示例

[0225] 现在转向图8C,共享预处理器设备840的实施例可以包括:预计算机841,用于预计算与共享图形分量有关的信息;以及预计算信息共享器842,用于与两个或更多个用户共享预计算的信息。例如,预计算的信息可以包括几何信息、图元信息、和物理信息中的一项或多项。设备840的一些实施例可以实现方法630和/或方法680(例如,参见图6J和6N)的一部分。

[0226] 上述预计算机841、预计算信息共享器842、以及设备840的其他部件中的每一个的实施例都可以采用硬件、软件或其组合的方式来实现。例如,设备840的一部分或全部可以被实现为并行处理器200的一部分,进一步被配置有如本文中所描述的预计算机和/或预计算信息共享器。设备840还可以被适配成与立体声头戴式系统一起工作,如例如以下结合图11至图15所描述的系统。例如,硬件实现方式可以包括如例如PLA、FPGA、CPLD等可配置逻辑,或者采用使用如例如ASIC、CMOS或TTL技术等电路技术的固定功能逻辑硬件,或其任意组合。替代性地或另外地,这些部件可以在作为存储在如RAM、ROM、PROM、固件、闪存等机器或计算机可读存储介质中的有待由处理器或计算装置执行的一组逻辑指令的一个或多个模块中实现。例如,用于执行这些部件的操作的计算机程序代码可以以一种或多种操作系统适用/适合的编程语言的任何组合来编写,包括如PYTHON、PERL、JAVA、SMALLTALK、C++、C#等面向对象的编程语言、以及如“C”编程语言或类似的编程语言等常规的程序化编程语言。

[0227] 现在转向图8D,多个用户 U_1 、 U_2 、至 U_N 可以正在环境中操作AR/VR装备(例如,HMD₁、

HMD₂、至HMD_N) (例如,其中,N>2)。多个用户可以正在使用同一个应用并且可以或可以不具有相同的AR/VR装备。有利地,一些实施例可以提供分解的共享仿真。在多用户环境中,例如,若干玩家可以正在经历可被预计算并分配为软件开发包(SDK)的一部分的共同AI或物理体验。在多个玩家正在经历一些相同事物的多玩家游戏中,一些实施例可以计算一些图形信息一次并在多个玩家/HMD或主机装置之中共享结果(例如,跨多个PC、游戏控制台等共享)。一些实施例可以包括用于预计算共同信息(例如,物理、AI等)并且然后将预计算的信息与多个HMD共享的服务器。例如,预计算的信息可以包括对象空间信息、几何信息、顶点信息、纹理信息、阴影信息、地形信息等。

[0228] 多端口图形子系统示例

[0229] 现在转向图9A,多端口图形子系统900的实施例可以包括用于第一显示器的第一GPU 921和用于第二显示器的第二GPU 922。图形子系统900的一些实施例可以在同一个衬底(例如,裸片/SoC/封装体/印刷电路板)上具有第一GPU和第二GPU两者,比如具有多个端口的独立显卡。图形子系统900的一些实施例可以在不同的衬底上具有第一GPU和第二GPU,比如多个独立显卡和/或主板上的集成显卡与独立显卡的组合,从而在各个可用GPU之间支持多个端口。

[0230] 根据一些实施例,多端口图形子系统900可以被配置成在多端口图形子系统900的每个端口上支持不同的用户。例如,多端口图形子系统900可以被配置成当仅第一虚拟现实设备连接至所述多端口图形子系统900时,支持所述第一虚拟现实设备的左眼显示器和右眼显示器。子系统900可以进一步被配置成:当所述第一虚拟现实设备和第二虚拟现实设备两者均连接至多端口图形子系统900时,支持所述第一虚拟现实设备和所述第二虚拟现实设备;并且当所述第一虚拟现实设备和所述第二虚拟现实设备两者均连接至多端口图形子系统900时,在所述第一虚拟现实设备与所述第二虚拟现实设备之间共享图形图元。子系统900的一些实施例可以实现方法630和/或方法680(例如,参见图6K和6N)的一部分。

[0231] 上述第一GPU 921、第二GPU 922、以及子系统900的其他部件中的每一个的实施例都可以采用硬件、软件或其组合的方式来实现。例如,子系统900的一部分或全部可以被实现为并行处理器200的一部分,进一步被配置成如本文中所描述的在多个端口上支持多个用户。子系统900还可以被适配成与立体声头戴式系统一起工作,如例如以下结合图11至图15所描述的系统。例如,硬件实现方式可以包括如例如PLA、FPGA、CPLD等可配置逻辑,或者采用使用如例如ASIC、CMOS或TTL技术等电路技术的固定功能逻辑硬件,或其任意组合。替代性地或另外地,这些部件可以在作为存储在如RAM、ROM、PROM、固件、闪存等机器或计算机可读存储介质中的有待由处理器或计算装置执行的一组逻辑指令的一个或多个模块中实现。例如,用于执行这些部件的操作的计算机程序代码可以以一种或多种操作系统适用/适合的编程语言的任何组合来编写,包括如PYTHON、PERL、JAVA、SMALLTALK、C++、C#等面向对象的编程语言、以及如“C”编程语言或类似的编程语言等常规的程序化编程语言。

[0232] 现在转向图9B和图9C,多端口图形子系统930可以包括第一端口931和第二端口932。当第一HMD 933附接至图形子系统930时,图形子系统930可以支持第一HMD 933的左眼显示器934和右眼显示器935。当第一HMD 933和第二HMD 936两者附接至图形子系统930时,图形子系统930可以支持在第一HMD 933和第二HMD 936中的每一个上的独立视图。

[0233] 有利地,一些实施例可以提供具有两个或多个用户的多用户VR。在一些实施例中,

代替左眼显示器和右眼显示器,图形子系统可以通过重用图元来支持两个用户,从而支持两个HMD。例如,在双图形设置(例如,可扩展链接界面(SLI)或独立显卡和集成显卡)中,可以存在足够的带宽来渲染至两个不同的HMD。当仅存在一个HMD时,SLI/GPU用于渲染左视图/右视图。一些实施例可以有利地共享图形分量(例如,几何图形、图元、内容、纹理等)以将3D内容渲染到在每个HMD中显示有不同内容的两个不同的HMD。一些实施例可以支持具有不同的“看”向量的一个视点,而一些实施例可以支持多个独立的视点。一些实施例还可以支持twitch,其中,一个用户可以利用其他用户的VR交互。例如,可以广播或分配VR场景以供由其他用户观看。

[0234] 一些显卡可以支持多个输出,其中,用户可以将多个HMD连接至输出以得到同一个VR场景。一些实施例可以有利地通过图元共享来支持所连接HMD的不同定向。例如,系统可以渲染一个360度图像,并且每个用户可以使其自己的视图进入到此360度图像中。一些实施例可以作出关于每个用户的位置和视场(例如,他们正在看什么)的判定,并且识别信息以在用户之间共享。在一些实施例中,可以在集成GPU与独立GPU之间发生共享。

[0235] 用户可以将多个HMD插入到同一个GPU中并且使用此GPU来针对两个HMD进行渲染。代替使显卡支持左眼显示器和右眼显示器,一个显卡可以支持两个用户。如果两个用户正在玩同一个游戏(例如,社交VR或协作VR),则可以共享用户的场景之间的重叠以便高效地利用GPU。例如,可能存在不需要重复渲染的一个或多个共同的几何图形。可以针对每个视点共享这些共同的几何图形,从而降低GPU上的工作负荷。当多个GPU可用时(例如,每个用户具有其自己的GPU),在多个GPU之间分割共同的几何图形可以降低每个GPU的工作负荷。一些实施例可以有利地节省功率或增大用于其他GPU操作(例如,用于改善用户体验的所增加的细节)的可用带宽。

[0236] 例如,第一GPU可以针对第一用户的视角得到照明计算、颜色、程度等。此信息可以传递至第二GPU以供其使用第二用户的不同视角。一些实施例可以通过不必从其自己的磁盘/存储器中检索此信息来节省第二GPU上的带宽。一些实施例可以利用高速缓存按需共享信息。一些实施例可以基于这两个GPU的处理能力来分割工作负荷。例如,第一GPU可以得到照明计算而第二GPU得到颜色信息。

[0237] 一些实施例可以在一个图形子系统具有两个显卡来支持一个HMD,一个用于左眼并且一个用于右眼。在一些实施例中,此图形子系统可以进一步支持两个HMD。用户可以将一个HMD连接至这一个显卡并且将另一个HMD连接至另一个显卡。图形子系统可以支持这两个显卡之间的图元共享。每个卡可以渲染两只眼睛,但通过图元共享仍然可以在支持两个用户而不是一个用户的同时提供良好的用户体验。

[0238] 一些实施例可以允许一组用户使用连接至显卡上的不同端口的多个HMD来播放来自云服务(例如,YOUTUBE或FACEBOOK)的360视频。例如,集成显卡系统可以包括VGA端口和HDMI端口和/或显示端口。这些端口可以能够同时支持多个监视器。一些实施例可以提供驱动程序来支持多个HMD。

[0239] 在显卡支持多个输出的一些实施例中,每个HMD可以具有其自己的视口和观看方向。在一些应用中,可以存在可能正在看一个方向的一个主用户,而其他观看者可能正在看不同的方向。一些实施例可以在一些协作性游戏中是有用的,其中,(多个)观看者可以提醒主用户发生在(多个)观看者的视场中的主用户可能未在观看的活动。

[0240] 解码共享器示例

[0241] 现在转向图10A,解码共享器1000的实施例可以包括:区识别器1021,用于识别第一用户的第一视图区与第二用户的第二视图区之间的共享场景的重叠区;解码器1022,用于对所述重叠区进行解码;以及区共享器1023,用于与所述第一用户和所述第二用户两者共享经解码的重叠区。设备1000的一些实施例可以进一步包括:解码优先级排序器1024,用于基于所述重叠区对所述解码进行优先级排序。设备1000的一些实施例可以实现方法630和/或方法680(例如,参见图6L和6N)的一部分。

[0242] 上述区识别器1021、解码器1022、区共享器1023、解码优先级排序器1024、以及设备1000的其他部件中的每一个的实施例都可以采用硬件、软件或其组合的方式来实现。例如,设备1000的一部分或全部可以被实现为并行处理器200的一部分,进一步被配置有如本文中所描述的区识别器、解码器1022、区共享器和/或解码优先级排序器。设备1000还可以被适配成与立体声头戴式系统一起工作,如例如以下结合图11至图15所描述的系统。例如,硬件实现方式可以包括如例如PLA、FPGA、CPLD等可配置逻辑,或者采用使用如例如ASIC、CMOS或TTL技术等电路技术的固定功能逻辑硬件,或其任意组合。替代性地或另外地,这些部件可以在作为存储在如RAM、ROM、PROM、固件、闪存等机器或计算机可读存储介质中的有待由处理器或计算装置执行的一组逻辑指令的一个或多个模块中实现。例如,用于执行这些部件的操作的计算机程序代码可以以一种或多种操作系统适用/适合的编程语言的任何组合来编写,包括如PYTHON、PERL、JAVA、SMALLTALK、C++、C#等面向对象的编程语言、以及如“C”编程语言或类似的编程语言等常规的程序化编程语言。

[0243] 现在转向图10B,多端口图形子系统1030可以包括第一端口1031、第二端口1032、第三端口1033和第四端口1034。第一HMD 1035、第二HMD1036和第三HMD 1037全部可以附接至图形子系统1030的对应端口。图形子系统1030可以支持在第一HMD 1035、第二HMD 1036和第三HMD 1037中的每一个上的独立视图。

[0244] 现在转向图10C至图10D,这三个HMD可以同时观看360帧1040。例如,第一HMD 1035可以定位在第一视点A处,第二HMD 1036可以定位在第二视点B处,并且第三HMD 1037可以定位在第三视点C处。第一视点A和第二视点B可以包括第一重叠区1041。第一视点A和第三视点C可以包括第二重叠区1042。各个视点和重叠区可以在HMD改变位置时发生变化。

[0245] 一些实施例可以有利地针对用于VR应用的多个HMD提供社交360视频渲染。例如,一些实施例可以针对一个GPU进行多次解码以支持多个用户,其中,硬件解码可以基于每个用户的定向来渲染360度帧的视场(FOV)部分。多个硬件解码单元可以在社交VR的单个流(例如,360视频)上支持多个HMD。例如,一些实施例可以支持360视频和/或360抽动的多个视口。一些实施例可以通过共享重叠区解码/编码以避免在同一个区上多次执行解码/编码和/或基于重叠量而对解码进行优先级排序来更高效地实现单个流的360解码。

[0246] 一些系统可以为360视频创建大型矩形帧,对整个帧进行编码,并且然后基于用户正在看的点而提供帧的这个部分来向用户显示。如果第二用户正观看360视频,则一些系统可以在不参考第一用户的情况下基于第二用户正在看何处来对第二用户的视图进行解码,从而使得可以对各个区解码两次(例如,或者对于具有重叠视图的更多用户解码更多次)。一些实施例可以有利地识别多个用户的视点的重叠区以避免对同一个区解码多于一次。

[0247] 一些实施例可以向共享解码器提供360帧。共享解码器可以将用户的位置和定向

作为输入。第一用户的第一解码可以存储在帧缓冲器 (FB1) 中。对于第二用户,共享解码器可以考虑第二用户的位置和定向以便为第二用户提供存储在第二帧缓冲器 (FB2) 中的第二解码。共享解码器可以确定这两个帧缓冲器之间的重叠以避免重复解码。共享解码器可以基于重叠区的存在而对解码进行优先级排序,并且可以重用帧的之前经解码部分。如果多个区重叠,则共享解码器可以首先处理重叠区。如果没有人观看特定区,则共享解码器可能根本不对未观看区进行解码。

[0248] 在一些实施例中,主机系统(例如,PC、服务器等)可以包括存储360视频的存储设备。多个HMD可以有有线和/或无线地连接至主机系统。例如,主机系统可以包括具有多个GPU插槽的视频卡。主机系统可以附加地或可替代地提供网关或路由器以支持多个HMD。HMD的多个用户可以各自具有其自己的视图方向,并且主机系统可以针对每个用户使用共享和优先解码来执行360解码。在一些实施例中,视频可以在被提供至多个用户时被同步。例如,通过云流送,一些实施例可以使用多点来同步,因此用户可以具有更好的共享体验。

[0249] 有利地,一些实施例可以不传输整个360帧。一些实施例可以基于用户正在观看什么而选择性地解码。如果HMD通过电缆连接,则网络带宽可能不那么重要,但是一些实施例仍然可以通过选择性地解码和共享对重叠区的解码来节省处理带宽。对于无线HMD,一些实施例可以通过仅传输单个解码数据来节省网络拥塞和带宽。

[0250] 一些实施例可以使用多个解码块来对单个360帧进行解码。例如,GPU可以包括具有被配置成共享GPU中的重叠解码区的多个解码单元的架构。位置和定向引擎可以合并所有用户的位置和定向信息以确定重叠并将所述区提供至多个解码块。GPU可以拍摄360视频,将其分成多个解码、转码、组合,并且对每个HMD的信息进行重新编码。

[0251] 头戴式显示系统概述

[0252] 图11示出了由用户在体验沉浸式环境时戴着的头戴式显示器 (HMD) 系统1100,沉浸式环境诸如例如虚拟现实 (VR) 环境、增强现实 (AR) 环境、多玩家三维 (3D) 游戏等。在所展示的示例中,一个或多个带子1120将HMD系统1100的帧1102保持于用户的眼睛前面。相应地,放置左眼显示器1104以被用户的左眼查看并且放置右眼显示器1106以被用户的右眼查看。在诸如例如由用户戴着的智能电话的某些示例中,可择一地将左眼显示器1104和右眼显示器1106集成进单个显示器中。在AR的情况中,显示器1104、1106可以是透视显示器,透视显示器允许用户查看实体环境而其他经渲染的内容(例如,虚拟角色、信息化注释、平视显示器/HUD)呈现于实体环境的实时馈送的上方。

[0253] 在一个示例中,帧1102包括左下看相机1108以从一般位于用户前面和左眼下方的区域捕捉图像(例如,左手姿势)。此外,右下看相机1110可从一般位于用户前面和右眼下方的区域捕捉图像(例如,右手姿势)。所展示的帧1102还包括左前看相机1112和右前看相机1114以分别捕捉用户的左和右眼前面的图像。帧1102还可包括左侧看相机1116以从用户左侧的区域捕捉图像和包括右侧看相机1118以从用户右侧的区域捕捉图像。

[0254] 由可能有重叠视场的相机1108、1110、1112、1114、1116、1118捕捉的图像可用来检测由用户做出的姿势以及分析外部环境和/或在显示器1104、1106上再现外部环境。在一个示例中,所检测的姿势由(例如,内部的和/或外部的)图形处理架构用来渲染和/或控制3D游戏中用户的虚拟表示。事实上,重叠的视场可启用对其他个体(例如,在多玩家游戏中)做出的姿势的捕捉,其中其他个体的姿势还可用来渲染/控制沉浸式体验。重叠的视场还可使

HMD系统1100能够自动地检测用户附近的阻挡或其他损害。这样的方法在高级驾驶辅助系统(ADAS)应用中特别有利。

[0255] 在一个示例中,提供有重叠视场的左下看相机1108和右下看相机1110提供了具有增加的分辨率的立体视图。增加的分辨率可进而使非常相似的用户移动能够互相区分开来(例如,以亚-毫米的准确性)。结果可以是HMD系统1100关于可靠性的增强的性能。事实上,所展示的解决方案在各种各样的应用中是有用的,诸如例如给AR设置中的信息上色、在多用户环境中的多个用户之间交换虚拟工具/装置、渲染虚拟项目(例如,武器、剑、人员)等。其他对象、肢体和/或身体部分的姿势也可被检测并用于渲染/控制虚拟环境。例如,无论是从佩戴者还是从共享环境中的其他个体,都可以实时地跟踪脊髓造影信号、脑电波信号、眼睛跟踪、呼吸或喘气、手部运动等。由相机1108、1110、1112、1114、1116、1118捕捉的图像也可用作上下文输入。例如,可能确定用户正在指示文字处理应用中待编辑的特定词或待按的特定键,游戏中待部署的特定武器或行进方向等等。

[0256] 此外,由相机1108、1110、1112、1114、1116、1118捕捉的图像可用来在设备操作、医疗训练和/或远程/远端操作指导应用中实施共享的通信或网络化的互动。任务特定的姿势库或神经网络机器学习可以启用工具标识和对任务的反馈。例如,可启用转换成远程、真实动作的虚拟工具。在又一个示例中,HMD系统1100将虚拟场景内的虚拟钻头的操纵转换成为搜索倒塌的建筑物而部署的机器人装置上的钻头的远程操作。而且,HMD系统1100可以是可编程到包括了例如使用户能够向与用户动作相关联的可标识姿势列表中添加新姿势的协议的程度。

[0257] 另外,HMD 1100中的各种相机可以是可配置的,以在光谱的可见波长之外检测频谱频率。输入相机的多谱成像能力通过消除非必要的图像特征(例如,背景噪音)来允许对用户和/或对象的位置跟踪。例如,在诸如手术的增强现实(AR)应用中,仪器和设备通过他们的红外反射性得到跟踪而无需额外的跟踪辅助设备。而且,HMD 1100可以在低可见度的情形中采用,其中来自各种相机的“实时馈送”可以通过计算机分析来提高或增强并作为视觉或音频线索显示给用户。

[0258] HMD系统1100还可放弃执行任何类型的与远程计算系统的或者需要电源线(例如,独立操作模式)的数据通信。就此而言,HMD系统1100可以是具有能使HMD系统1100独立于外部电源系统而操作的电源单元的“无绳”装置。相应地,用户可以玩全特性游戏而无需栓系到另一个装置(例如,游戏控制台)或电源。在文字处理示例中,HMD系统1100在显示器1104和1106上呈现虚拟键盘和/或虚拟鼠标以提供虚拟桌面或文字处理场景。因此,由相机中的一个或多个捕捉的姿势标识数据表示虚拟键盘上的用户键入活动或虚拟鼠标的移动。优点包括但不限于:便携性和虚拟桌面隔绝附近个体的隐私性。底层的图形处理架构可支持视频和音频信号的压缩和/或解压缩。而且,对用户的左眼和右眼提供分开的图像可帮助3D场景的渲染、生成和/或感知。左眼显示器1104和右眼显示器1106的相对位置也可以是可调整的,以匹配不同用户之间眼间隔的变化。

[0259] 图11中所展示的相机的数量只是为了帮助讨论。事实上,取决于环境,HMD系统1100可包括少于六个或多于六个的相机。

[0260] HMD系统的功能性部件

[0261] 图12更详细地示出HMD系统。在所展示的示例中,帧1102包括向HMD系统提供电力

的电源单元1200 (例如, 电池电源、适配器)。所展示的帧1102还包括运动跟踪模块1220 (例如, 加速度计、陀螺仪), 其中运动跟踪模块1220向处理器系统1204提供运动跟踪数据、取向数据和/或位置数据。处理器系统1204可包括耦合至I/O桥1206的网络适配器1224。I/O桥1206可启用网络适配器1224和诸如例如音频输入模块1210、音频输出模块1208、显示装置1207、输入相机1202等等的各种部件之间的通信。

[0262] 在所展示的示例中, 音频输入模块1210包括右音频输入1218和左音频输入1216, 该两个输入检测为了标识用户以及附近个体的语音命令而可被处理的声音。在所捕捉的音频信号中标识的语音命令可在模态切换和其他应用期间增强姿势标识。而且, 所捕捉的音频信号可提供用来增强沉浸式体验的3D信息。

[0263] 音频输出模块1208可包括右音频输出1214和左音频输出1212。音频输出模块1208可将声音递送给用户和/或其他附近个体的耳朵。音频输出模块1208可以是耳塞、贴耳式扬声器、盖耳式扬声器、扩音器等或其任何组合的形式, 音频输出模块1208可将立体和/或3D音频内容递送给用户 (例如, 空间定位)。所展示的帧1102还包括无线模块1222, 该无线模块帮助HMD系统和各种其他系统 (例如, 计算机、可穿戴式装置、游戏控制台) 之间的通信。在一个示例中, 无线模块1222经由网络适配器1224与处理器系统1204通信。

[0264] 所展示的显示装置1207包括左眼显示器1104和右眼显示器1106, 其中呈现在显示器1104、1106上的虚拟内容可经由I/O桥1206从处理器系统1204获得。输入相机1202可包括已经讨论过的左侧看相机1116、右侧看相机1118、左下看相机1108、左前看相机1112、右前看相机1114和右下看相机1110。

[0265] 现在转向图13, 示出了通用处理集群 (GPC) 1300。所展示的GPC1300可合并入诸如例如已经讨论过的处理器系统1204 (图12) 的处理系统中。GPC 1300可包括与调度器通信的流水线管理器1302。在一个示例中, 流水线管理器1302从调度器接收任务并将任务分布给一个或多个串流多处理器 (SM) 1304。每一个SM 1304可被配置成处理线程组, 其中线程组可以被看作是对不同输入数据执行相同或相似操作的多个相关的线程。因此, 可将线程组中的每一个线程分配给特定SM 1304。在另一个示例中, 线程的数量可大于SM 1304中执行单元的数量。就此而言, 线程组中的多个线程可并行操作。流水线管理器1302还可将经处理的数据目的地指定给工作分布交叉开关1308, 该工作分布交叉开关与存储器交叉开关通信。

[0266] 因此, 在每一个SM 1304将经处理的任务发送给工作分布交叉开关1308时, 该经处理的任務可被提供给另一个GPC 1300供进一步处理。还可将SM 1304的输出发送到预光栅操作 (preROP) 单元1314, 该预光栅操作单元进而将数据导引至一个或多个光栅操作单元, 或者执行其他操作 (例如, 执行地址转换、组织图片颜色数据、混合颜色等等)。SM 1304可包括内部第一级 (L1) 高速缓存 (未示出), SM 1304可存储数据于其中。SM 1304还可具有经由存储器管理单元 (MMU) 1310到第二级 (L2) 高速缓存 (未示出) 的访问以及第一点五级 (L1.5) 高速缓存1306。MMU 1310可将虚拟地址映射到物理地址。就此而言, MMU 1310可包括页表条目 (PTE), 页表条目用来将虚拟地址映射到图块、存储器页和/或高速缓存线索引的物理地址。所展示的GPU1300包括纹理单元1312。

[0267] 图形流水线架构

[0268] 现在转向图14, 示出了图形流水线1400。在所展示的示例中, 世界空间流水线1420包括图元分布器 (PD) 1402。PD 1402可收集与高阶服务、图形图元、三角形等相关联的顶点

数据,并将顶点数据发送给顶点属性获取单元(VAF)1404。VAF 1404可从共享存储器获取与每一个传入顶点相关联的顶点属性并将顶点数据和相关联的顶点属性一起存储到共享存储器中。

[0269] 所展示的世界空间流水线1420还包括顶点、曲面细分、几何处理单元(VTG)1406。VTG 1406可包括例如顶点处理单元、曲面细分初始化处理单元、任务分布器、任务生成单元、拓扑生成单元、几何处理单元、曲面细分处理单元等或其任何组合。在一个示例中,VTG 1406是被配置成执行几何程序、曲面细分程序和顶点着色器程序的可编程执行单元。由VTG 1406执行的程序可处理从VAF 1404接收到的顶点数据和顶点属性。而且,由VTG 1406执行的程序可产生图形图元、颜色值、表面归一因子和图形图元的每个顶点处的透明度值以供在图形处理流水线1400内进一步处理。

[0270] VTG 1406的顶点处理单元可以是执行顶点着色器程序的可编程执行单元,按顶点着色器程序规定照明和变换顶点数据。例如,可将顶点处理单元编程为将顶点数据从基于对象的坐标表示(例如,对象空间)变换到诸如世界空间或归一化装置坐标(NDC)空间的替换性基于的坐标系。此外,顶点处理单元可读取由VAF 1404存储在共享存储器中的顶点数据和顶点属性,并且处理顶点数据和顶点属性。在一个示例中,顶点处理单元将经处理的顶点存储在共享存储器中。

[0271] 曲面细分初始化处理单元(例如,外壳着色器、曲面细分控制着色器)可执行曲面细分初始化着色器程序。在一个示例中,曲面细分初始化处理单元处理由顶点处理单元产生的顶点并生成有时称为“补片”的图形图元。曲面细分初始化处理单元还可生成各种补片属性,其中补片数据和补片属性被存储在共享存储器中。VTG 1406的任务生成单元可从共享存储器获取顶点和补片的数据和属性。在一个示例中,任务生成单元生成用于处理顶点和补片的任务,用于由图形处理流水线1400中的稍后级处理。

[0272] 由任务生成单元产生的任务可由VTG 1406的任务分布器重分布。例如,由顶点着色器程序和曲面细分初始化程序的各种实例产生的任务可在一个图形处理流水线1400和另一个图形处理流水线之间显著地不同。相应地,任务分布器可重分布这些任务以便每一个图形处理流水线1400在稍后流水线级有几乎相同的工作负荷。

[0273] 如已经论述的,VTG 1406还可包括拓扑生成单元。在一个示例中,拓扑生成单元获取由任务分布器分布的任务,将包括与补片相关联的顶点在内的顶点进行索引,并计算曲面细分顶点以及连接曲面细分顶点以形成图形图元的索引的坐标(UV)。经索引的顶点可由拓扑生成单元存储在共享存储器中。VTG 1406的曲面细分处理单元可被配置成执行曲面细分着色器程序(例如,域着色器、曲面细分评估着色器)。曲面细分处理单元可从共享存储器读取输入数据并将输出数据写到共享存储器中。输出数据可从共享存储器被传递到几何处理单元(例如下着色器级)作为输入数据。

[0274] VTG 1406的几何处理单元可执行几何着色器程序以变换图形图元(例如,三角形、线段、点等)。在一个示例中,将顶点分组以构建图形图元,其中几何处理单元将图形图元细分成一个或多个新图形图元。几何处理单元还可计算诸如例如可用来光栅化新图形图元的平面方程系数的参数。

[0275] 所展示的世界空间流水线1420还包括从VTG 1406获取规定新图形图元的参数和顶点的视口缩放、拣选和剪辑单元(VPC)1408。在一个示例中,VPC 1408执行剪辑、翻边、透

视修正和视口变换以标识在最终经渲染的图像中潜在地可以查看的图形图元。VPC 1408还可标识可能不可查看的图形图元。

[0276] 图形处理流水线1400还可包括耦合至世界空间流水线1420的分块(tiling)单元1410。分块单元1410可以是图形图元排序引擎,其中图形图元在世界空间流水线1420中被处理并且随后被发送给分块单元1410。就此而言,图形处理流水线1400还可包括屏幕空间流水线1422,其中屏幕空间可被分成高速缓存图块。每一个高速缓存图块可因此与屏幕空间的一部分相关联。对于每一个图形图元,分块单元1410可标识与图形图元相交(例如,分块)的一组高速缓存图块。在将数个图形图元分块以后,分块单元1410可逐高速缓存图块地处理图形图元。在一个示例中,与特定高速缓存图块相关联的图形图元每次一个图块地被发送给屏幕空间流水线1422中的设置单元1412。与多个高速缓存图块相交的图形图元可在世界空间流水线1420中被处理一次,而被发送到屏幕空间流水线1422多次。

[0277] 在一个示例中,设置单元1412经由分块单元1410从VPC 1408接收顶点数据并且计算与图形图元相关联的参数。参数可以包括例如边方程、偏平面方程和深度平面方程。屏幕空间流水线1422还可包括耦合至设置单元1412的光栅化器1414。光栅化器可扫描转换新图形图元并将片段和覆盖数据发送到像素着色单元(PS) 1416。光栅化器1414还可执行Z拣选和其他基于Z的优化。

[0278] 可访问共享存储器的PS 1416可执行对从光栅化器1414接收的片段进行变换的片段着色器程序。更具体地,片段着色器程序可以像素级粒度对片段着色(例如作为像素着色器程序工作)。在另一个示例中,片段着色器程序以样本级粒度对片段着色,其中每一个像素包括多个样本,且每一个样本表示像素的一部分。而且,取决于环境(例如采样率),片段着色器程序可以任何其他粒度对片段着色。PS 1416可执行混色、着色、透视修正、纹理映射等以生成经着色的片段。

[0279] 所展示的屏幕空间流水线1422还包括光栅操作单元(ROP) 1418,其可执行诸如例如模板印刷、Z测试、混合等等。ROP 1418随后可将像素数据作为经处理的图形数据发送給一个或多个经渲染的目标(例如,图形存储器)。ROP 1418可被配置成压缩写到存储器的Z或颜色数据并且解压缩从存储器读取的Z或颜色数据。ROP 1418的位置可取决于环境而变化。

[0280] 图形处理流水线1400可由一个或多个处理元件实现。例如,VTG1406和/或PS 1416可实现于一个或多个SM中,PD 1402、VAF 1408、分块单元1410、设置单元1412、光栅化器1414和/或ROP 1418可在结合相对应的分区单元的具体GPC的处理元件中实现。图形处理流水线1400还可在固定功能硬件逻辑中实现。事实上,图形处理流水线1400可在PPU中实现。

[0281] 因此,所展示的世界空间流水线1420处理3D空间中的图形对象,其中已知每一个图形对象相对于其他图形对象以及相对于3D坐标系的位置。相反,屏幕空间流水线1422可处理已经从3D坐标系投射到表示显示装置的表面的2D平面表面上的图形对象。此外,可将世界空间流水线1420划分为阿尔法阶段流水线和贝塔阶段流水线,其中阿尔法阶段流水线包括从PD 1402直到任务生成单元的流水线级。贝塔阶段流水线包括从拓扑生成单元直到VPC1408的流水线级。在这样的情况中,图形处理流水线1400可在阿尔法阶段流水线中执行一组第一操作(例如,单个线程、线程组、一致行动的多个线程组)并在贝塔阶段流水线中执行一组第二操作(例如,单个线程、线程组、一致行动的多个线程组)。

[0282] 如果多个图形流水线1400在使用中,则可将与一组图形对象相关联的顶点数据和

顶点属性划分以便每一个图形处理流水线1400贯穿阿尔法阶段有相似的工作负荷。相应地,阿尔法阶段处理可以实质地扩展顶点数据和顶点属性的数量,从而由任务生成单元产生的顶点数据和顶点属性的数量显著地大于由PD 1402和VAF 1404处理的顶点数据和顶点属性的数量。而且,与不同的图形处理流水线1400相关联的任务生成单元可产生具有不同质量等级的顶点数据和顶点属性,即使用相同数量的属性开始阿尔法阶段。在这样的情况中,任务分布器可重分布由阿尔法阶段流水线产生的属性,从而每一个图形处理流水线1400在贝塔阶段流水线的开始有大致相同的工作负荷。

[0283] 现在转向图15,示出了串流多处理器(SM)1500。所展示的SM 1500包括耦合至指令高速缓存1502的K调度器单元1504,其中每一个调度器单元1504从流水线管理器(未示出)接收线程块阵列,并管理每一个活跃线程块阵列中的一个或多个线程块的指令调度。调度器单元1504可调度线程用于按并行线程组执行,其中每一个组可称为“线程束(warp)”。因此,每一个线程束可能包括例如64个线程。此外,调度器单元1504可管理多个不同的线程块,将线程块分配给线程束用于执行。调度器单元可随后在每一个时钟周期期间在各种功能单元上调度来自多个不同线程束的指令。每一个调度器单元1504可包括一个或多个指令分派单元1522,其中每一个分派单元1522向功能单元中的一个或多个发送指令。分派单元1522的数量可取决于环境而变化。在所展示的示例中,调度器单元1504包括使来自同一线程束的两个不同指令在每个时钟周期期间待分派的两个分派单元1522。

[0284] SM 1500还可以包括寄存器堆1506。寄存器堆1506包括一组寄存器,该组寄存器在功能单元之间进行划分使得每一个功能单元被分配到寄存器堆1506的专用部分。寄存器堆1506还可在正由SM 1500执行的不同的线程束之间进行划分。在一个示例中,寄存器堆1506为连接到功能单元的数据路径的操作数提供临时存储。所展示的SM 1500还包括L个处理核1508,其中L可以是相对较大的数(例如192)。每一个核1508都可以是包括浮点算术逻辑单元(例如,IEEE 754-2008)和整数算术逻辑单元的流水线单精度处理单元。

[0285] 所展示的SM 1500还包括M个双精度单元(DPU)1510、N个特殊功能单元(SFU)1512和P个加载/存储单元(LSU)1514。每一个DPU 1510都可实现双精度浮点算术且每一个SFU 1512都可执行诸如例如矩形复制像素混合等的特殊功能。此外,每一个LSU 1514都可在共享存储器1518与寄存器堆1506之间实施加载和存储操作。在一个示例中,加载和存储操作是通过J个纹理单元/L1高速缓存1520和互连网络1516来实施的。在一个示例中,J个纹理单元/L1高速缓存1520还耦合至交叉开关(未示出)。因此,互连网络1516可将功能单元的每一个连接到寄存器堆1506和共享存储器1518。在一个示例中,互连网络1516作为将功能单元中的任何一个连接到寄存器堆1506中的任何寄存器的交叉开关。

[0286] SM 1500可在图形处理器(例如,图形处理单元/GPU)内实现,其中纹理单元/L1高速缓存1520可访问来自存储器的纹理映射并对纹理映射采样以产生经采样的纹理值以用于着色器程序中。由纹理单元/L1高速缓存执行的纹理操作包括但不限于基于mip图的抗混叠。

[0287] 附加的系统概述示例

[0288] 图16是根据实施例的处理系统1600的框图。在各种实施例中,系统1600包括一个或多个处理器1602和一个或多个图形处理器1608,并且可以是单处理器台式计算机系统、多处理器工作站系统或具有大量处理器1602或处理器核1607的服务器系统。在一个实施例

中,系统1600是包括在系统芯片(SoC)中的处理平台以供用于移动装置、手持式装置或嵌入式装置中。

[0289] 系统1600的实施例可以包括以下各者或可以包括在以下各者内:基于服务器的游戏平台、游戏控制台(包括游戏和媒体控制台)、移动游戏控制台、手持式游戏控制台或在线游戏控制台。在一些实施例中,系统1600是移动电话、智能电话、平板计算装置或移动互联网装置。数据处理系统1600还可以包括以下各者、与以下各者耦合或被集成在以下各者中:穿戴式装置,比如智能手表穿戴式装置、智能眼镜装置、增强现实装置或虚拟显示装置。在一些实施例中,数据处理系统1600是电视或机顶盒装置,其具有一个或多个处理器1602和由一个或多个图形处理器1608产生的图形接口。

[0290] 在一些实施例中,一个或多个处理器1602各自包括用于处理指令的一个或多个处理器核1607,这些指令在被执行时执行系统和用户软件的操作。在一些实施例中,一个或多个处理器核1607中的每一者被配置成处理具体的指令集1609。在一些实施例中,指令集1609可以促进复杂指令集计算(CISC)、精简指令集计算(RISC)、或经由超长指令字(VLIW)的计算。多个处理器核1607可以各自处理不同的指令集1609,所述指令集可以包括用于促进对其他指令集进行仿真的指令。处理器核1607还可包括其他处理装置,比如数字信号处理器(DSP)。

[0291] 在一些实施例中,处理器1602包括高速缓存存储器1604。取决于架构,处理器1602可以具有单个内部高速缓存或多级内部高速缓存。在一些实施例中,在处理器1602的各种部件当中共享高速缓存存储器。在一些实施例中,处理器1602还使用外部高速缓存(例如,3级(L3)高速缓存或最后一级高速缓存(LLC)(未示出),可使用已知的高速缓存一致性技术在处理器核1607之间共享所述外部高速缓存。寄存器堆1606被另外包括在处理器1602中,所述寄存器堆可包括用于存储不同类型的数据的不同类型的寄存器(例如,整数寄存器、浮点寄存器、状态寄存器和指令指针寄存器)。一些寄存器可以是通用寄存器,而其他寄存器可以是特定于处理器1602的设计的。

[0292] 在一些实施例中,处理器1602耦合至处理器总线1610,以在处理器1602与系统1600中的其他部件之间传输通信信号(比如,地址、数据或控制信号)。在一个实施例中,系统1600使用示例性‘中枢’系统架构,包括存储器控制器中枢1616和输入输出(I/O)控制器中枢1630。存储器控制器中枢1616促进存储器装置与系统1600的其他部件之间的通信,而I/O控制器中枢(ICH)1630经由本地I/O总线来提供至I/O装置的连接。在一个实施例中,存储器控制器中枢1616的逻辑被集成在处理器内。

[0293] 存储器装置1620可以是动态随机存取存储器(DRAM)装置、静态随机存取存储器(SRAM)装置、闪存装置、相变存储器装置或具有合适的性能以充当进程存储器的某一其他存储器装置。在一个实施例中,存储器装置1620可以作为系统1600的系统存储器来操作,以存储数据1622和供在一个或多个处理器1602执行应用或进程时使用的指令1621。存储器控制器中枢1616也与可选的外部图形处理器1612耦合,所述外部图形处理器可与处理器1602中的图形处理器1608耦合,以执行图形和媒体操作。

[0294] 在一些实施例中,ICH 1630使得能够经由高速I/O总线将外围装置连接到存储器装置1620和处理器1602。I/O外围装置包括但不限于:音频控制器1646、固件接口1628、无线收发机1626(例如,Wi-Fi、蓝牙)、数据存储装置1624(例如,硬盘驱动器、闪存等)和用于将

传统(例如,个人系统2(PS/2))装置耦合至系统的传统I/O控制器1640。一个或多个通用串行总线(USB)控制器1642连接输入装置(比如,键盘和鼠标1644组合)。网络控制器1634还可以耦合至ICH 1630。在一些实施例中,高性能网络控制器(未示出)耦合至处理器总线1610。将认识到的是,所示出的系统1600是示例性的而非限制性的,因为还可以使用以不同方式配置的其他类型的数据处理系统。例如,I/O控制器中枢1630可以集成在所述一个或多个处理器1602内,或者存储器控制器中枢1616和I/O控制器中枢1630可以集成在分立式外部图形处理器(比如外部图形处理器1612)内。

[0295] 图17是处理器1700的实施例的框图,所述处理器具有一个或多个处理器核1702A至1702N、集成式存储器控制器1714和集成式图形处理器1708。图17中具有与本文任何其他图的元件相同的参考数字(或名称)的那些元件可以以与本文别处描述的方式类似的任何方式来操作或起作用,但并不限于此。处理器1700可包括多达且包括由虚线框表示的附加核1702N的附加核。处理器核1702A至1702N中的每一者包括一个或多个内部高速缓存单元1704A至1704N。在一些实施例中,每个处理器核还能够访问一个或多个共享高速缓存单元1706。

[0296] 内部高速缓存单元1704A至1704N和共享高速缓存单元1706表示处理器1700内的高速缓存存储器层级结构。高速缓存存储器层级结构可包括每个处理器核内的至少一级指令和数据高速缓存以及共享中间级高速缓存的一个或多个级(比如,2级(L2)、3级(L3)、4级(L4)或其他级高速缓存),其中,在外部存储器前面的最高级高速缓存被归类为LLC。在一些实施例中,高速缓存一致性逻辑保持各种高速缓存单元1706和1704A至1704N之间的一致性。

[0297] 在一些实施例中,处理器1700还可包括一组一个或多个总线控制器单元1716以及系统代理核1710。所述一个或多个总线控制器单元1716管理一组外围总线,比如一个或多个外围部件互连总线(例如,PCI、PCI快速总线)。系统代理核1710提供对各处理器部件的管理功能。在一些实施例中,系统代理核1710包括一个或多个集成式存储器控制器1714,所述集成式存储器控制器用于管理对各种外部存储器装置(未示出)的访问。

[0298] 在一些实施例中,处理器核1702A至1702N中的一者或多者包括对同时多线程处理的支持。在这样的实施例中,系统代理核1710包括用于在多线程处理期间协调和操作核1702A至1702N的部件。系统代理核1710可另外包括功率控制单元(PCU),所述PCU包括用于调节处理器核1702A至1702N和图形处理器1708的功率状态的逻辑和部件。

[0299] 在一些实施例中,处理器1700另外包括用于执行图形处理操作的图形处理器1708。在一些实施例中,图形处理器1708与一组共享高速缓存单元1706和系统代理核1710包括一个或多个集成式存储器控制器1714耦合。在一些实施例中,显示控制器1711与图形处理器1708耦合以便将图形处理器输出驱动到一个或多个耦合的显示器。在一些实施例中,显示控制器1711可以是经由至少一个互连与图形处理器耦合的单独模块,或者可以集成在图形处理器1708或系统代理核1710内。

[0300] 在一些实施例中,使用基于环形的互连单元1712来耦合处理器1700的内部部件。然而,可以使用替代性互连单元,比如点到点互连、切换式互连、或其他技术,包括本领域众所周知的技术。在一些实施例中,图形处理器1708经由I/O链路1713与环形互连1712耦合。

[0301] 示例性I/O链路1713表示多种I/O互连中的至少一者,包括促进各种处理器部件与

高性能嵌入式存储器模块1718(比如eDRAM模块)之间的通信的封装(on package) I/O互连。在一些实施例中,处理器核1702至1702N中的每一者和图形处理器1708将嵌入式存储器模块1718用作共享的最后一级高速缓存。

[0302] 在一些实施例中,处理器核1702A至1702N是执行相同的指令集架构的同质核。在另一个实施例中,处理器核1702A至1702N就指令集架构(ISA)而言是异质的,其中,处理器核1702A至1702N中的一者或多者执行第一指令集,而其他核中的至少一者执行第一指令集的子集或不同的指令值。在一个实施例中,处理器核1702A至1702N就微架构而言是异质的,其中,具有相对更高功率消耗的一个或多个核与具有更低功率消耗的一个或多个功率核耦合。另外,处理器1700可以实现在一个或多个芯片上或者被实现为具有除其他部件之外的所展示的部件的SoC集成电路。

[0303] 图18是图形处理器1800的框图,所述图形处理器可以是分立的图形处理单元,或可以是与多个处理核集成的图形处理器。在一些实施例中,图形处理器经由到图形处理器上的寄存器的映射I/O接口并且利用被放置在处理器存储器中的命令与存储器进行通信。在一些实施例中,图形处理器1800包括用于访问存储器的存储器接口1814。存储器接口1814可以是到本地存储器、一个或多个内部高速缓存、一个或多个共享外部高速缓存、和/或到系统存储器的接口。

[0304] 在一些实施例中,图形处理器1800还包括用于将显示输出数据驱动到显示装置1820的显示控制器1802。显示控制器1802包括用于显示器的一个或多个重叠平面的硬件以及多层视频或用户接口元件的组成。在一些实施例中,图形处理器1800包括用于编码、解码、或者向、从或在一个或多个媒体编码格式之间进行媒体代码转换的视频编解码器引擎1806,包括但不限于:运动图像专家组(MPEG)格式(比如MPEG-2)、高级视频译码(AVC)格式(比如H.264/MPEG-4AVC)、以及电影&电视工程师协会(SMPTE) 421M/VC-1、和联合图像专家组(JPEG)格式(比如JPEG、以及运动JPEG(MJPEG)格式)。

[0305] 在一些实施例中,图形处理器1800包括用于执行二维(2D)光栅化器操作的块图像传输(BLIT)引擎1804,所述2D光栅化器操作包括(例如)位边界块传输。然而,在一个实施例中,使用图形处理引擎(GPE) 1810的一个或多个部件执行2D图形操作。在一些实施例中,图形处理引擎1810是用于执行图形操作的计算引擎,所述图形操作包括三维(3D)图形操作和媒体操作。

[0306] 在一些实施例中,GPE 1810包括用于执行3D操作的3D流水线1812,所述3D操作为比如使用作用于3D图元形状(例如,矩形、三角形等)的处理功能来渲染三维图像和场景。3D流水线1812包括在元件和/或生成的执行线程内向3D/媒体子系统1815执行各种任务的可编程和固定功能元件。虽然3D流水线1812可以用于执行媒体操作,但是GPE 1810的实施例还包括媒体流水线1816,所述媒体流水线具体地用于执行媒体操作,比如视频后处理和图像增强。

[0307] 在一些实施例中,媒体流水线1816包括用于代替或代表视频编解码器引擎1806执行一个或多个专门的媒体操作的固定功能或可编程逻辑单元,所述专门的媒体操作为比如视频解码加速、视频解交织和视频编码加速。在一些实施例中,媒体流水线1816另外包括线程生成单元以便生成用于在3D/媒体子系统1815上执行的线程。所生成的线程对3D/媒体子系统1815中所包括的一个或多个图形执行单元执行对媒体操作的计算。

[0308] 在一些实施例中,3D/媒体子系统1815包括用于执行由3D流水线1812和媒体流水线1816生成的线程的逻辑。在一个实施例中,流水线向3D/媒体子系统1815发送线程执行请求,所述3D/媒体子系统包括用于仲裁并将各请求分派到可用的线程执行资源的线程分派逻辑。执行资源包括用于处理3D和媒体线程的图形执行单元的阵列。在一些实施例中,3D/媒体子系统1815包括用于线程指令和数据的一个或多个内部高速缓存。在一些实施例中,所述子系统还包括共享存储器(包括寄存器和可寻址存储器),以便在线程之间共享数据并存储输出数据。

[0309] 3D/媒体处理

[0310] 图19是根据一些实施例的图形处理器的图形处理引擎1910的框图。在一个实施例中,GPE 1910是图18中所示的GPE 1810的一个版本。图19中具有与本文任何其他图的元件相同的参考数字(或名称)的元件可以以与本文别处描述的方式类似的任何方式来操作或起作用,但并不限于此。

[0311] 在一些实施例中,GPE 1910与命令流转化器1903耦合,所述命令流转化器将命令流提供给GPE的3D流水线1912和媒体流水线1916。在一些实施例中,命令流转化器1903耦合至存储器,所述存储器可以是系统存储器,或可以是内部高速缓存存储器和共享高速缓存存储器中的一者或多者。在一些实施例中,命令流转化器1903从存储器接收命令,并且将命令发送给3D流水线1912和/或媒体流水线1916。所述命令是从存储用于3D流水线1912和媒体流水线1916的环形缓冲器获取的指示。在一个实施例中,所述环形缓冲器可另外包括存储多批多命令的批命令缓冲器。3D流水线1912和媒体流水线1916通过经由各自流水线内的逻辑执行操作或者通过将一个或多个执行线程分派至执行单元阵列1914来处理所述命令。在一些实施例中,执行单元阵列1914是可缩放的,使得所述阵列基于GPE 1910的目标功率和性能级别而包括可变数目的执行单元。

[0312] 在一些实施例中,采样引擎1930与存储器(例如,高速缓存存储器或系统存储器)以及执行单元阵列1914耦合。在一些实施例中,采样引擎1930为执行单元阵列1914提供存储器访问机制,所述存储器访问机制允许执行阵列1914从存储器读取图形和媒体数据。在一些实施例中,采样引擎1930包括用于执行针对媒体的专门图像采样操作的逻辑。

[0313] 在一些实施例中,采样引擎1930中的专门的媒体采样逻辑包括去噪/解交织模块1932、运动估计模块1934以及图像缩放和过滤模块1936。在一些实施例中,去噪/解交织模块1932包括用于对经解码的视频数据执行去噪或解交织算法中的一者或多者的逻辑。解交织逻辑将经交织的视频内容的交变长组合为视频的单个帧。去噪逻辑从视频和图像数据减少或去除数据噪声。在一些实施例中,所述去噪和解交织逻辑是运动自适应的并且使用基于在视频数据中检测到的运动量的空间或时间过滤。在一些实施例中,去噪/解交织模块1932包括专门的运动检测逻辑(例如,在运动估计引擎1934内)。

[0314] 在一些实施例中,运动估计引擎1934通过对视频数据执行视频加速度函数(诸如,运动向量估计和预测)来提供对视频操作的硬件加速度。运动估计引擎确定描述连续视频帧之间的图像数据变换的运动向量。在一些实施例中,图形处理器媒体编解码器使用视频运动估计引擎1934来对宏块级视频执行操作,对于其利用通用处理器来执行可以另外地是太计算密集型的。在一些实施例中,运动估计引擎1934通常可用于图形处理器部件以便辅助视频解码和处理功能,所述视频解码和处理功能对于视频数据内的运动的方向或幅度是

敏感或自适应的。

[0315] 在一些实施例中,图像缩放和过滤模块1936执行图像处理操作,以提高所产生的图像和视频的视觉质量。在一些实施例中,缩放和过滤模块1936在向执行单元阵列1914提供数据之前在采样操作期间处理图像和视频数据。

[0316] 在一些实施例中,GPE 1910包括数据端口1944,所述数据端口提供用于使图形子系统访问存储器的附加机制。在一些实施例中,数据端口1944针对操作促进存储器访问,所述操作包括渲染目标写入、恒定缓冲器读取、暂时存储器空间读取/写入、和媒体表面访问。在一些实施例中,数据端口1944包括用于高速缓存对存储器的访问的高速缓存存储器空间。高速缓存存储器可以是单个数据高速缓存,或被分离成用于经由数据端口来访问存储器的多个子系统的多个高速缓存(例如,渲染缓冲高速缓存、恒定缓冲器高速缓存等)。在一些实施例中,执行在执行单元阵列1914中的执行单元上的线程通过经由数据分布互连交换消息来与数据端口通信,所述数据分布互连耦合GPE 1910的每个子系统。

[0317] 执行单元

[0318] 图20是图形处理器2000的另一个实施例的框图。图20中具有与本文任何其他图的元件相同的参考数字(或名称)的元件可以以与本文别处描述的方式类似的任何方式来操作或起作用,但并不限于此。

[0319] 在一些实施例中,图形处理器2000包括环形互连2002、流水线前端2004、媒体引擎2037和图形核2080A至2080N。在一些实施例中,环形互连2002将图形处理器耦合至其他处理单元,包括其他图形处理器或者一个或多个通用处理器核。在一些实施例中,图形处理器是集成在多核处理系统内的多个处理器之一。

[0320] 在一些实施例中,图形处理器2000经由环形互连2002接收多批命令。由流水线前端2004中的命令流转化器2003翻译传入的命令。在一些实施例中,图形处理器2000包括用于经由图形核2080A至2080N来执行3D几何处理和媒体处理的可缩放执行逻辑。针对3D几何处理命令,命令流转化器2003将命令供应给几何流水线2036。针对至少一些媒体处理命令,命令流转化器2003将命令供应给视频前端2034,所述视频前端与媒体引擎2037耦合。在一些实施例中,媒体引擎2037包括用于视频和图像后处理的视频质量引擎(VQE) 2030以及用于提供硬件加速的媒体数据编码和解码的多格式编码/解码(MFX) 2033引擎。在一些实施例中,几何流水线2036和媒体引擎2037各自生成执行线程,所述执行线程用于由至少一个图形核2080A提供的线程执行资源。

[0321] 在一些实施例中,图形处理器2000包括以模块化核2080A至2080N(有时称为核切片)为特征的可缩放线程执行资源,每个模块化核具有多个子核2050A至2050N、2060A至2060N(有时称为核子切片)。在一些实施例中,图形处理器2000可以具有任意数量的图形核2080A至2080N。在一些实施例中,图形处理器2000包括图形核2080A,所述图形核至少具有第一子核2050A和第二子核2060A。在其他实施例中,图形处理器是具有单个子核(例如,2050A)的低功率处理器。在一些实施例中,图形处理器2000包括多个图形核2080A至2080N,每个图形核包括一组第一子核2050A至2050N和一组第二子核2060A至2060N。所述一组第一子核2050A至2050N中的每个子核至少包括第一组执行单元2052A至2052N和媒体/纹理采样器2054A至2054N。所述一组第二子核2060A至2060N中的每个子核至少包括第二组执行单元2062A至2062N和采样器2064A至2064N。在一些实施例中,每个子核2050A至2050N、2060A-

2060N共享一组共享资源2070A至2070N。在一些实施例中,这些共享资源包括共享高速缓存存储器和像素操作逻辑。其他共享资源也可包括在图形处理器的各种实施例中。

[0322] 图21展示了可线程执行逻辑2100,包括在GPE的一些实施例中所采用的处理元件的阵列。图21中具有与本文任何其他图的元件相同的参考数字(或名称)的那些元件可以以与本文别处描述的方式类似的任何方式来操作或起作用,但并不限于此。

[0323] 在一些实施例中,线程执行逻辑2100包括像素着色器2102、线程分派器2104、指令高速缓存2106、可缩放执行单元阵列,包括多个执行单元2108A至2108N、采样器2110、数据高速缓存2112和数据端口2114。在一个实施例中,这些所包括的部件经由互连结构而互连,所述互连结构链接到这些部件中每一者。在一些实施例中,通过指令高速缓存2106、数据端口2114、采样器2110和执行单元阵列2108A至2108N中的一者,线程执行逻辑2100包括至存储器(比如,系统存储器或高速缓存存储器)的一个或多个连接。在一些实施例中,每个执行单元(例如,2108A)是个别向量处理器,能够执行多个同时的线程并且针对每个线程来并行处理多个数据元素。在一些实施例中,执行单元阵列2108A至2108N包括任何数目的个别执行单元。

[0324] 在一些实施例中,执行单元阵列2108A至2108N主要用于执行“着色器”程序。在一些实施例中,阵列2108A至2108N中的执行单元执行包括对许多标准3D图形着色器指令的原生支持的指令集,使得以最小的转换执行来自图形库(例如,直接3D和OpenGL)的着色器程序。执行单元支持顶点和几何处理(例如,顶点程序、几何程序、顶点着色器)、像素处理(例如,像素着色器、片段着色器)和通用处理(例如,计算和媒体着色器)。

[0325] 执行单元阵列2108A至2108N中的每个执行单元对数据元素的阵列进行操作。数据元素的数目是“执行大小”或用于指令的通道的数目。执行通道是用于数据元素访问、掩码和指令内的流控制的逻辑执行单元。通道的数目可以与针对特定图形处理器的物理算术逻辑单元(ALU)或浮点单元(FPU)的数目无关。在一些实施例中,执行单元2108A至2108N支持整数和浮点数据类型。

[0326] 执行单元指令集包括单指令多数据(SIMD)。可以将各种数据元素作为压缩数据类型存储在寄存器中,并且执行单元将基于各种元素的数据大小来处理这些元素。例如,当在256位宽的向量上进行操作时,所述256位的向量存储在寄存器中,并且所述执行单元作为四个单独64位压缩数据元素(四倍字长(QW)大小的数据元素)、八个单独32位压缩数据元素(双倍字长(DW)大小的数据元素)、十六个单独16位压缩数据元素(字长(W)大小的数据元素)、或三十二个单独8位数据元素(字节(B)大小的数据元素)在所述向量上进行操作。然而,不同的向量宽度和寄存器大小是可能的。

[0327] 一个或多个内部指令高速缓存(例如,2106)被包括在线程执行逻辑2100中,以高速缓存用于执行单元的线程指令。在一些实施例中,一个或多个数据高速缓存(例如,2112)被包括用于高速缓存在线程执行期间的线程数据。在一些实施例中,采样器2110被包括用于为3D操作提供纹理采样并且为媒体操作提供媒体采样。在一些实施例中,采样器2110包括专门的纹理或媒体采样功能,以便在向执行单元提供采样数据之前在采样过程期间处理纹理或媒体数据。

[0328] 在执行期间,图形流水线和媒体流水线经由线程生成和分派逻辑将线程发起请求发送给线程执行逻辑2100。在一些实施例中,线程执行逻辑2100包括本地线程分派器2104,

所述本地线程分派器仲裁来自图形流水线和媒体流水线的线程发起请求并在一个或多个执行单元2108A至2108N上实例化所请求的线程。例如,几何流水线(例如,图20的2036)将顶点处理、曲面细分或几何处理线程分派给线程执行逻辑2100(图21)。在一些实施例中,线程分派器2104还可以处理来自执行着色器程序的运行时间线程生成请求。

[0329] 一旦一组几何对象已被处理并被光栅化为像素数据,就调用像素着色器2102以进一步计算输出信息并导致将结果写入到输出表面(例如,颜色缓冲器、深度缓冲器、模板印刷缓冲器等)。在一些实施例中,像素着色器2102计算各顶点属性的值,所述各顶点属性跨栅格化对象被内插。在一些实施例中,像素着色器2102然后执行应用编程接口(API)供应的像素着色器程序。为了执行所述像素着色器程序,像素着色器2102经由线程分派器2104将线程分派给执行单元(例如,2108A)。在一些实施例中,像素着色器2102使用采样器2110中的纹理采样逻辑来访问存储器中所存储的纹理图中的纹理数据。对纹理数据和输入几何进行的算术运算计算每个几何片段的像素颜色数据,或放弃一个或多个像素以供进一步处理。

[0330] 在一些实施例中,数据端口2114提供用于使线程执行逻辑2100将已处理的数据输出到存储器以供在图形处理器输出流水线上处理的存储器访问机制。在一些实施例中,数据端口2114包括或耦合至一个或多个高速缓存存储器(例如,数据高速缓存2112)从而经由数据端口高速缓存数据以供存储器访问。

[0331] 图22是根据一些实施例的示意图形处理器指令格式2200的框图。在一个或多个实施例中,图形处理器执行单元支持具有多种格式的指令的指令集。实线框展示通常包括在执行单元指令中的分量,而虚线包括可选的或仅包括在指令的子集中的分量。在一些实施例中,所描述和展示的指令格式2200是宏指令,因为它们是供应至执行单元的指令,这与从指令解码产生的微操作相反(一旦所述指令被处理)。

[0332] 在一些实施例中,图形处理器执行单元原生地支持呈128位格式2210的指令。64位紧凑指令格式2230可用于基于所选的指令、指令选项和操作数的数目的一些指令。原生128位格式2210提供对所有指令选项的访问,而一些选项和操作在64位格式2230中则被限制。以64位格式2230可用的原生指令根据实施例而变化。在一些实施例中,使用索引字段2213中的一组索引值将指令部分地紧凑。执行单元硬件基于这些索引值参考一组压缩表,并且使用压缩表输出来以128位格式2210重建原生指令。

[0333] 对每种格式,指令操作码2212定义执行单元要执行的操作。执行单元跨越每个操作数的多个数据元素并行执行每个指令。例如,响应于加法指令,执行单元跨越表示纹理元素或图片元素的每个颜色通道来执行同时加法运算。默认情况下,执行单元跨越操作数的所有数据通道执行每个指令。在一些实施例中,指令控制字段2214使得能控制某些执行选项,比如通道选择(例如,预测)以及数据通道排序(例如,混合)。针对128位指令2210,执行大小字段2216限制将被并行执行的数据通道的数目。在一些实施例中,执行大小字段2216不可用于64位紧凑指令格式2230。

[0334] 一些执行单元指令具有多达三个操作数,包括两个源操作数src0 2220、src1 2222和一个目的地2218。在一些实施例中,执行单元支持双目的地指令,其中,这些目的地之一是隐式的。数据操纵指令可以具有第三源操作数(例如, SRC2 2224),其中,指令操作码2212确定源操作数的数目。指令的最后一个源操作数可以通过所述指令传递的立即(例如,硬

编码) 值。

[0335] 在一些实施例中,128位指令格式2210包括访问/地址模式信息2226,所述访问/地址模式信息指定(例如)使用直接寄存器寻址模式还是间接寄存器寻址模式。当使用直接寄存器寻址模式时,直接由指令2210中的位来提供一个或多个操作数的寄存器地址。

[0336] 在一些实施例中,128位指令格式2210包括访问/地址模式字段2226,所述访问/地址模式字段指定所述指令的地址模式和/或访问模式。在一个实施例中,访问模式定义所述指令的数据访问对齐。一些实施例支持包括16字节对齐访问模式和1字节对齐访问模式的访问模式,其中,访问模式的字节对齐确定指令操作数的访问对齐。例如,当处于第一模式时,指令2210可针对源操作数和目的地操作数使用字节对齐寻址,并且当处于第二模式时,指令2210可针对所有的源操作数和目的地操作数使用16字节对齐寻址。

[0337] 在一个实施例中,访问/地址模式字段2226的地址模式部分确定指令将使用直接寻址还是间接寻址。当使用直接寄存器寻址模式时,指令2210中的位直接提供一个或多个操作数的寄存器地址。当使用间接寄存器寻址模式时,可基于地址寄存器值和指令中的地址立即字段来计算一个或多个操作数的寄存器地址。

[0338] 在一些实施例中,基于操作码2212位字段对指令分组,以简化操作码解码2240。针对8位操作码,位4、5和6允许执行单元确定操作码的类型。所示出的精确操作码分组仅是示例性的。在一些实施例中,移动和逻辑操作码组2242包括数据移动和逻辑指令(例如,移动(mov)、比较(cmp))。在一些实施例中,移动和逻辑组2242共享五个最高有效位(MSB),其中,移动(mov)指令采用0000xxxxb的形式,而逻辑指令采用0001xxxxb的形式。流控制指令组2244(例如,调用(call)、跳(jmp))包括采用0010xxxxb形式(例如,0x20)的指令。混杂指令组2246包括指令的混合体,这些指令包括采用0011xxxxb形式(例如,0x30)的同步指令(例如,等待、发送)。并行数学指令组2248包括采用0100xxxxb形式(例如,0x40)的按分量逐个作出的(component-wise)算术指令(例如,加(add)、减(sub))。并行数学组2248跨越数据通道并行执行算术运算。向量数学组2250包括采用0101xxxxb形式(例如,0x50)的算术指令(例如,dp4)。向量数学组执行比如对向量操作数的点积计算的算术。

[0339] 图形流水线

[0340] 图23是图形处理器2300的另一个实施例的框图。图23中具有与本文任何其他图的元件相同的参考数字(或名称)的元件可以以与本文别处描述的方式类似的任何方式来操作或起作用,但并不限于此。

[0341] 在一些实施例中,图形处理器2300包括图形流水线2320、媒体流水线2330、显示引擎2340、线程执行逻辑2350和渲染输出流水线2370。在一些实施例中,图形处理器2300是包括一个或多个通用处理核的多核处理系统内的图形处理器。图形处理器受到至一个或多个控制寄存器(未示出)的寄存器写入的控制或者经由环形互连2302经由发布至图形处理器2300的命令被控制。在一些实施例中,环形互连2302将图形处理器2300耦合至其他处理部件,比如其他图形处理器或通用处理器。由命令流转化器2303翻译来自环形互连2302的命令,所述命令流转化器将指令供应给图形流水线2320或媒体流水线2330的个别部件。

[0342] 在一些实施例中,命令流转化器2303指导顶点获取器2305的操作,所述顶点获取器从存储器读取顶点数据并执行由命令流转化器2303提供的顶点处理命令。在一些实施例中,顶点获取器2305将顶点数据提供给顶点着色器2307,所述顶点着色器向每个顶点执行

坐标空间变换和照明操作。在一些实施例中,顶点获取器2305和顶点着色器2307通过经由线程分派器2331将执行线程分派给执行单元2352A、2352B来执行顶点处理指令。

[0343] 在一些实施例中,执行单元2352A、2352B是具有用于执行图形和媒体操作的指令集的向量处理器的阵列。在一些实施例中,执行单元2352A、2352B具有特定用于每个阵列或在阵列之间共享的附加L1高速缓存2351。所述高速缓存可以被配置为数据高速缓存、指令高速缓存或单个高速缓存,所述单个高速缓存被分割成将数据和指令包含在不同的分区中。

[0344] 在一些实施例中,图形流水线2320包括用于执行对3D对象的硬件加速曲面细分的曲面细分部件。在一些实施例中,可编程的外壳着色器2311配置曲面细分操作。可编程的域着色器2317提供对曲面细分输出的后端评估。曲面细分器2313在外壳着色器2311的方向上进行操作并且包含专用逻辑,所述专用逻辑用于基于粗糙几何模型来生成详细的几何对象集合,所述粗糙几何模型作为输入被提供至图形流水线2320。在一些实施例中,如果未使用曲面细分,则可以对曲面细分部件2311、2313、2317进行旁路。

[0345] 在一些实施例中,完整的几何对象可以由几何着色器2319经由被分派给执行单元2352A、2352B的一个或多个线程来处理,或可以直接继续进行至剪辑器2329。在一些实施例中,几何着色器在整个几何对象(而非顶点或者如图形流水线的先前级中的顶点补片)上进行操作。如果曲面细分被禁用,那么几何着色器2319从顶点着色器2307接收输入。在一些实施例中,几何着色器2319可由几何着色器程序编程以便在曲面细分单元被禁用时执行几何曲面细分。

[0346] 在光栅化之前,剪辑器2329处理顶点数据。剪辑器2329可以是固定功能的剪辑器或者具有剪裁和几何着色器功能的可编程剪辑器。在一些实施例中,渲染输出流水线2370中的光栅化器2373(例如,深度测试部件)分派像素着色器以将几何对象转换为它们的逐像素表示。在一些实施例中,像素着色器逻辑包括在线程执行逻辑2350中。在一些实施例中,应用可以对光栅化器2373进行旁路,并且经由流出单元2323来访问未光栅化的顶点数据。

[0347] 图形处理器2300具有互连总线、互连结构或某种其他互连机制,其允许在处理器主要部件当中传递数据和消息。在一些实施例中,执行单元2352A、2352B和(多个)关联高速缓存2351、纹理和媒体采样器2354以及纹理/采样器高速缓存2358经由数据端口2356互连,以执行存储器访问并与处理器的渲染输出流水线部件通信。在一些实施例中,采样器2354、高速缓存2351、2358以及执行单元2352A、2352B各自具有单独的存储器访问路径。

[0348] 在一些实施例中,渲染输出流水线2370包含光栅化器2373,所述光栅化器将基于顶点的对象转换为关联的基于像素的表示。在一些实施例中,光栅化器逻辑包括用于执行固定功能三角形和线光栅化的窗口器/掩码器单元。相关联的渲染高速缓存2378和深度高速缓存2379在一些实施例中也是可用的。像素操作部件2377对数据执行基于像素的操作,不过在一些示例中,与2D操作相关联的像素操作(例如,位块图像传输和混合)由2D引擎2341执行,或在显示时间由使用重叠显示平面的显示控制器2343代替。在一些实施例中,共享的L3高速缓存2375可用于所有的图形部件,从而允许在无需使用主系统存储器的情况下共享数据。

[0349] 在一些实施例中,图形处理器媒体流水线2330包括媒体引擎2337和视频前端2334。在一些实施例中,视频前端2334从命令流转化器2303接收流水线命令。在一些实施例

中,媒体流水线2330包括单独的命令流转化器。在一些实施例中,视频前端2334在将所述命令发送给媒体引擎2337之前处理媒体命令。在一些实施例中,媒体引擎2337包括用于生成线程以用于经由线程分派器2331分派给线程执行逻辑2350的线程生成功能。

[0350] 在一些实施例中,图形处理器2300包括显示引擎2340。在一些实施例中,显示引擎2340在处理器2300外部并且经由环形互连2302、或某个其他互连总线或结构耦合至图形处理器。在一些实施例中,显示引擎2340包括2D引擎2341和显示控制器2343。在一些实施例中,显示引擎2340包含能够独立于3D流水线而操作的专用逻辑。在一些实施例中,显示控制器2343与显示装置(未示出)耦合,所述显示装置可以是系统集成式显示装置(如在膝上型计算机中),或可以是经由显示装置连接器所外接的外部显示装置。

[0351] 在一些实施例中,图形流水线2320和媒体流水线2330可配置成基于多个图形和媒体编程接口来执行操作,并且不特定于任何一个应用编程接口(API)。在一些实施例中,图形处理器的驱动器软件将特定于特定图形或媒体库的API调度转换成可由图形处理器处理的命令。在一些实施例中,为来自科纳斯(Khronos)集团的开放图形库(OpenGL)和开放计算语言(OpenCL)、来自微软公司的Direct 3D库提供支持、或者可以向OpenGL和D3D两者提供支持。还可以为开源计算机视觉库(OpenCV)提供支持。如果可以进行从未来API调用的流水线至图形处理器的流水线的映射,那么还将支持具有兼容的3D流水线的未来API。

[0352] 图形流水线编程

[0353] 图24A是根据一些实施例的示意图形处理器命令格式2400的框图。图24B是根据实施例的示意图形处理器命令序列2410的框图。图24A中的实线框展示通常包括在图形命令中的分量,而虚线框包括可选的或仅包括在图形命令的子集中的分量。图24A的示例性图形处理器命令格式2400包括用于标识命令的目标客户端2402、命令操作代码(操作码)2404和命令的相关数据2406的数据字段。在一些命令中还包括子操作码2405和命令大小2408。

[0354] 在一些实施例中,客户端2402指定处理命令数据的图形装置的客户端单元。在一些实施例中,图形处理器命令解析器检查每个命令的客户端字段以便调整对命令的进一步处理并将命令数据路由至合适的客户端单元。在一些实施例中,图形处理器客户端单元包括存储器接口单元、渲染单元、2D单元、3D单元、和媒体单元。每个客户端单元具有处理命令的对应的处理流水线。一旦命令被客户端单元接收,客户端单元就读取操作码2404以及(如果存在的话)子操作码2405以确定待执行的操作。客户端单元使用数据字段2406中的信息来执行命令。针对一些命令,期待显式命令大小2408以指定命令的大小。在一些实施例中,命令解析器基于命令操作码自动地确定命令中的至少一些命令的大小。在一些实施例中,经由双倍字长的倍数对命令进行对齐。

[0355] 图24B中的流程图示出了示例性图形处理器命令序列2410。在一些实施例中,以图形处理器的实施例为特征的数据处理系统的软件或固件使用所示出的命令序列的版本来启动、执行并终止图形操作集合。仅出于示例性目的示出并描述了样本命令序列,因为实施例并不限于这些特定命令或者此命令序列。此外,所述命令可以作为一批命令以命令序列被发布,从而使得图形处理器将以至少部分同时的方式处理命令序列。

[0356] 在一些实施例中,图形处理器命令序列2410可以流水线转储清除命令2412开始,以使任何活跃的图形流水线完成所述流水线的当前未决命令。在一些实施例中,3D流水线2422和媒体流水线2424不同时进行操作。执行流水线转储清除以使活跃的图形流水线完任

何未决命令。响应于流水线转储清除,图形处理器的命令解析器将暂停命令处理,直到活跃的绘图引擎完成未决操作且相关的读取高速缓存无效。可选地,渲染高速缓存中被标记为‘脏(dirty)’的任何数据可以被转储清除到存储器。在一些实施例中,可以针对流水线同步或在将图形处理器放置处于低功率状态之前使用流水线转储清除命令2412。

[0357] 在一些实施例中,当命令序列要求图形处理器在流水线之间作明确切换时,使用流水线选择命令2413。在一些实施例中,在发布流水线命令之前在执行情境中仅需要一次流水线选择命令2413,除非所述情境要发布针对两条流水线的命令。在一些实施例中,在经由流水线选择命令2413作流水线切换之前立即需要流水线转储清除命令是2412。

[0358] 在一些实施例中,流水线控制命令2414配置用于操作的图形流水线,并用于对3D流水线2422和媒体流水线2424编程。在一些实施例中,流水线控制命令2414配置活跃流水线的流水线状态。在一个实施例中,流水线控制命令2414被用于流水线同步,以及用于在处理一批命令之前将数据从活跃的流水线内的一个或多个高速缓存存储器中清除。

[0359] 在一些实施例中,使用返回缓冲器状态命令2416来配置用于使相应的流水线写入数据的一组返回缓冲器。一些流水线操作需要分配、选择或配置一个或多个返回缓冲器,这些操作在处理期间将中间数据写入到所述返回缓冲器中。在一些实施例中,图形处理器还使用一个或多个返回缓冲器以便存储输出数据并且执行跨线程通信。在一些实施例中,返回缓冲器状态2416包括选择返回缓冲器的大小和数量以用于流水线操作集合。

[0360] 命令序列中的剩余命令基于用于操作的活跃流水线而不同。基于流水线确定2420,根据3D流水线2422和媒体流水线2424来定制命令序列,所述3D流水线以3D流水线状态2430开始,所述媒体流水线始于媒体流水线状态2440处。

[0361] 用于3D流水线状态2430的命令包括用于以下各者的3D状态设置命令:顶点缓冲器状态、顶点元素状态、恒定颜色状态、深度缓冲器状态和将在处理3D图元命令之前配置的其他状态变量。至少部分地基于使用中的特定3D API来确定这些命令的值。在一些实施例中,3D流水线状态2430命令还能够选择性地禁用或旁路掉特定流水线元件(如果将不使用那些元件的话)。

[0362] 在一些实施例中,3D图元2432命令用于提交待由3D流水线处理的3D图元。经由3D图元2432传递到图形处理器的命令和关联的参数被转发到图形流水线中的顶点获取函数。顶点获取函数使用3D图元2432命令数据来产生顶点数据结构。顶点数据结构被存储在一个或多个返回缓冲器中。在一些实施例中,3D图元2432命令用于经由顶点着色器对3D图元执行顶点操作。为了处理顶点着色器,3D流水线2422将着色器执行线程分派给图形处理器执行单元。

[0363] 在一些实施例中,经由执行2434命令或事件来触发3D流水线2422。在一些实施例中,寄存器写入触发命令执行。在一些实施例中,经由命令序列中的‘go’或‘kick’命令来触发执行。在一个实施例中,使用流水线同步命令来触发命令执行,以通过图形流水线来转储清除命令序列。3D流水线将执行针对3D图元的几何处理。一旦操作完成,便对所得几何对象光栅化,并且像素引擎给所得像素上色。针对那些操作还可包括用于控制像素着色和像素后端操作的附加命令。

[0364] 在一些实施例中,当执行媒体操作时,图形处理器命令序列2410遵循媒体流水线2424路径。一般地,媒体流水线2424的特定用途和编程方式取决于待执行的媒体或计算操

作。在媒体解码期间,可将特定的媒体解码操作卸载到媒体流水线。在一些实施例中,还可以对媒体流水线进行旁路,并且可以使用由一个或多个通用处理核提供的资源来整体地或部分地执行媒体解码。在一个实施例中,媒体流水线还包括用于通用图形处理器单元(GPGPU)操作的元件,其中,图形处理器用于使用计算着色器程序来执行SIMD向量操作,所述计算着色器程序与图形图元的渲染不明确相关。

[0365] 在一些实施例中,以与3D流水线2422类似的方式配置媒体流水线2424。在媒体对象命令2442之前将一组媒体流水线状态命令2440分派到或放置到命令队列中。在一些实施例中,媒体流水线状态命令2440包括用于配置媒体流水线元件的数据,所述媒体流水线元件将用于处理媒体对象。这包括用于配置媒体流水线内的视频解码和视频编码逻辑的数据(比如编码或解码模式)。在一些实施例中,媒体流水线状态命令2440还支持将一个或多个指针用于包含一批状态设置的“间接”状态元件。

[0366] 在一些实施例中,媒体对象命令2442将指针供应给供由媒体流水线处理的媒体对象。媒体对象包括包含待处理的视频数据的存储器缓冲器。在一些实施例中,在发布媒体对象命令2442之前,所有的媒体流水线状态必须是有效的。一旦流水线状态被配置并且媒体对象命令2442被排队,则经由执行命令2444或等效的执行事件(例如,寄存器写入)来触发媒体流水线2424。然后可以通过由3D流水线2422或媒体流水线2424提供的操作对来自媒体流水线2424的输出进行后处理。在一些实施例中,以与媒体操作类似的方式来配置和执行GPGPU操作。

[0367] 图形软件架构

[0368] 图25展示根据一些实施例的数据处理系统2500的示例性图形软件架构。在一些实施例中,软件架构包括3D图形应用2510、操作系统2520、以及至少一个处理器2530。在一些实施例中,处理器2530包括图形处理器2532以及一个或多个通用处理器核2534。图形应用2510和操作系统2520各自在数据处理系统的系统存储器2550中执行。

[0369] 在一些实施例中,3D图形应用2510包含一个或多个着色器程序,所述着色器程序包括着色器指令2512。着色器语言指令可以呈高阶着色器语言,比如高阶着色器语言(HLSL)或OpenGL着色器语言(GLSL)。所述应用还包括呈适合于由通用处理器核2534执行的机器语言的可执行指令2514。所述应用还包括由顶点数据定义的几何对象2516。

[0370] 在一些实施例中,操作系统2520是来自微软公司的Microsoft® Windows®操作系统、使用Linux内核的变体的专属类UNIX操作系统或开源类UNIX操作系统。当Direct3D API在使用中时,操作系统2520使用前端着色器编译器2524以将呈HLSL的任何着色器指令2512编译为低阶着色器语言。所述编译可以是即时(JIT)编译,或者所述应用可执行着色器预编译。在一些实施例中,在对3D图形应用2510进行编译期间,将高阶着色器编译成低阶着色器。

[0371] 在一些实施例中,用户模式图形驱动器2526包含后端着色器编译器2527,所述后端着色器编译器用于将着色器指令2512转换为硬件特定表示。当OpenGL API在使用中时,呈GLSL高阶语言的着色器指令2512被传递到用户模式图形驱动器2526以供编译。在一些实施例中,用户模式图形驱动器2526使用操作系统内核模式函数2528来与内核模式图形驱动器2529进行通信。在一些实施例中,内核模式图形驱动器2529与图形处理器2532进行通信以便分派命令和指令。

[0372] IP核实现方式

[0373] 至少一个实施例的一个或多个方面可由存储在机器可读介质上的代表性代码来实现,所述机器可读介质表示和/或定义集成电路(比如,处理器)内的逻辑。例如,机器可读介质可以包括表示处理器内的各个逻辑的指令。当由机器读取时,所述指令可以使所述机器制造用于执行本文描述的技术的逻辑。这类表示(称为“IP核”)是集成电路的逻辑的可重复使用单元,所述可重复使用单元可以作为对集成电路的结构进行描述的硬件模型而存储在有形、机器可读介质上。可以将硬件模型供应至在制造集成电路的制造机器上加载硬件模型的各消费者或制造设施。可以制造集成电路,从而使得所述电路执行与在此描述的实施例中的任一实施例相关联地描述的操作。

[0374] 图26是根据实施例的展示IP核开发系统2600的框图,所述IP核开发系统可用于制造集成电路以执行操作。IP核开发系统2600可以用于生成可并入到更大的设计中或用于构建整个集成电路(例如,SOC集成电路)的模块化、可重复使用设计。设计设施2630可采用高阶编程语言(例如,C/C++)生成对IP核设计的软件仿真2610。软件仿真2610可用于设计、测试并验证IP核的行为。然后可由仿真模型2600来创建或合成寄存器传输级(RTL)设计。RTL设计2615是对硬件寄存器之间的数字信号的流动进行建模的集成电路(包括使用建模的数字信号执行的相关联逻辑)的行为的抽象。除了RTL设计2615之外,还可以创建、设计或合成逻辑电平或晶体管电平处的较低层次设计。由此,初始设计和仿真的具体细节可以发生变化。

[0375] 可由设计设施进一步将RTL设计2615或等效物合成为硬件模型2620,所述硬件模型可以呈硬件描述语言(HDL)或物理设计数据的某个其他表示。可以进一步仿真或测试HDL以验证IP核设计。可以使用非易失性存储器2640(例如,硬盘、闪存或任何非易失性存储介质)来存储IP核设计以供递送到第3方制造设施2665。可替代地,可以通过有线连接2650或无线连接2660来传输(例如,经由互联网)IP核设计。制造设施2665然后可以制造至少部分地基于IP核设计的集成电路。所制造的集成电路可被配置用于执行根据本文描述的至少一个实施例的操作。

[0376] 图27是根据实施例的展示示例性片上系统集成电路2700的框图,可使用一个或多个IP核来制造所述系统芯片集成电路。示例性集成电路包括一个或多个应用处理器2705(例如,CPU)、至少一个图形处理器2710,并且可以另外包括图像处理2715和/或视频处理器2720,其中的任一中可以是来自相同或多个不同设计设施的模块化IP核。集成电路包括外围或总线逻辑,包括USB控制器2725、UART控制器2730、SPI/SDIO控制器2735、I²S/I²C控制器2740。另外,集成电路可以包括显示装置2745,所述显示装置耦合至高清晰度多媒体接口(HDMI)控制器2750和移动行业处理器接口(MIPI)显示接口2755中的一者或多者。可以由闪存子系统2760(包括闪存和闪存控制器)来提供存储。可经由存储器控制器2765来提供存储器接口以用于访问SDRAM或SRAM存储器装置。一些集成电路另外包括嵌入式安全引擎2770。

[0377] 另外,其他逻辑和电路可被包括在集成电路2700的处理器中,这些逻辑和电路包括附加的图形处理器/核、外围接口控制器或通用处理器核。

[0378] 有利地,上述系统、处理器、图形处理器、设备、和/或方法中的任何一项可以集成有或配置有本文中所描述的各个实施例中的任何一个(例如,或其部分),包括例如在以下附加注释和示例中所描述的这些。

[0379] 附加的注释和示例

[0380] 示例1可以包括一种图形设备,包括:处理器;存储器,通信地耦合至所述处理器;以及协作引擎,通信地耦合至所述处理器以用于识别环境中的两个或更多个用户之间的共享图形分量;以及与所述环境中的所述两个或更多个用户共享所述共享图形分量。

[0381] 示例2可以包括如示例1所述的设备,其中,所述协作引擎进一步包括集中式共享器,所述集中式共享器用于:广播器,用于将所述共享图形分量广播至所有所述两个或更多个用户;以及分配器,用于将单独的图形分量单独地分配至所述两个或更多个用户中的一个用户。

[0382] 示例3可以包括如示例2所述的设备,其中,所述集中式共享器进一步用于:基于目标虚拟现实设备来分割所述共享图形分量的工作负荷。

[0383] 示例4可以包括如示例1所述的设备,其中,所述协作引擎进一步包括深度共享器,所述深度共享器用于:从独立于所述两个或更多个用户中的第一用户的源收集深度信息;以及与所述第一用户共享所述深度信息。

[0384] 示例5可以包括如示例4所述的设备,其中,所述深度共享器进一步用于:从所述第一用户的视场之外的所述独立源收集可见性信息;以及与所述第一用户共享所述可见性信息。

[0385] 示例6可以包括如示例1所述的设备,其中,所述协作引擎进一步包括共享预处理器,所述共享预处理器用于:预计算与所述共享图形分量有关的信息;以及与所述两个或更多个用户共享预计算的信息。

[0386] 示例7可以包括如示例6所述的设备,其中,所述预计算的信息包括几何信息、图元信息、和物理信息中的一项或多项。

[0387] 示例8可以包括如示例1所述的设备,其中,所述协作引擎进一步包括多端口图形子系统,所述多端口图形子系统用于:在所述多端口图形子系统的每个端口上支持不同的用户。

[0388] 示例9可以包括如示例8所述的设备,其中,所述多端口图形子系统进一步用于:当仅第一虚拟现实设备连接至所述多端口图形子系统时,支持所述第一虚拟现实设备的左眼显示器和右眼显示器;当所述第一虚拟现实设备和第二虚拟现实设备两者均连接至所述多端口图形子系统时,支持所述第一虚拟现实设备和所述第二虚拟现实设备;以及当所述第一虚拟现实设备和所述第二虚拟现实设备两者均连接至所述多端口图形子系统时,在所述第一虚拟现实设备与所述第二虚拟现实设备之间共享图形图元。

[0389] 示例10可以包括如示例1所述的设备,其中,所述协作引擎进一步包括解码共享器,所述解码共享器用于:识别第一用户的第一视图区与第二用户的第二视图区之间的共享场景的重叠区;对所述重叠区进行解码;以及与所述第一用户和所述第二用户两者共享经解码的重叠区。

[0390] 示例11可以包括如示例10所述的设备,其中,所述解码共享器进一步用于:基于所述重叠区对所述解码进行优先级排序。

[0391] 示例12可以包括一种图形协作的方法,所述方法包括:识别环境中的两个或更多个用户之间的共享图形分量;以及与所述环境中的所述两个或更多个用户共享所述共享图形分量。

[0392] 示例13可以包括如示例12所述的方法,进一步包括:将所述共享图形分量广播至所有所述两个或更多个用户;以及将单独的图形分量单独地分配至所述两个或更多个用户中的一个用户。

[0393] 示例14可以包括如示例13所述的方法,进一步包括:基于目标虚拟现实设备来分割所述共享图形分量的工作负荷。

[0394] 示例15可以包括如示例12所述的方法,进一步包括:从独立于所述两个或更多个用户中的第一用户的源收集深度信息;以及与所述第一用户共享所述深度信息。

[0395] 示例16可以包括如示例15所述的方法,进一步包括:从所述第一用户的视场之外的所述独立源收集可见性信息;以及与所述第一用户共享所述可见性信息。

[0396] 示例17可以包括如示例12所述的方法,进一步包括:预计算与所述共享图形分量有关的信息;以及与所述两个或更多个用户共享预计算的信息。

[0397] 示例18可以包括如示例17所述的方法,其中,所述预计算的信息包括几何信息、图元信息、和物理信息中的一项或多项。

[0398] 示例19可以包括如示例12所述的方法,进一步包括:在多端口图形子系统的每个端口上支持不同的用户。

[0399] 示例20可以包括如示例19所述的方法,进一步包括:当仅第一虚拟现实设备连接至所述多端口图形子系统时,支持所述第一虚拟现实设备的左眼显示器和右眼显示器;当所述第一虚拟现实设备和第二虚拟现实设备两者均连接至所述多端口图形子系统时,支持所述第一虚拟现实设备和所述第二虚拟现实设备;以及当所述第一虚拟现实设备和所述第二虚拟现实设备两者均连接至所述多端口图形子系统时,在所述第一虚拟现实设备与所述第二虚拟现实设备之间共享图形图元。

[0400] 示例21可以包括如示例12所述的方法,进一步包括:识别第一用户的第一视图区与第二用户的第二视图区之间的共享场景的重叠区;对所述重叠区进行解码;以及与所述第一用户和所述第二用户两者共享经解码的重叠区。

[0401] 示例22可以包括如示例21所述的方法,进一步包括:基于所述重叠区对所述解码进行优先级排序。

[0402] 示例23可以包括至少一种计算机可读介质,包括一组指令,所述指令当由计算装置执行时使所述计算装置:识别环境中的两个或更多个用户之间的共享图形分量;以及与所述环境中的所述两个或更多个用户共享所述共享图形分量。

[0403] 示例24可以包括如示例23所述的至少一种计算机可读介质,包括另一组指令,所述指令当由计算装置执行时使所述计算装置:将所述共享图形分量广播至所有所述两个或更多个用户;以及将单独的图形分量单独地分配至所述两个或更多个用户中的一个用户。

[0404] 示例25可以包括如示例24所述的至少一种计算机可读介质,包括另一组指令,所述指令当由计算装置执行时使所述计算装置:基于目标虚拟现实设备来分割所述共享图形分量的工作负荷。

[0405] 示例26可以包括如示例23所述的至少一种计算机可读介质,包括另一组指令,所述指令当由计算装置执行时使所述计算装置:从独立于所述两个或更多个用户中的第一用户的源收集深度信息;以及与所述第一用户共享所述深度信息。

[0406] 示例27可以包括如示例26所述的至少一种计算机可读介质,包括另一组指令,所

述指令当由计算装置执行时使所述计算装置:从所述第一用户的视场之外的所述独立源收集可见性信息;以及与所述第一用户共享所述可见性信息。

[0407] 示例28可以包括如示例23所述的至少一种计算机可读介质,包括另一组指令,所述指令当由计算装置执行时使所述计算装置:预计算与所述共享图形分量有关的信息;以及与所述两个或更多个用户共享预计算的信息。

[0408] 示例29可以包括如示例28所述的至少一种计算机可读介质,其中,所述预计算的信息包括几何信息、图元信息、和物理信息中的一项或多项。

[0409] 示例30可以包括如示例23所述的至少一种计算机可读介质,包括另一组指令,所述指令当由计算装置执行时使所述计算装置:在多端口图形子系统的每个端口上支持不同的用户。

[0410] 示例31可以包括如示例30所述的至少一种计算机可读介质,包括另一组指令,所述指令当由计算装置执行时使所述计算装置:当仅第一虚拟现实设备连接至所述多端口图形子系统时,支持所述第一虚拟现实设备的左眼显示器和右眼显示器;当所述第一虚拟现实设备和第二虚拟现实设备两者均连接至所述多端口图形子系统时,支持所述第一虚拟现实设备和所述第二虚拟现实设备;以及当所述第一虚拟现实设备和所述第二虚拟现实设备两者均连接至所述多端口图形子系统时,在所述第一虚拟现实设备与所述第二虚拟现实设备之间共享图形图元。

[0411] 示例32可以包括如示例23所述的至少一种计算机可读介质,包括另一组指令,所述指令当由计算装置执行时使所述计算装置:识别第一用户的第一视图区与第二用户的第二视图区之间的共享场景的重叠区;对所述重叠区进行解码;以及与所述第一用户和所述第二用户两者共享经解码的重叠区。

[0412] 示例33可以包括如示例32所述的至少一种计算机可读介质,包括另一组指令,所述指令当由计算装置执行时使所述计算装置:基于所述重叠区对所述解码进行优先级排序。

[0413] 示例34可以包括一种图形设备,包括:用于识别环境中的两个或更多个用户之间的共享图形分量的装置;以及用于与所述环境中的所述两个或更多个用户共享所述共享图形分量的装置。

[0414] 示例35可以包括如示例34所述的设备,进一步包括:用于将所述共享图形分量广播至所有所述两个或更多个用户的装置;以及用于将单独的图形分量单独地分配至所述两个或更多个用户中的一个用户的装置。

[0415] 示例36可以包括如示例35所述的设备,进一步包括:用于基于目标虚拟现实设备来分割所述共享图形分量的工作负荷的装置。

[0416] 示例37可以包括如示例34所述的设备,进一步包括:用于从独立于所述两个或更多个用户中的第一用户的源收集深度信息的装置;以及用于与所述第一用户共享所述深度信息的装置。

[0417] 示例38可以包括如示例37所述的设备,进一步包括:用于从所述第一用户的视场之外的所述独立源收集可见性信息的装置;以及用于与所述第一用户共享所述可见性信息的装置。

[0418] 示例39可以包括如示例34所述的设备,进一步包括:用于预计算与所述共享图形

分量有关的信息的装置;以及用于与所述两个或更多个用户共享预计算的信息的装置。

[0419] 示例40可以包括如示例39所述的设备,其中,所述预计算的信息包括几何信息、图元信息、和物理信息中的一项或多项。

[0420] 示例41可以包括如示例34所述的设备,进一步包括:用于在多端口图形子系统的每个端口上支持不同的用户的装置。

[0421] 示例42可以包括如示例41所述的设备,进一步包括:用于当仅第一虚拟现实设备连接至所述多端口图形子系统时支持所述第一虚拟现实设备的左眼显示器和右眼显示器的装置;用于当所述第一虚拟现实设备和第二虚拟现实设备两者均连接至所述多端口图形子系统时支持所述第一虚拟现实设备和所述第二虚拟现实设备的装置;以及用于当所述第一虚拟现实设备和所述第二虚拟现实设备两者均连接至所述多端口图形子系统时在所述第一虚拟现实设备与所述第二虚拟现实设备之间共享图形图元的装置。

[0422] 示例43可以包括如示例34所述的设备,进一步包括:用于识别第一用户的第一视图区与第二用户的第二视图区之间的共享场景的重叠区的装置;用于对所述重叠区进行解码的装置;以及用于与所述第一用户和所述第二用户两者共享经解码的重叠区的装置。

[0423] 示例44可以包括如示例43所述的设备,进一步包括:用于基于所述重叠区对所述解码进行优先级排序的装置。

[0424] 示例45可以包括一种图像设备,包括:处理器;存储器,通信地耦合至所述处理器;协作引擎,通信地耦合至所述处理器以接收共享图形分量;以及合成器,通信地耦合至所述处理器以合并共享图形分量与单独的图形分量。

[0425] 示例46可以包括如示例45所述的设备,进一步包括:可穿戴外壳,待由用户穿戴;以及显示器,通信地耦合至所述处理器,其中,所述显示器、所述处理器、所述存储器、协作接口和所述合成器全部由所述可穿戴外壳支持。

[0426] 示例47可以包括如示例45所述的设备,其中,所述协作接口进一步用于:接收包括所述共享图形分量的广播信息。

[0427] 示例48可以包括如示例45所述的设备,其中,所述共享图形分量包括共享深度信息、共享几何信息、和共享物理信息中的一项或多项。

[0428] 示例49可以包括如示例45所述的设备,进一步包括:多端口图形子系统,用于每个端口上支持不同的用户。

[0429] 示例50可以包括如示例45所述的设备,其中,所述共享图形分量包括360视频内容。

[0430] 示例51可以包括一种图形协作的方法,所述方法包括:接收共享图形分量;以及合并所述共享图形分量与单独的图形分量。

[0431] 示例52可以包括如示例51所述的方法,进一步包括:提供待由用户穿戴的可穿戴外壳;以及使用所述可穿戴外壳来支持显示器、处理器、存储器、协作接口以及合成器。

[0432] 示例53可以包括如示例51所述的方法,进一步包括:接收包括所述共享图形分量的广播信息。

[0433] 示例54可以包括如示例51所述的方法,其中,所述共享图形分量包括共享深度信息、共享几何信息、和共享物理信息中的一项或多项。

[0434] 示例55可以包括如示例51所述的方法,进一步包括:在多端口图形子系统的每个

端口上支持不同的用户。

[0435] 示例56可以包括如示例51所述的方法,其中,所述共享图形分量包括360视频内容。

[0436] 示例57可以包括至少一种计算机可读介质,包括一组指令,所述指令当由计算装置执行时使所述计算装置:接收共享图形分量;以及合并共享图形分量与单独的图形分量。

[0437] 示例58可以包括如示例57所述的至少一种计算机可读介质,包括另一组指令,所述指令当由计算装置执行时使所述计算装置:接收包括所述共享图形分量的广播信息。

[0438] 示例59可以包括如示例57所述的至少一种计算机可读介质,其中,所述共享图形分量包括共享深度信息、共享几何信息、和共享物理信息中的一项或多项。

[0439] 示例60可以包括如示例57所述的至少一种计算机可读介质,包括另一组指令,所述指令当由计算装置执行时使所述计算装置:在多端口图形子系统的每个端口上支持不同的用户。

[0440] 示例61可以包括如示例57所述的至少一种计算机可读介质,其中,所述共享图形分量包括360视频内容。

[0441] 示例62可以包括一种图形设备,包括:用于接收共享图形分量的装置;以及用于合并所述共享图形分量与单独的图形分量的装置。

[0442] 示例63可以包括如示例62所述的设备,进一步包括:用于提供待由用户穿戴的可穿戴外壳的装置;以及用于使用所述可穿戴外壳来支持显示器、处理器、存储器、协作接口以及合成器的装置。

[0443] 示例64可以包括如示例62所述的设备,进一步包括:用于接收包括所述共享图形分量的广播信息的装置。

[0444] 示例65可以包括如示例62所述的设备,其中,所述共享图形分量包括共享深度信息、共享几何信息、和共享物理信息中的一项或多项。

[0445] 示例66可以包括如示例62所述的设备,进一步包括:用于在多端口图形子系统的每个端口上支持不同的用户的装置。

[0446] 示例67可以包括如示例62所述的设备,其中,所述共享图形分量包括360视频内容。

[0447] 实施例适用于所有类型的半导体集成电路(“IC”)芯片。这些IC芯片的示例包括但不限于:处理器、控制器、芯片组部件、可编程逻辑阵列(PLA)、存储器芯片、网络芯片、片上系统(SoC)、SSD/NAND控制器ASIC等。另外,在一些附图中,利用线条表示信号导体线。一些线条可以是不同的以指示更多组成的信号路径,具有数字标记以指示组成的信号路径的编号和/或在一端或多端具有箭头以指示主要信息流方向。然而,这不应被解释为限制性方式。而是,这种附加的细节可以与一个或多个示例性实施例结合使用以帮助更容易地理解电路。任何表示的信号线,无论是否有附加信息,都实际可以包括可以在多个方向上行进的一个或多个信号,并且可以利用任何适当类型的信号方案来实现,例如利用差分对、光纤线路和/或单端线路实现的数字或模拟线路。

[0448] 可能已经给出了示例大小/模型/值/范围,尽管实施例不限于此。随着制造技术(例如,光刻)随时间推移而成熟,预计可以制造出更小尺寸的器件。另外,为了简化图示和讨论以及为了不使实施例的一些方面不清晰,可以在图内示出或不示出到IC芯片和其他部

件的公知的电力/接地连接。此外,安排可以以框图的形式示出,以避免模糊实施例,并且还鉴于以下事实:关于这样的框图安排的实施方式的细节高度依赖于在其中实现实施例的平台,即,这样的细节应当完全处在本领域技术人员的视界中。在阐述具体细节(例如电路)以描述示例实施例的情况下,对于本领域技术人员来说应当显而易见的是,可以在没有这些具体细节或具有这些具体细节的变化的情况下实践实施例。描述因此被视为是说明性的而非限制性的。

[0449] 术语“耦合”此处可以用来指所讨论的部件之间任何类型的直接或间接的关系,并可以应用于电子、机械、流体、光学、电磁、电机或其他连接。此外,术语“第一”、“第二”等此处可以仅用来促进讨论,并不带有任何具体的暂时或时间顺序的显著性意义,除非另外指明。而且,应理解不定冠词“一”或“一个”带有“一个或多个”或“至少一个”的意思。

[0450] 如在本申请和权利要求书中使用的,由术语“一个或多个”描述的项目列表可意指所列表项目的任何组合。例如,短语“A、B和C中的一个或多个”可意指A、B、C;A和B;A和C;B和C;或A、B和C。

[0451] 上面已经参考具体实施例描述了各实施例。然而,本领域内技术人员会理解,可以对其进行各种修改和改变而不偏离如在所附的权利要求书中所阐述的实施例的较宽的精神和范围。因此认为前述说描述和附图是说明性的而不是限制性的。

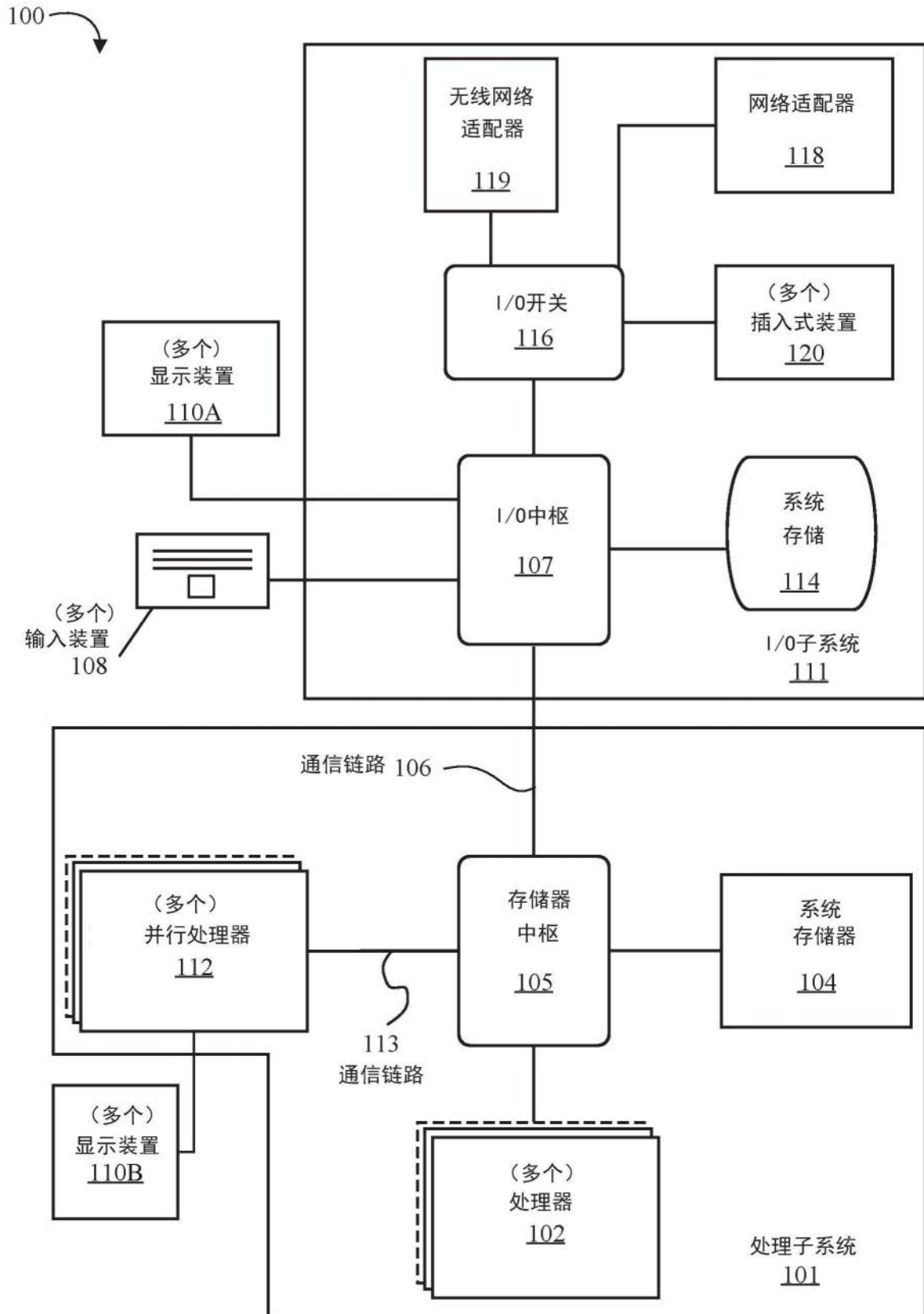


图1

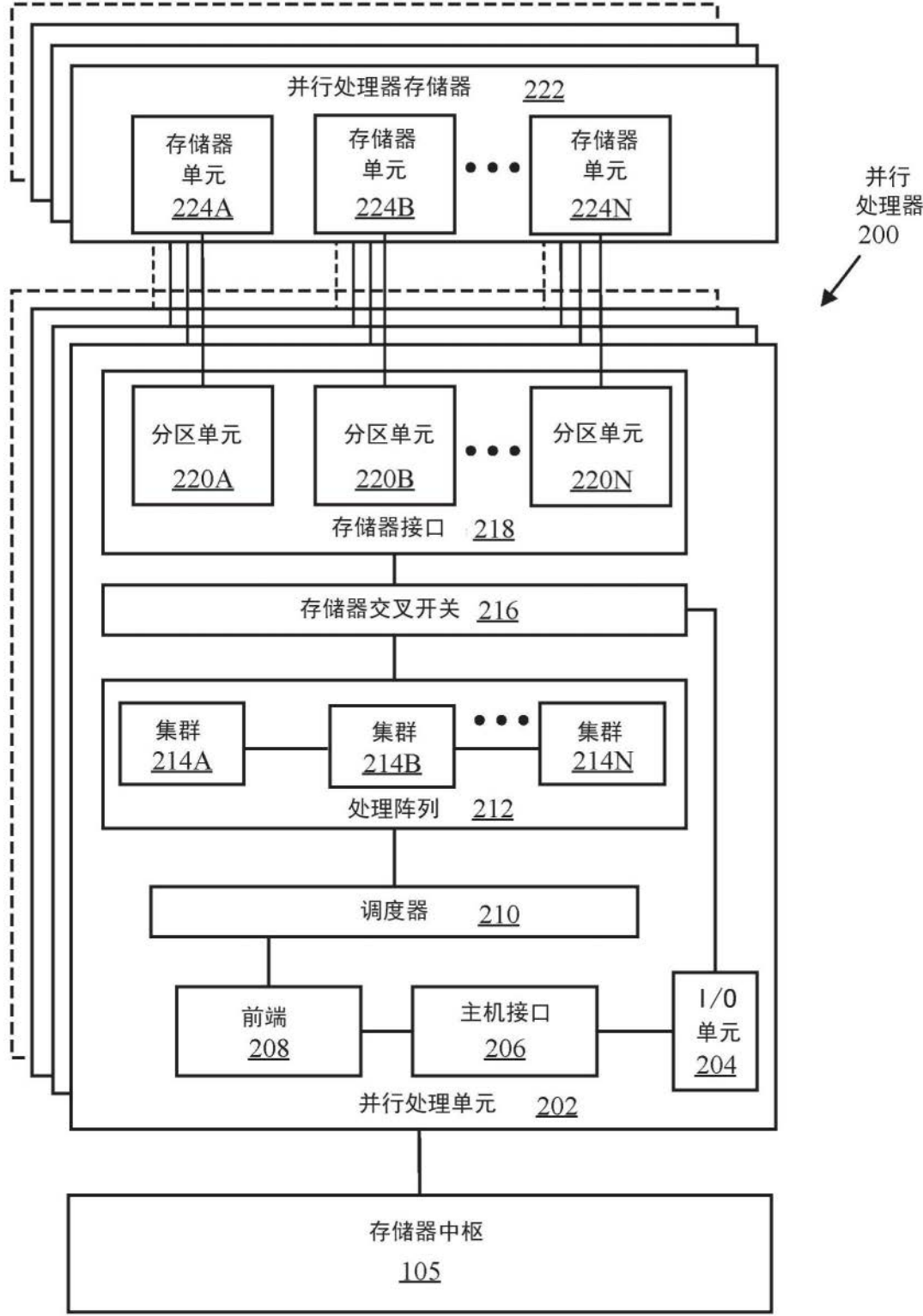


图2A

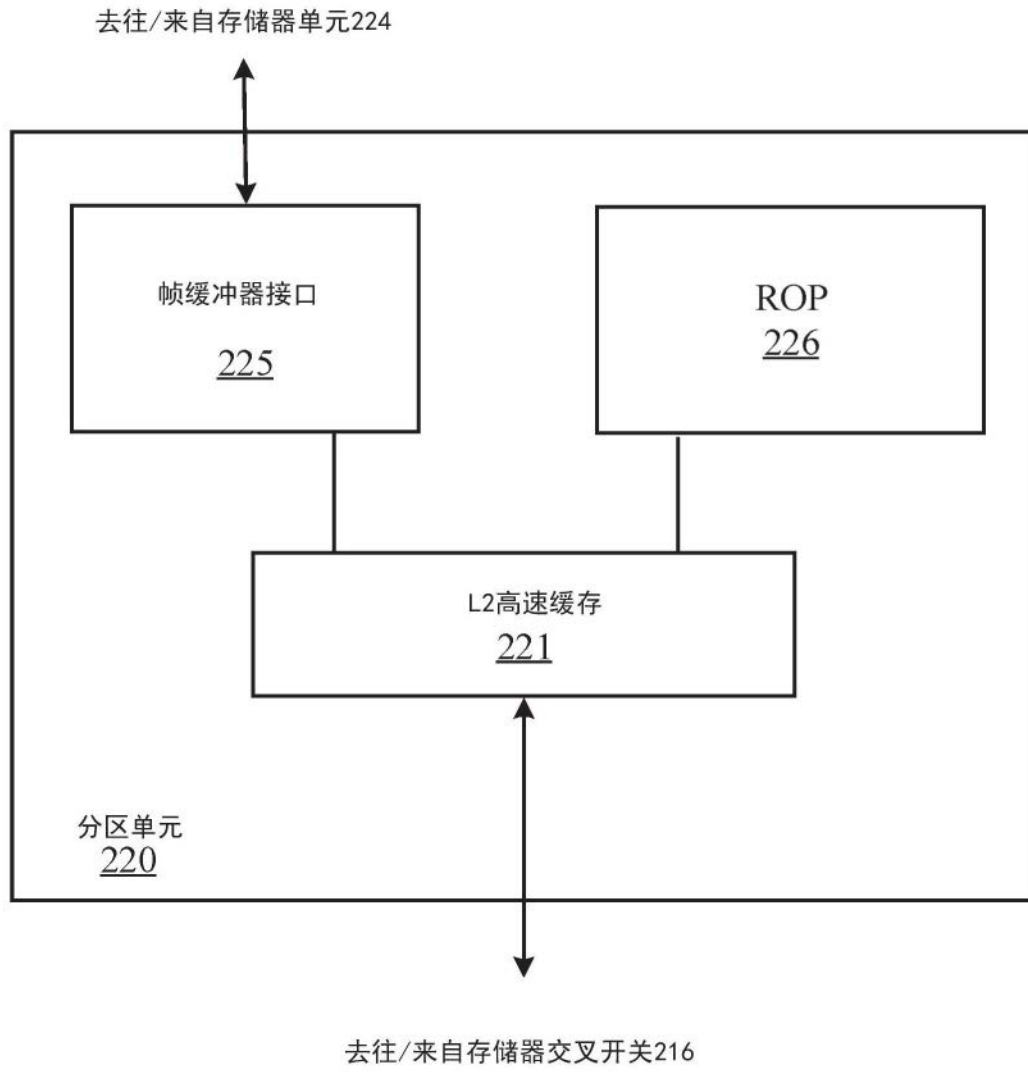


图2B

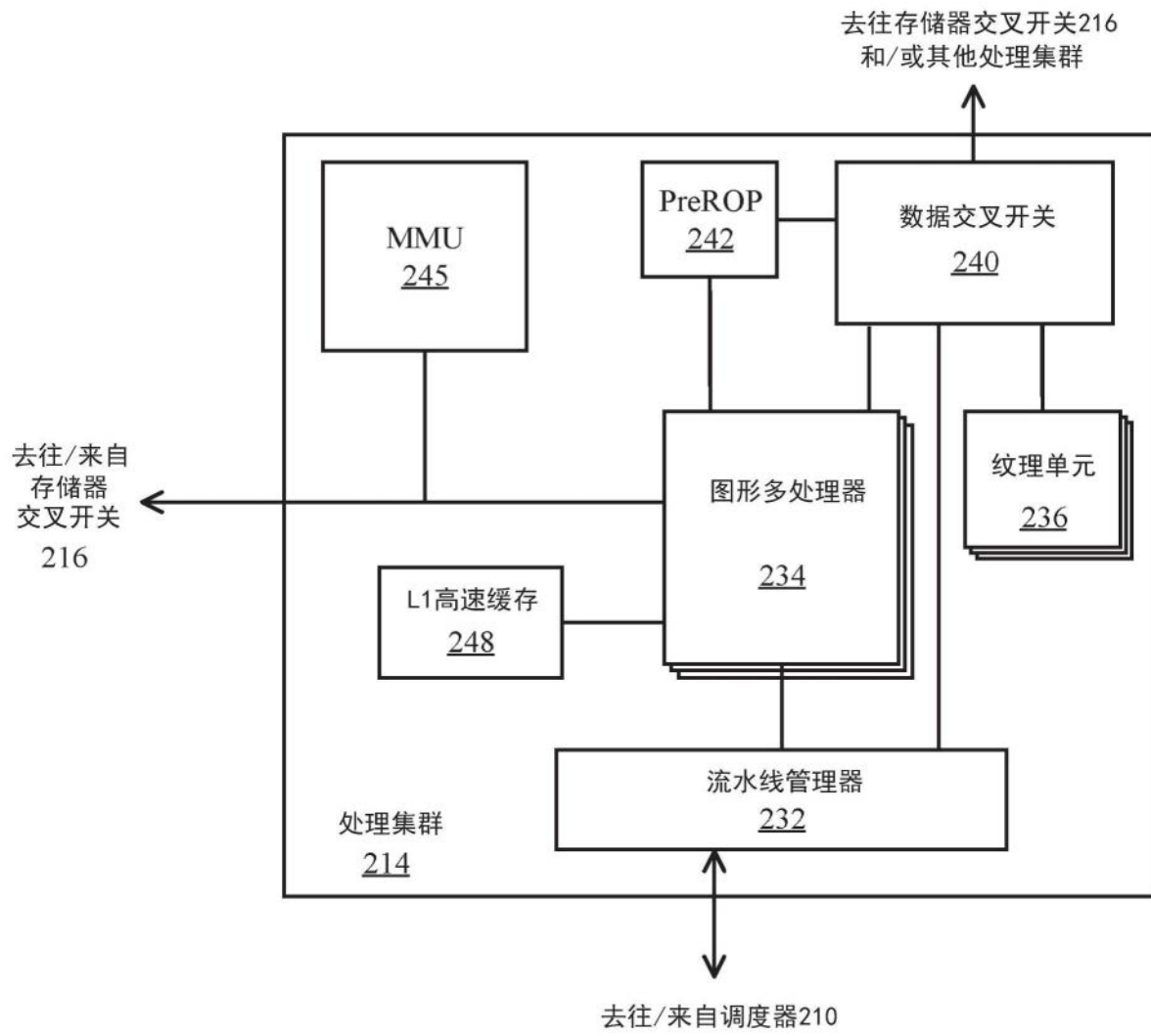


图2C

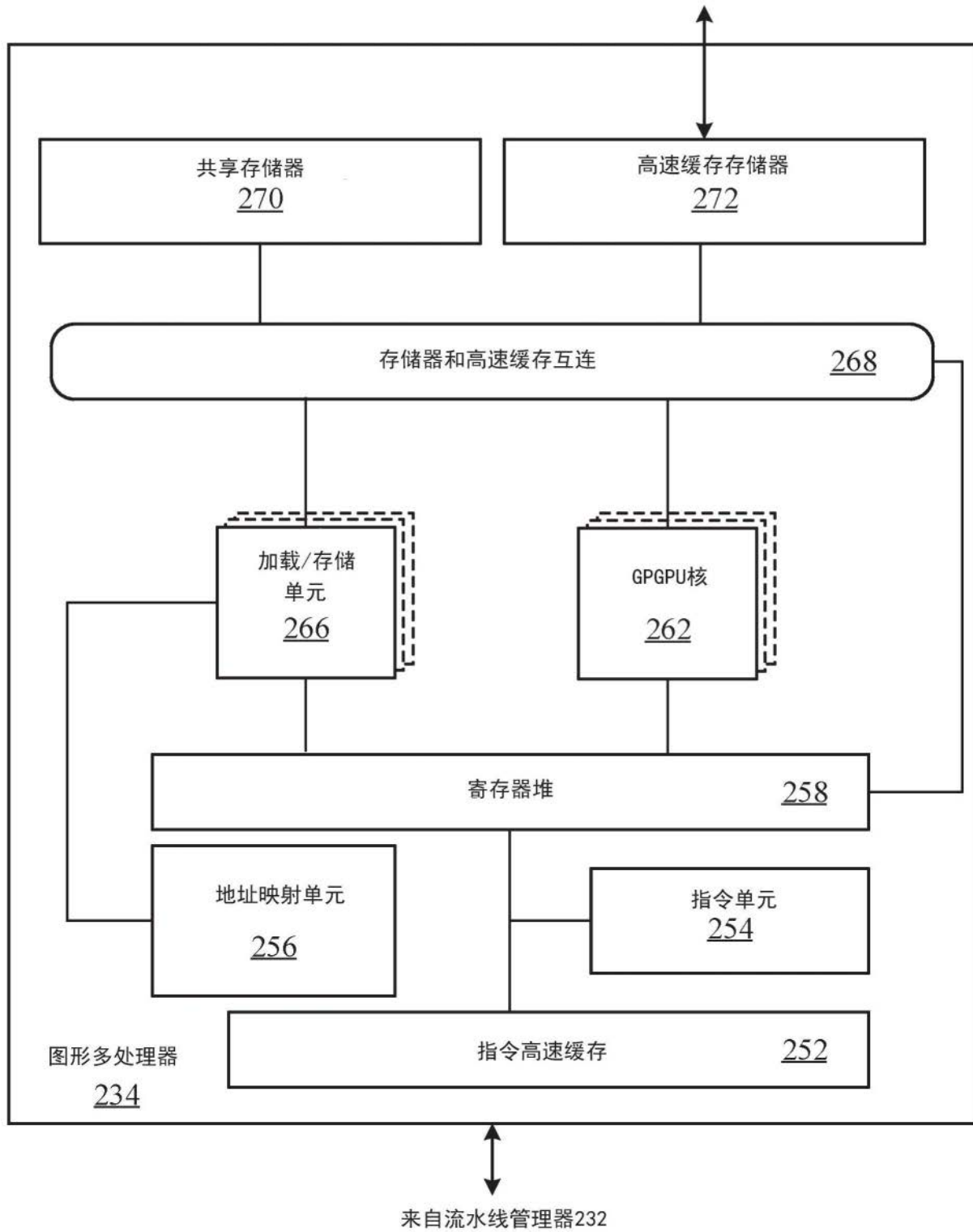


图2D

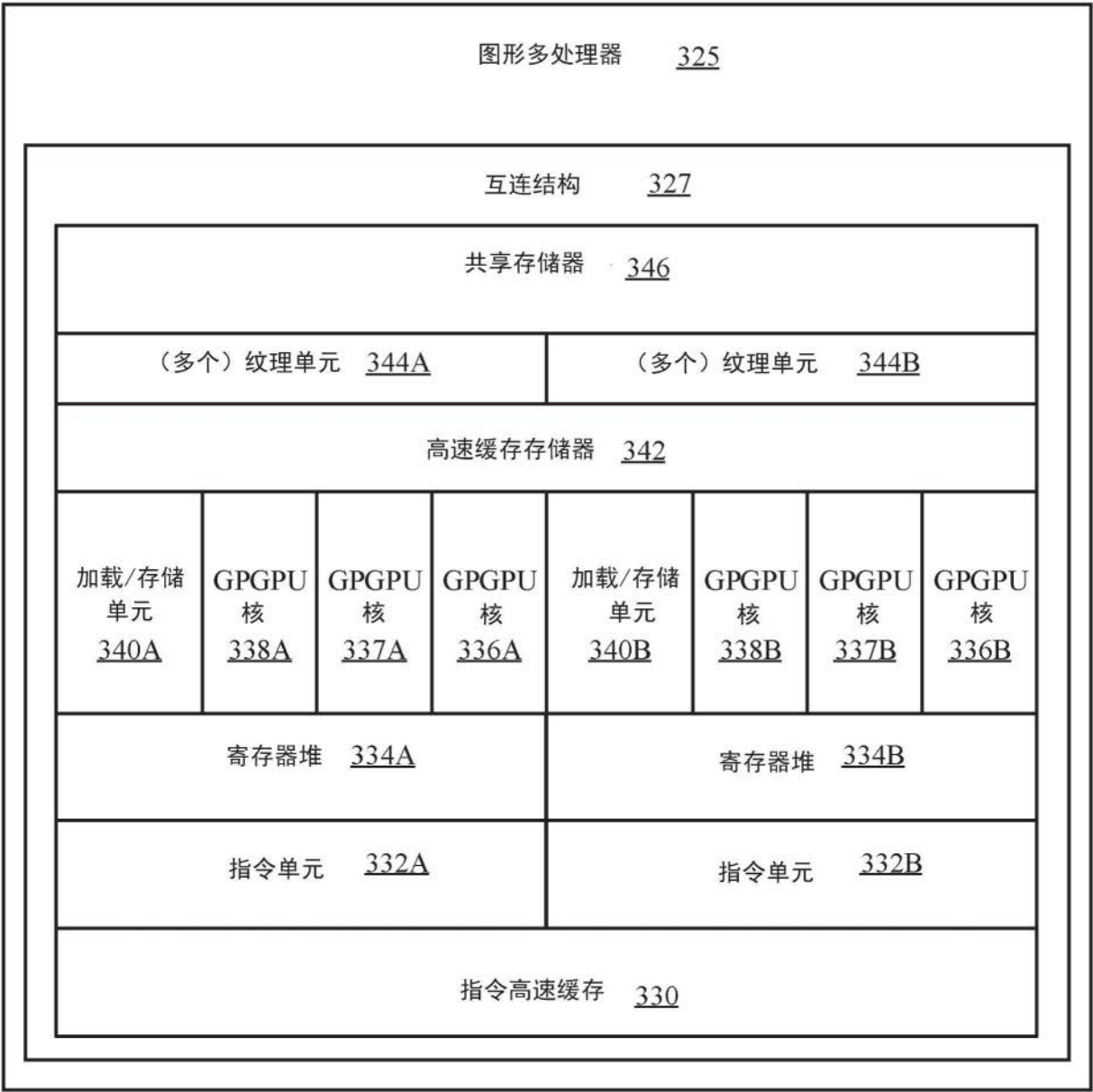


图3A

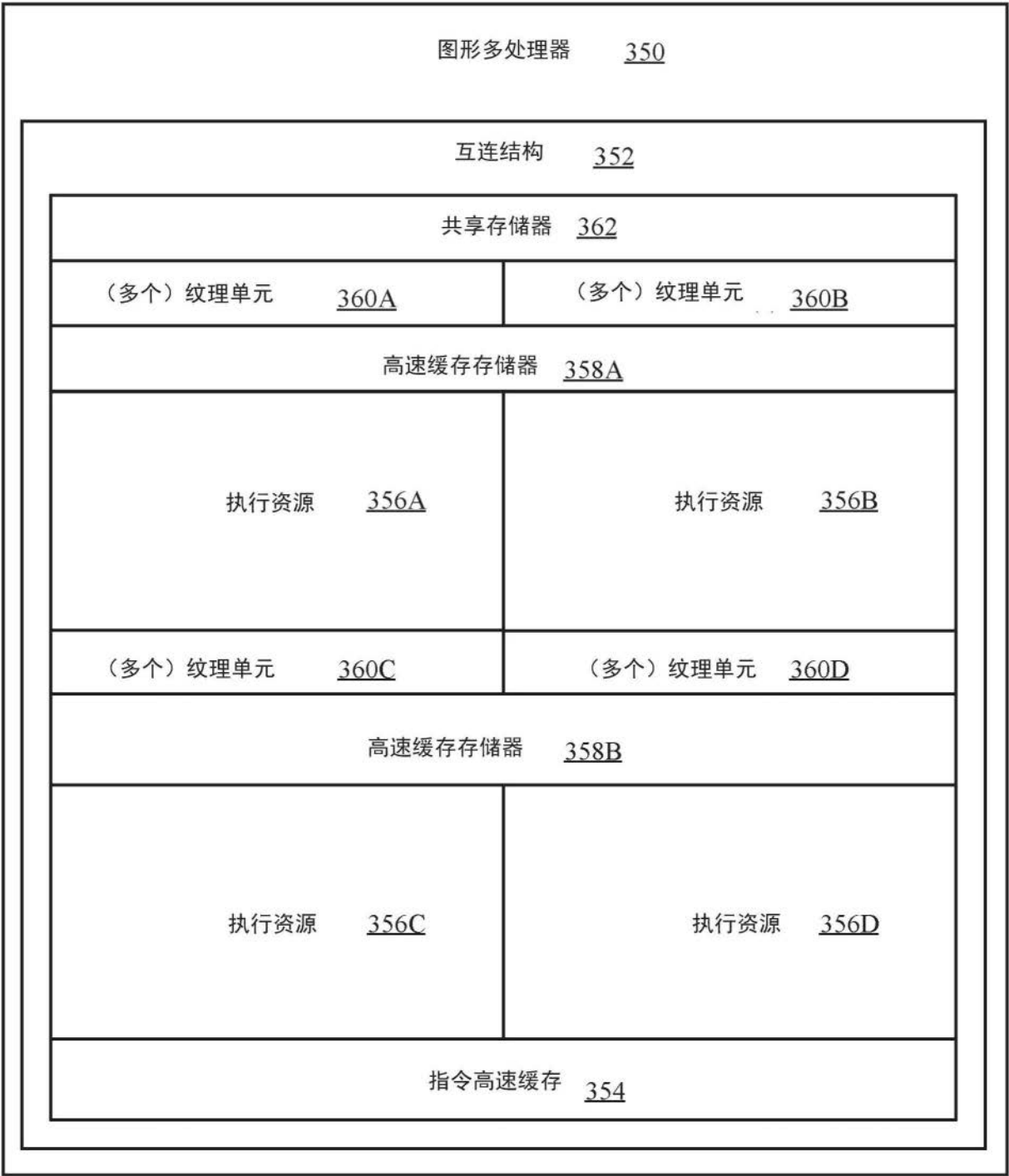


图3B

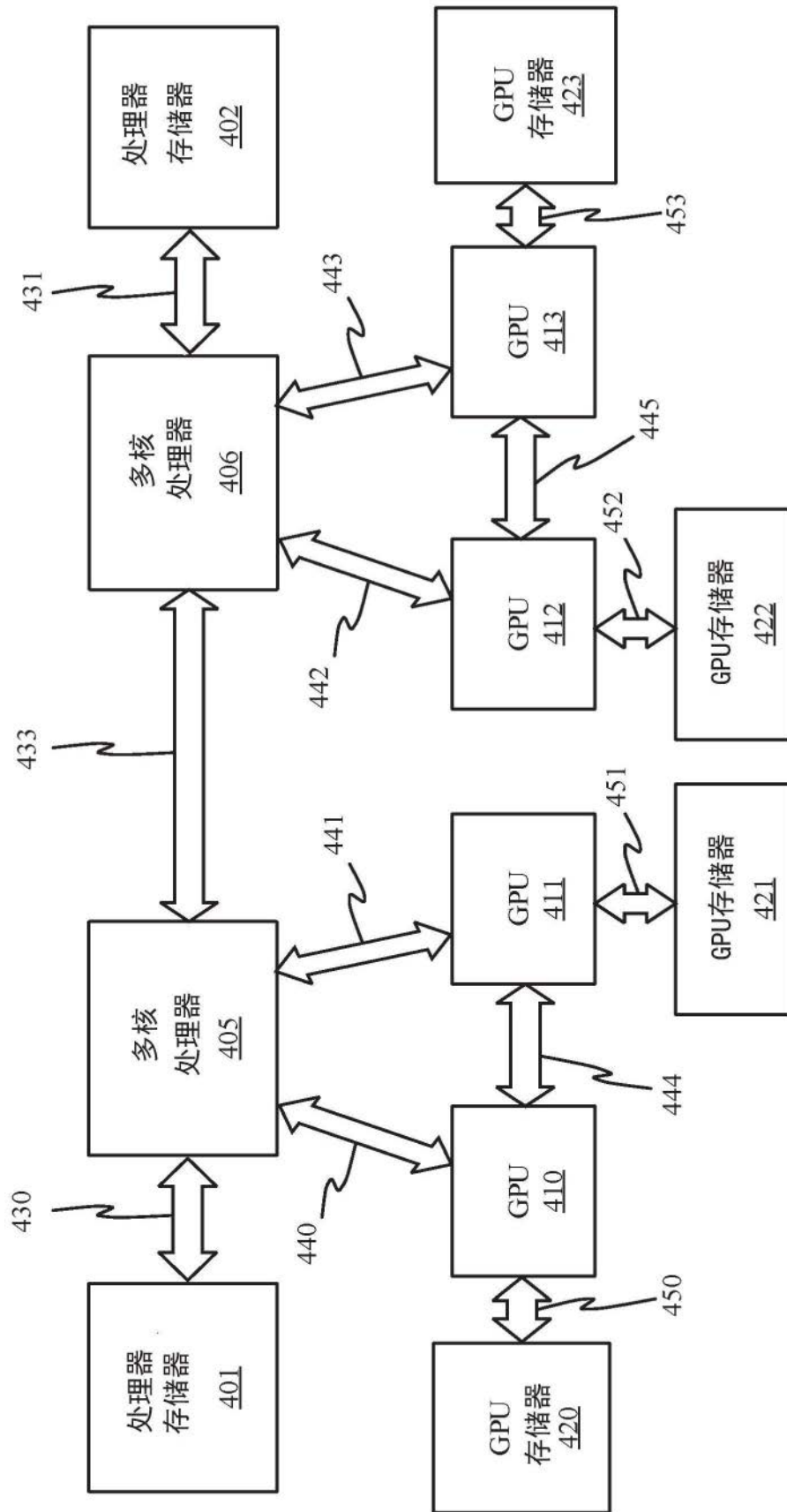


图4A

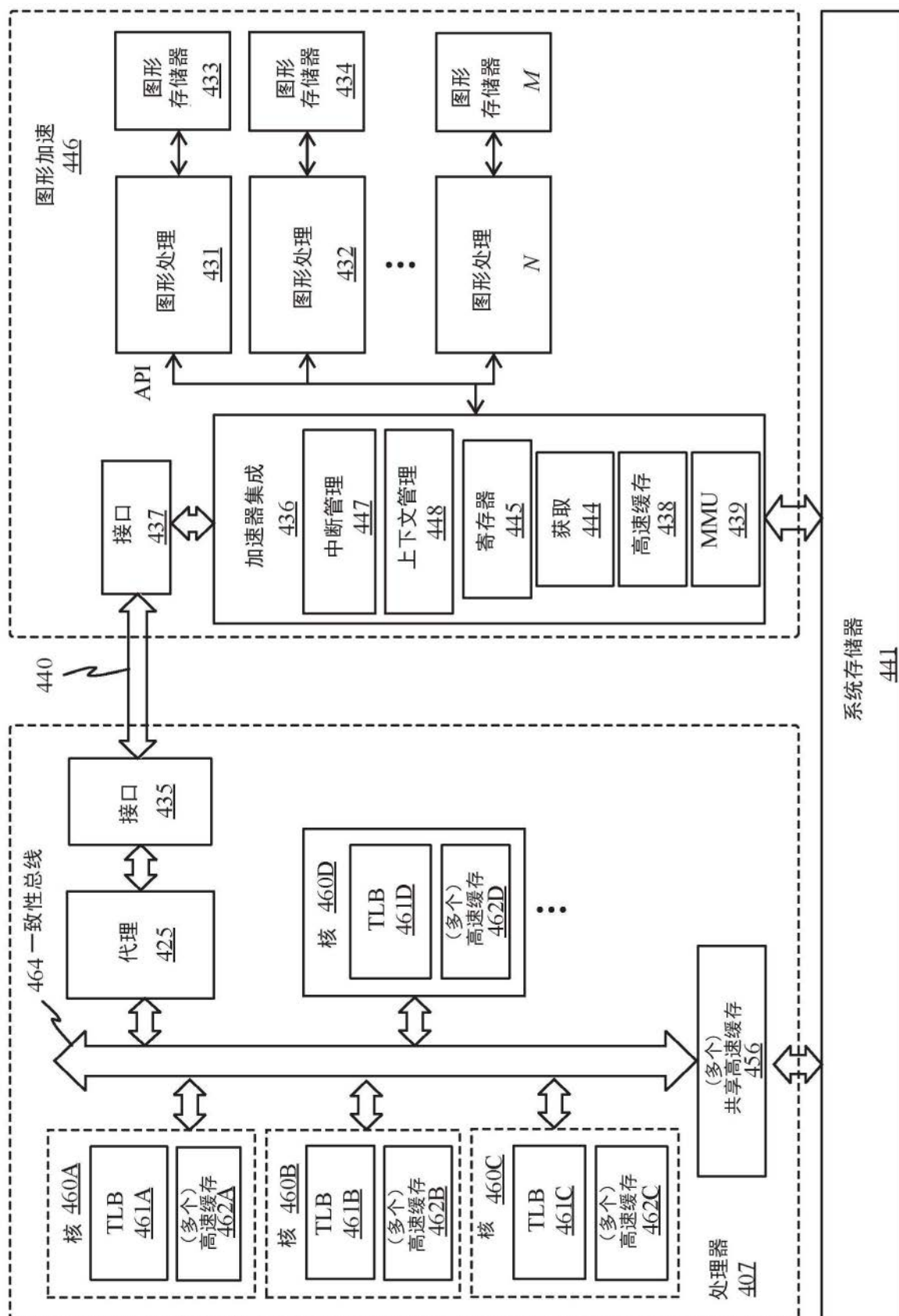


图4B

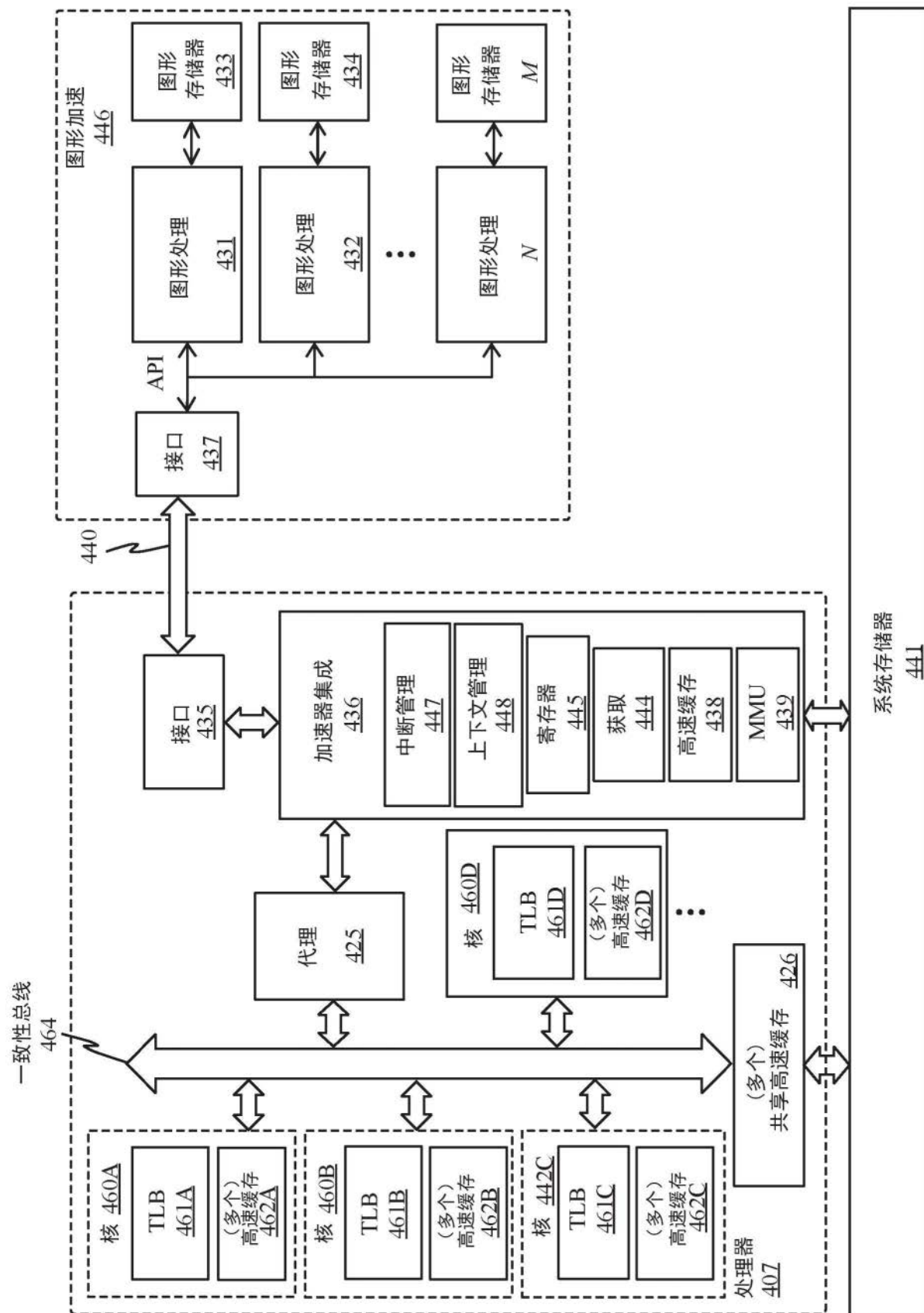


图4C

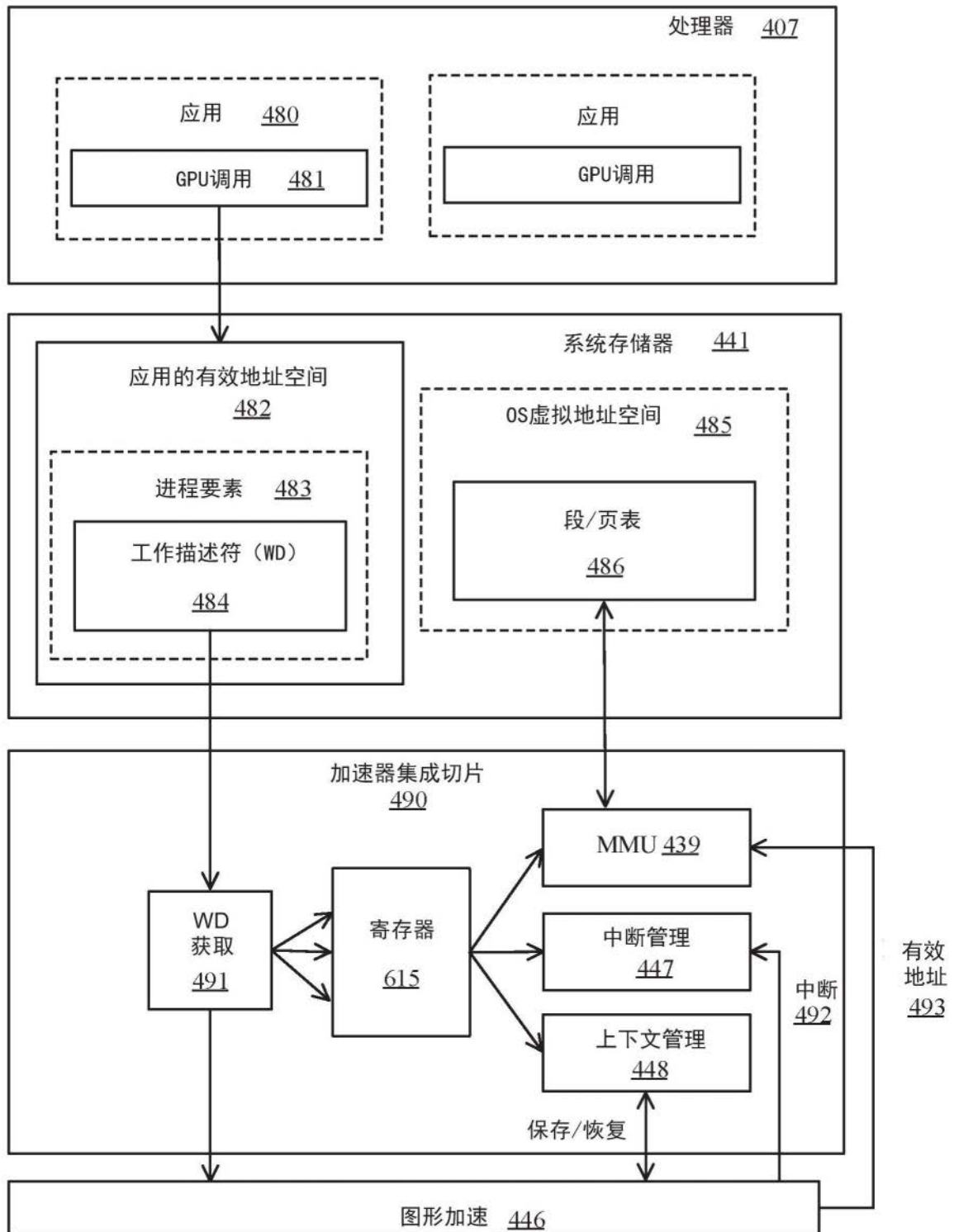


图4D

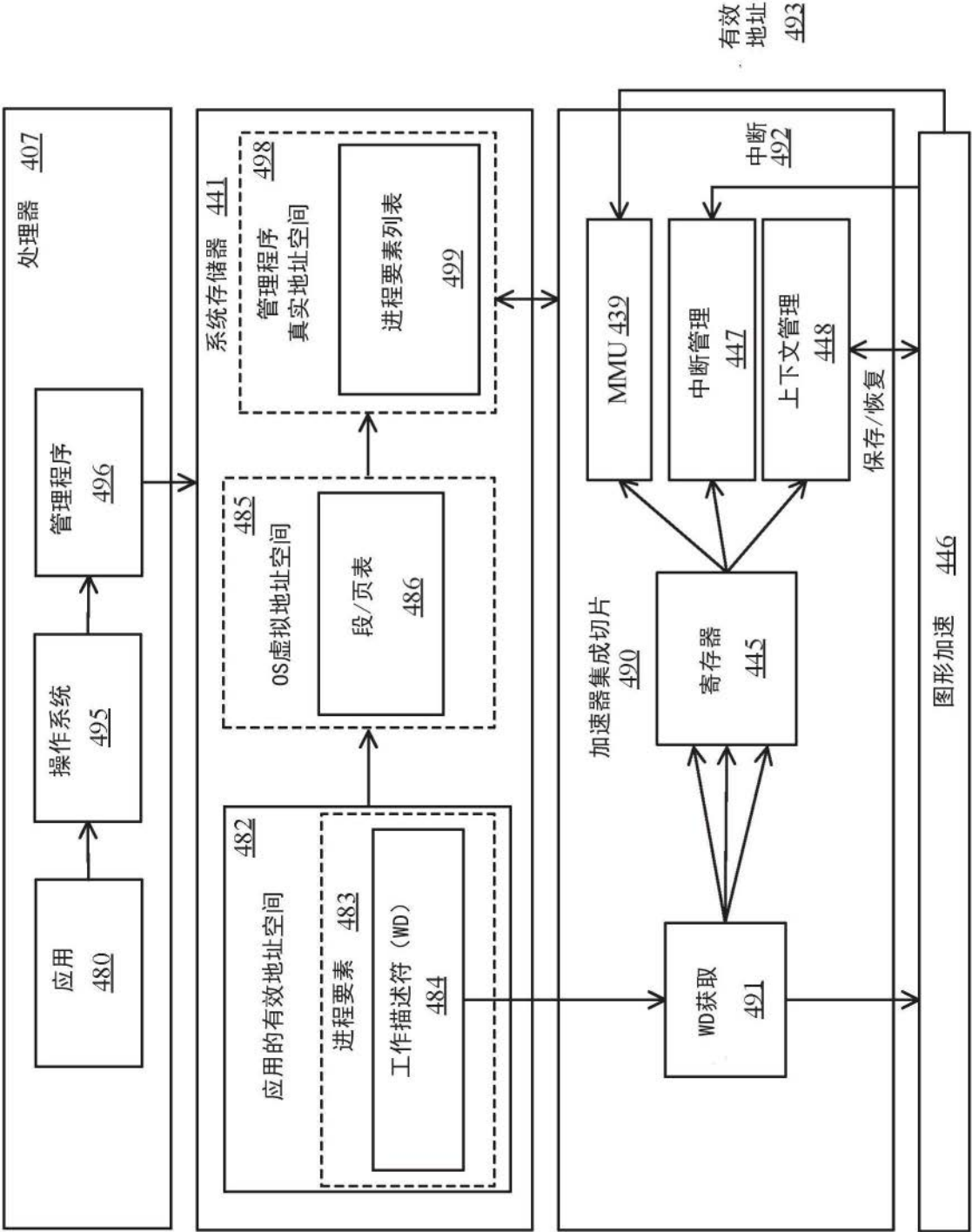


图4E

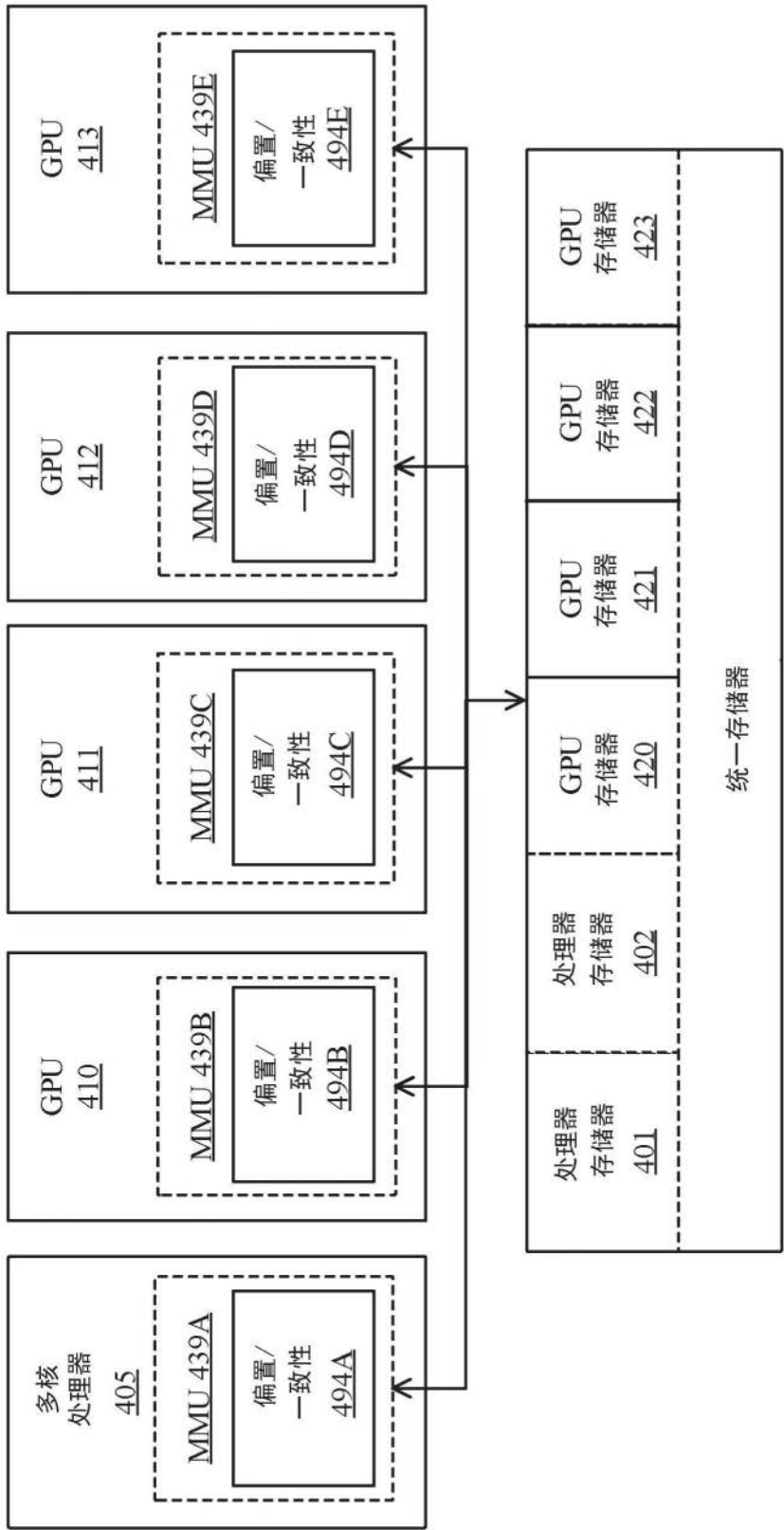


图4F

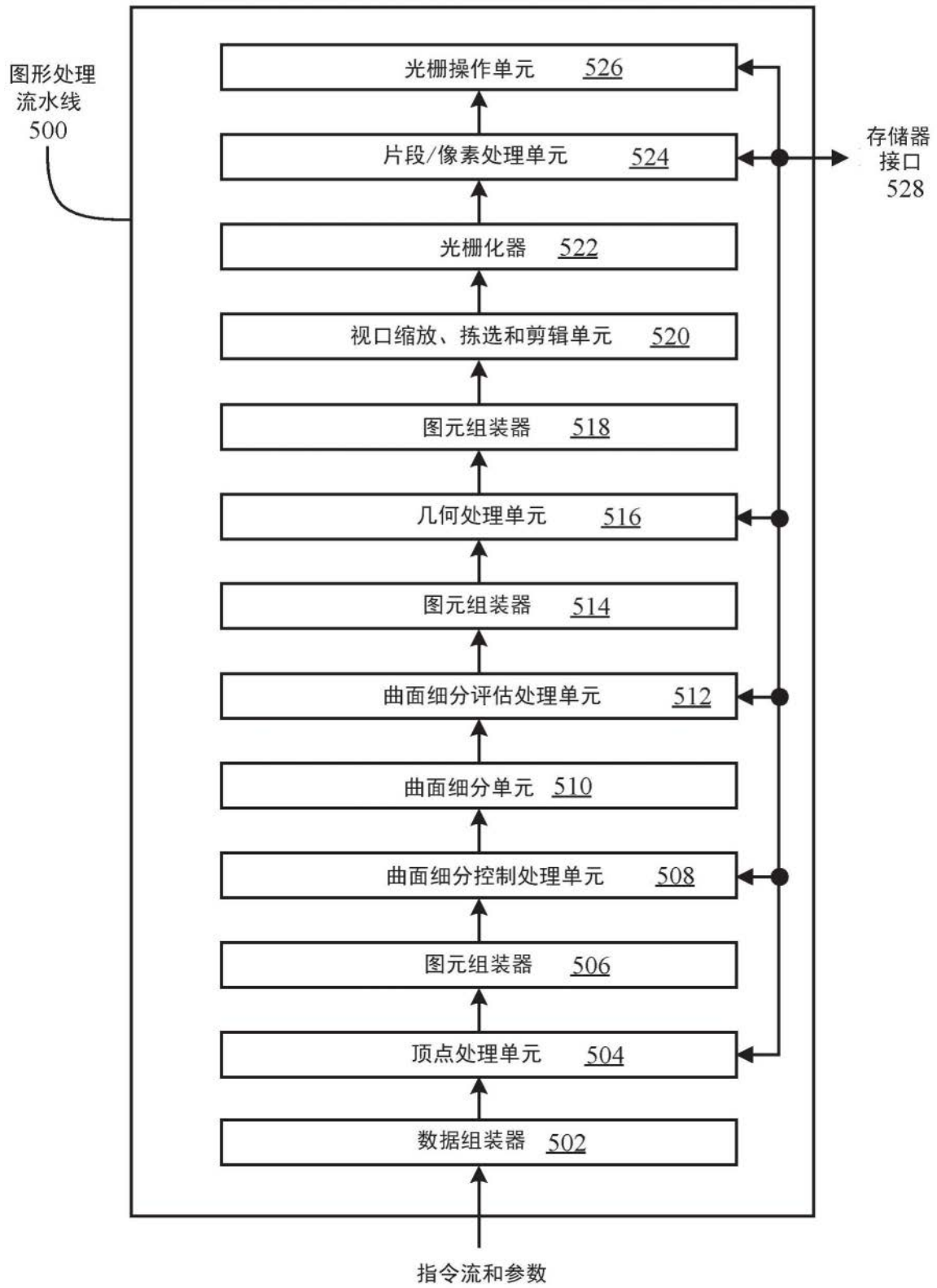


图5

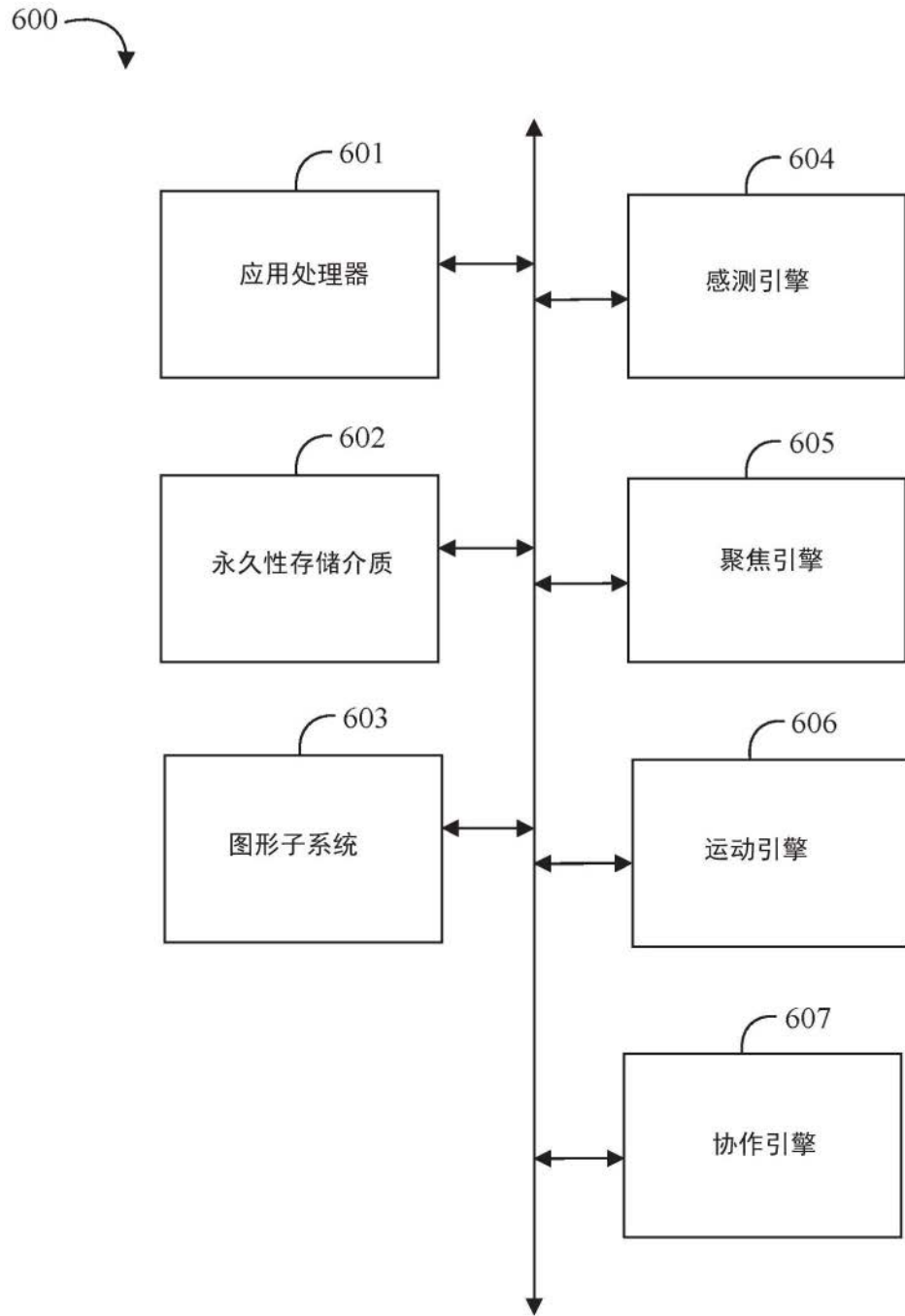


图6A

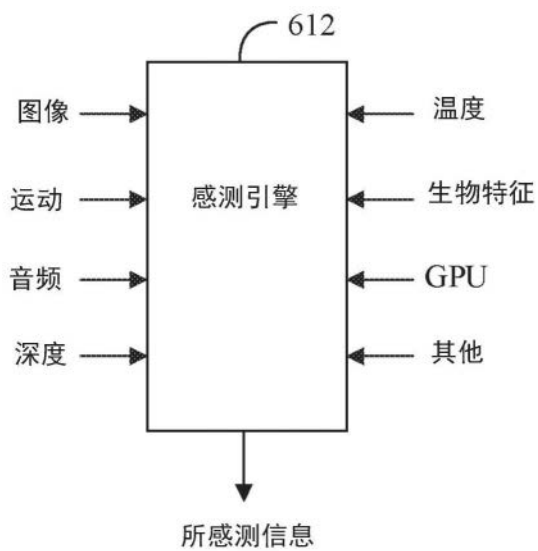


图6B

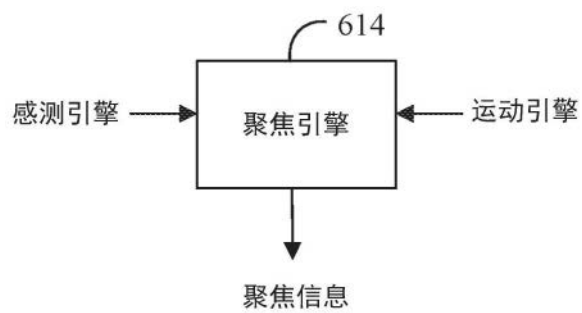


图6C

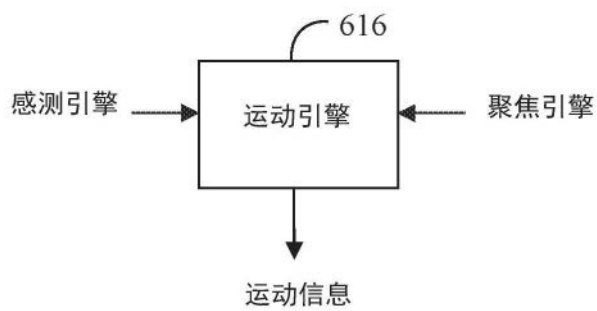


图6D

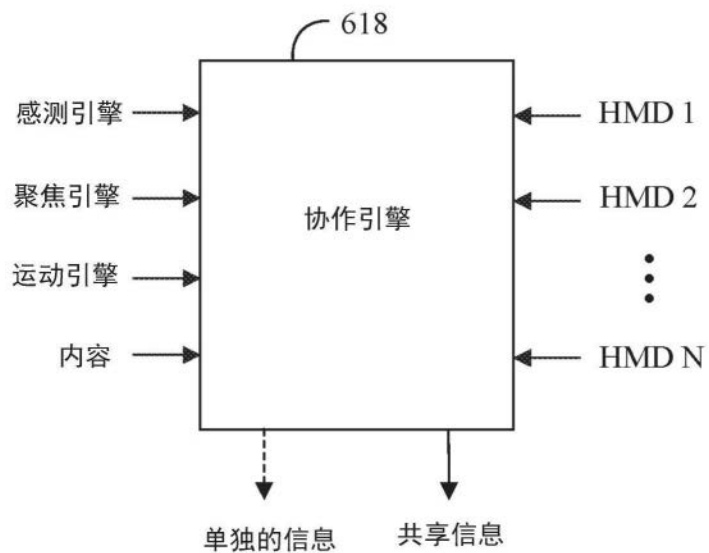


图6E

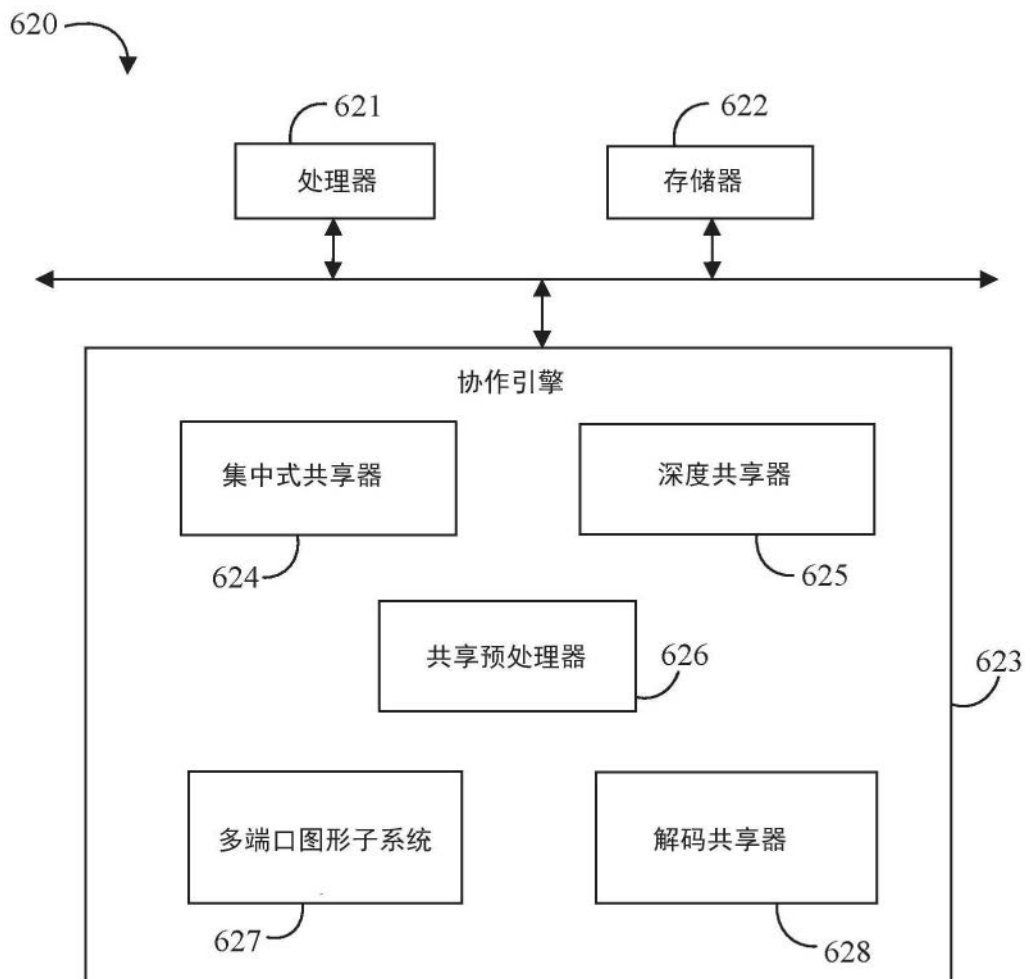


图6F

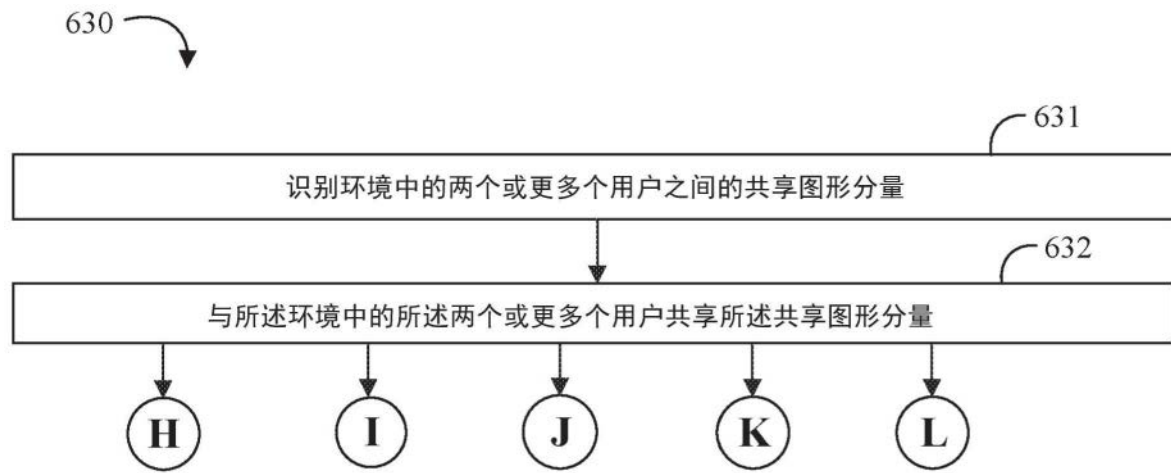


图6G

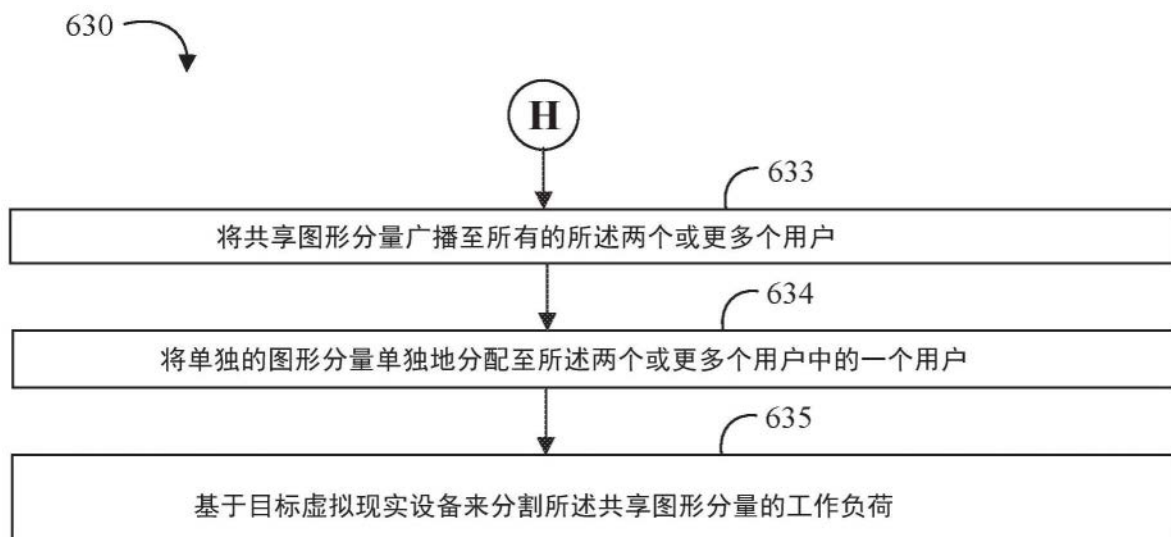


图6H

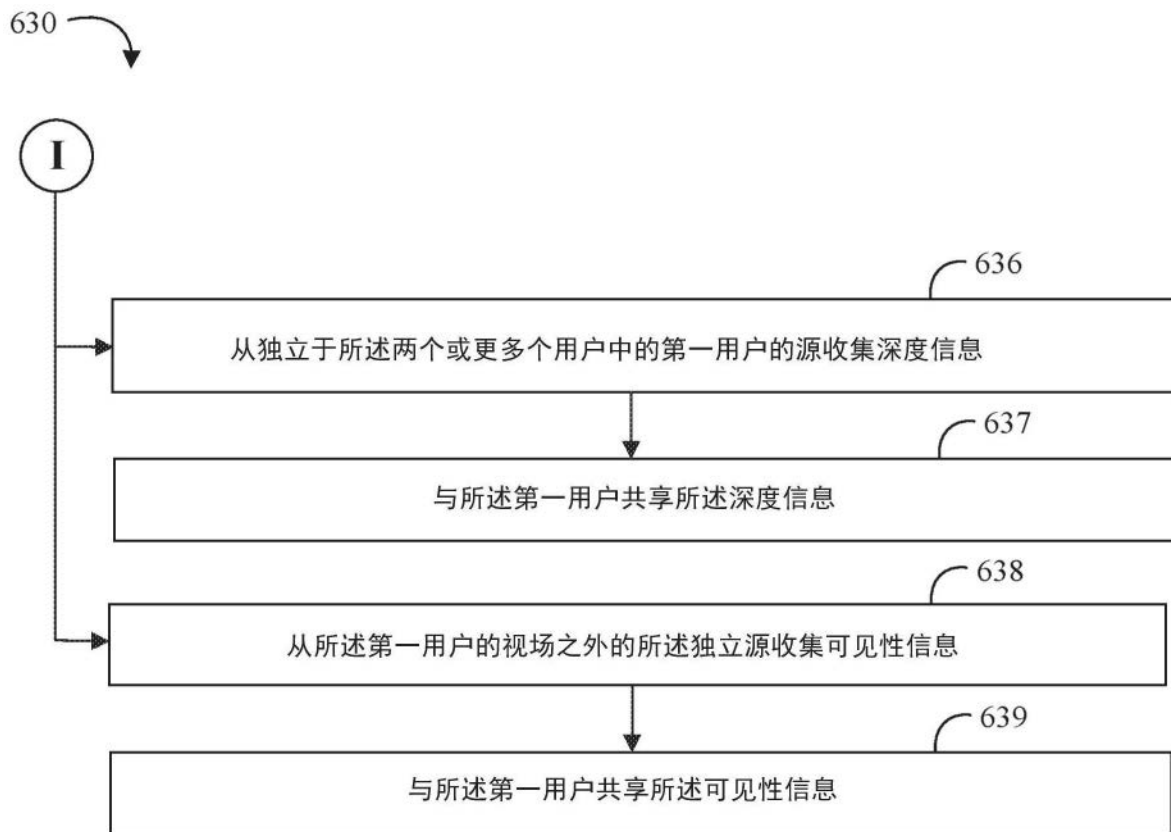


图6I

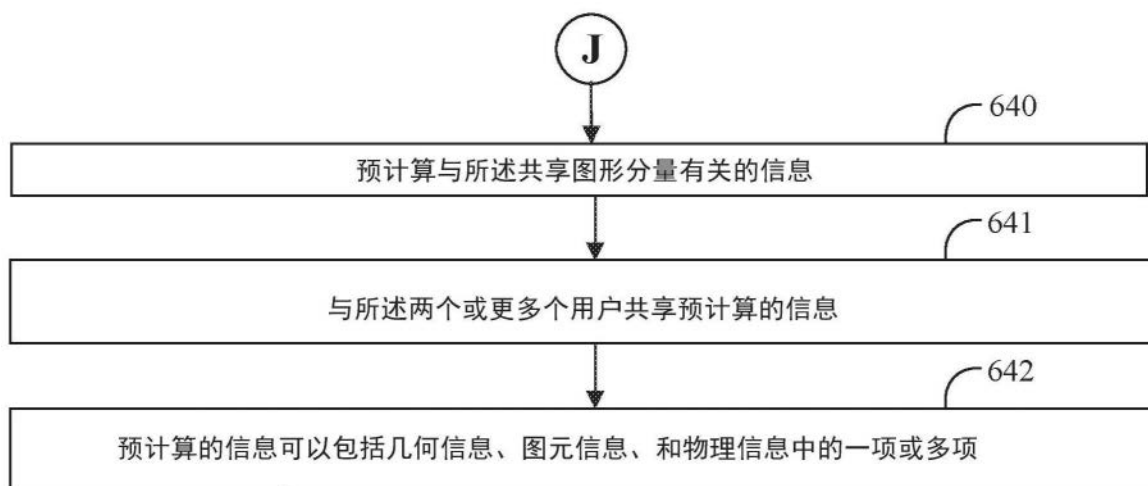


图6J

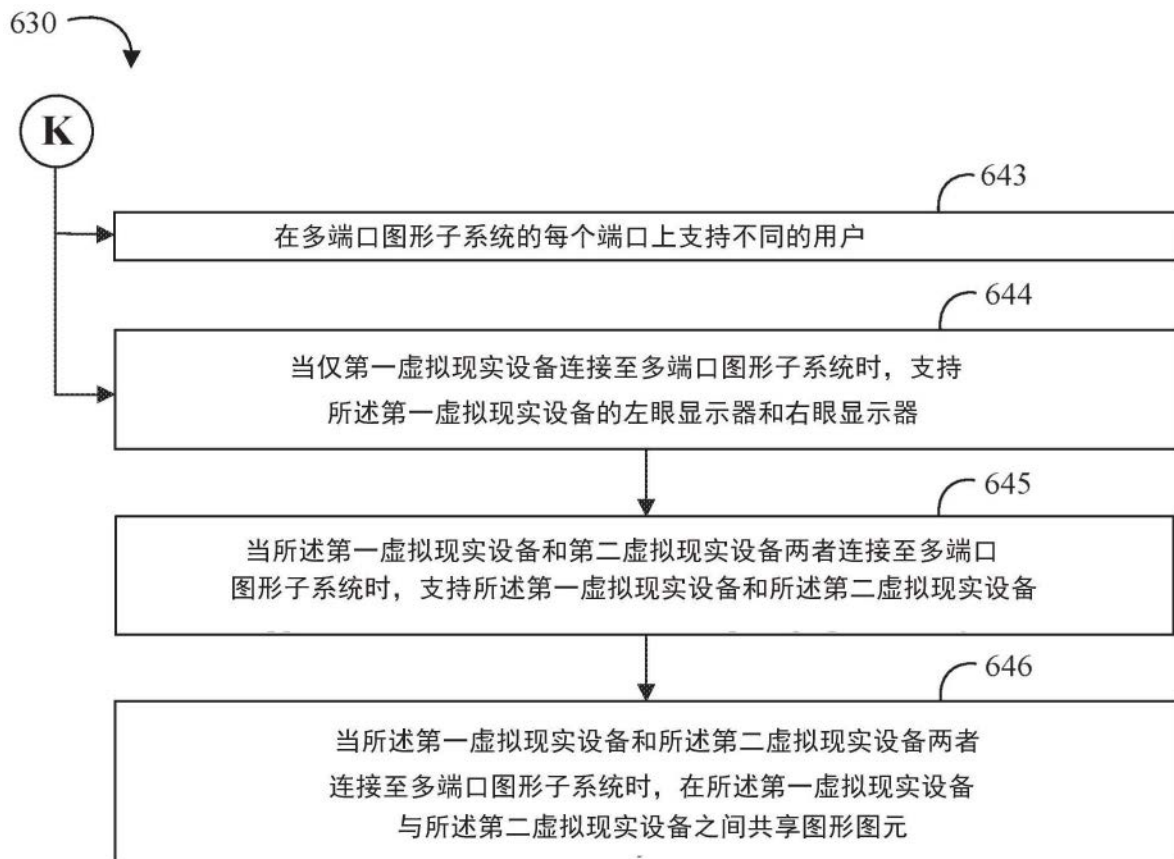


图6K

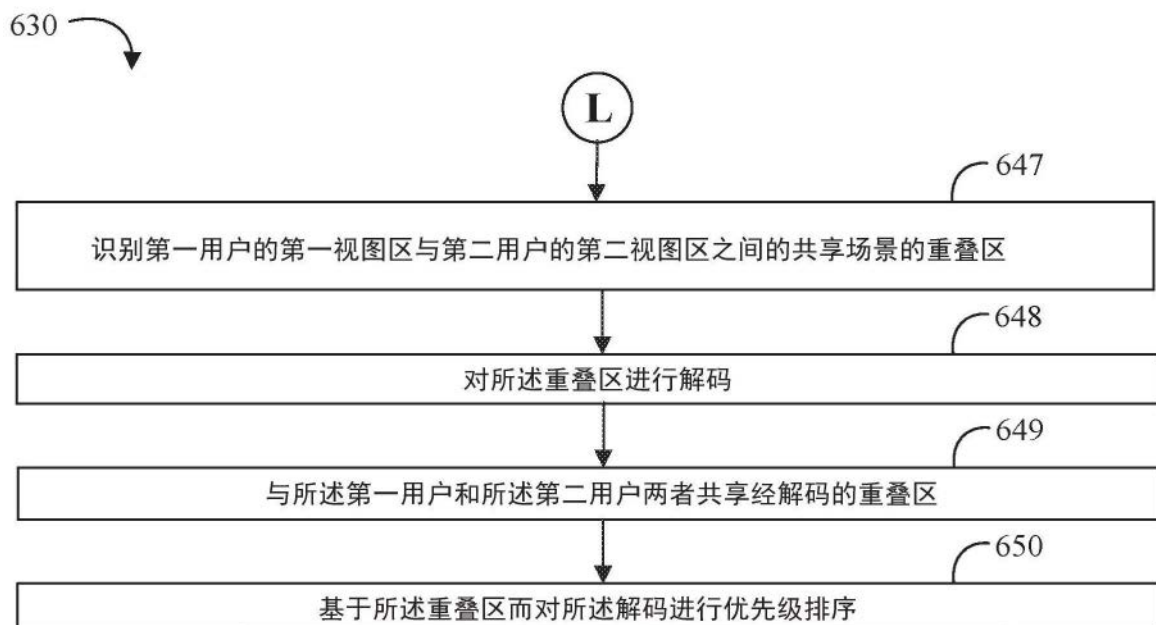


图6L

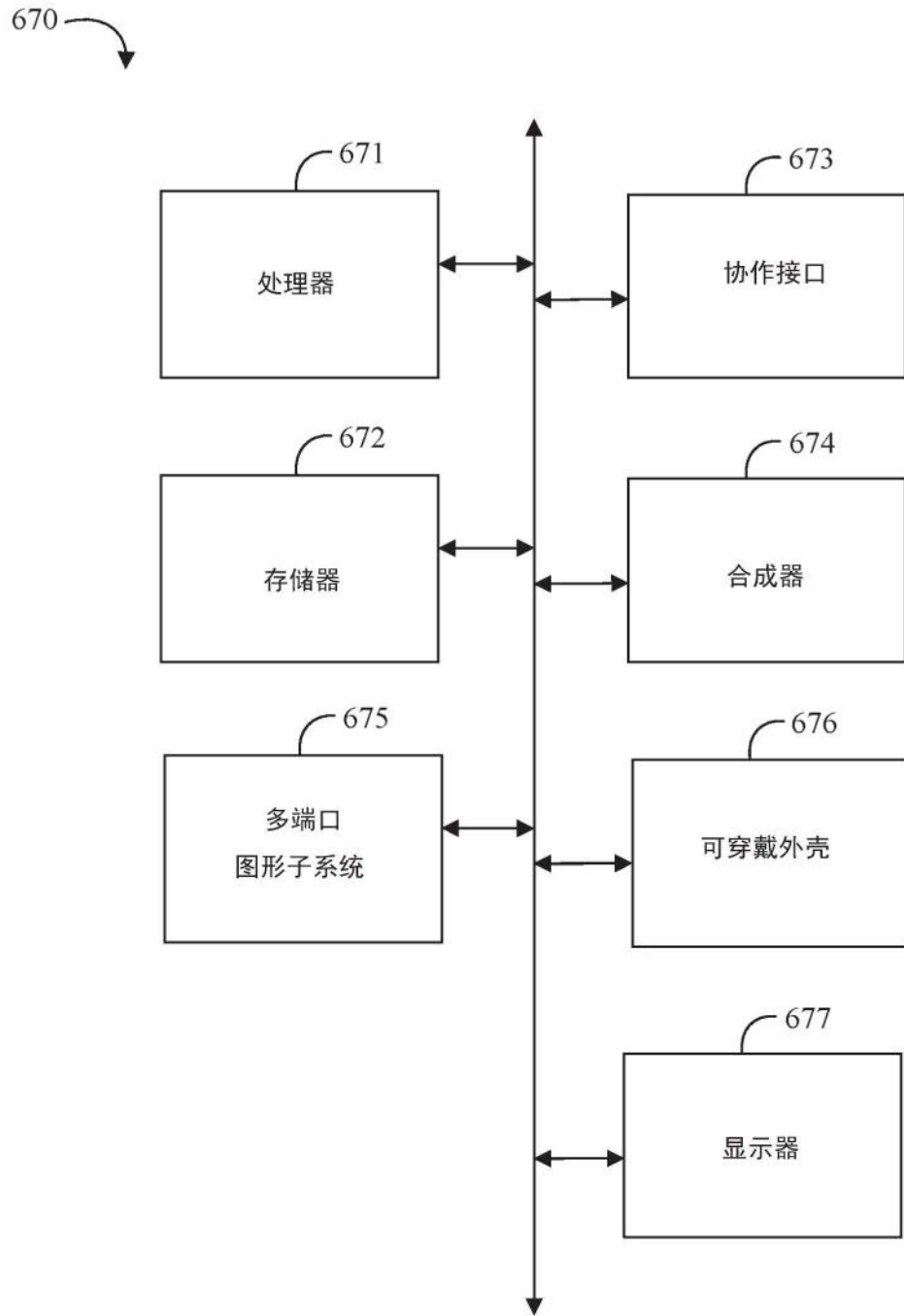


图6M

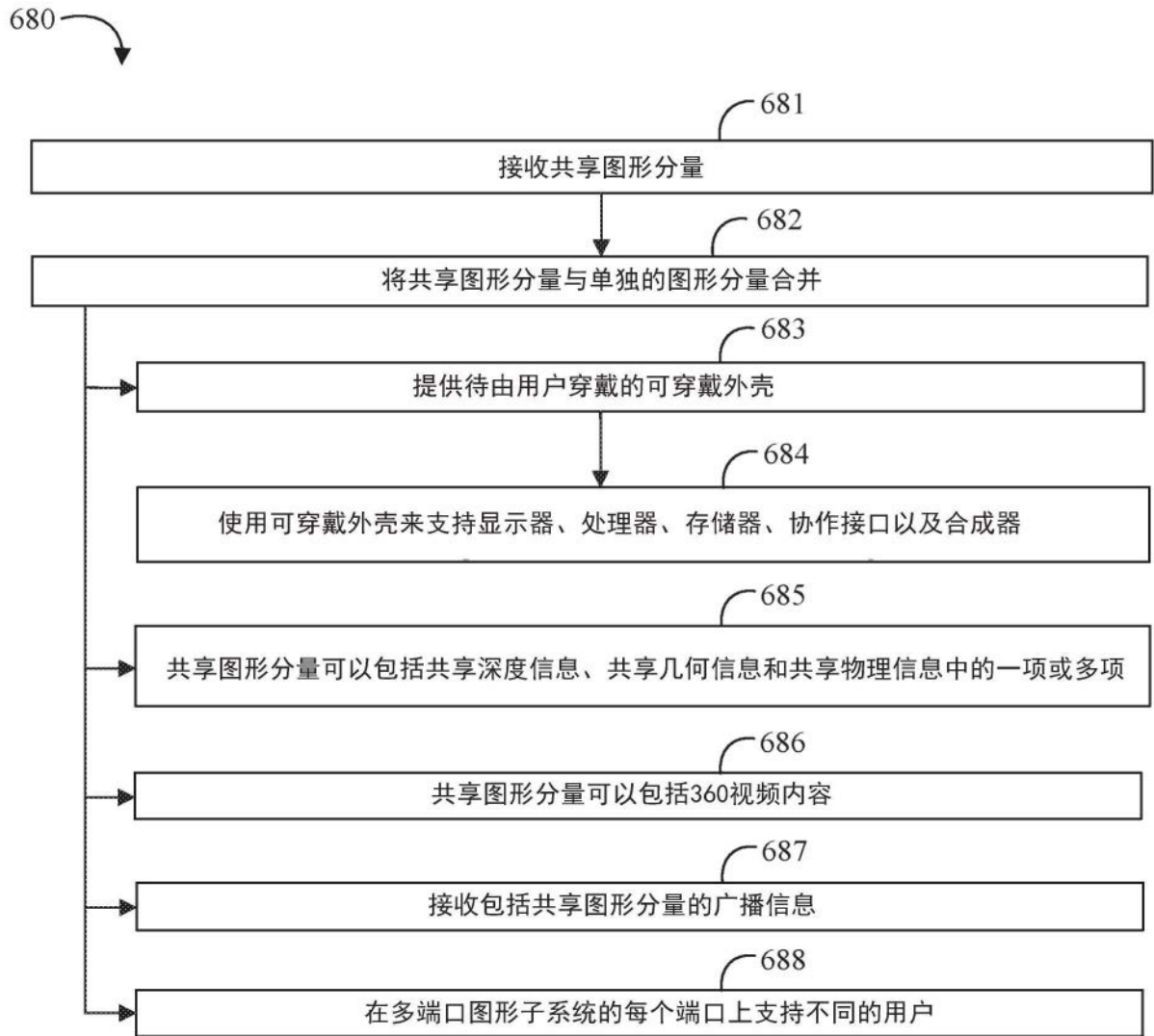


图6N

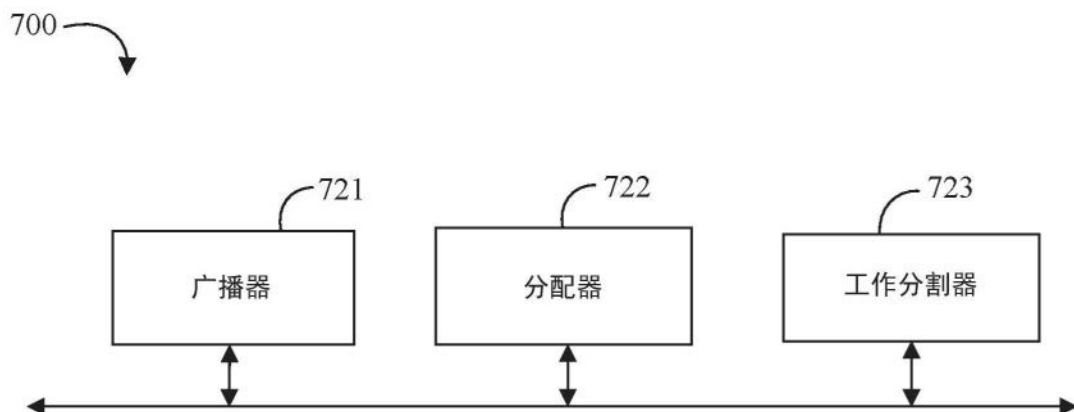


图7A

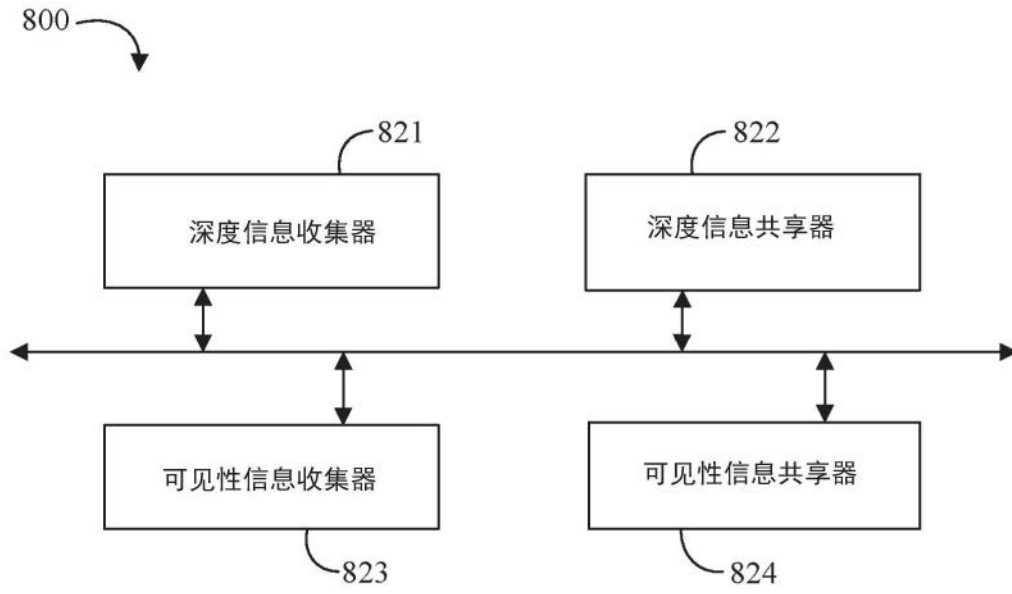


图8A

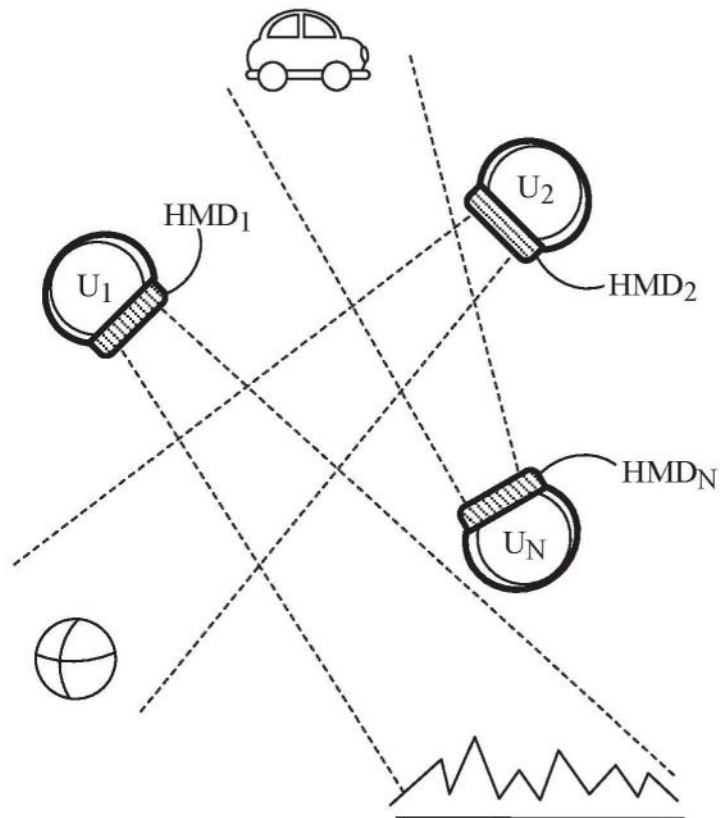


图8B

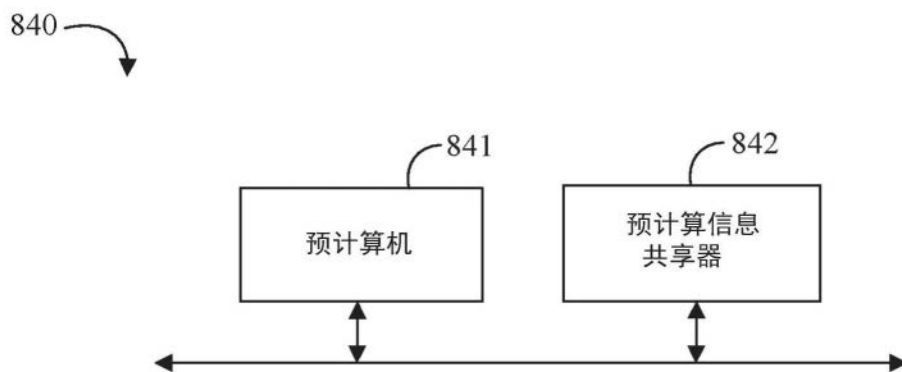


图8C

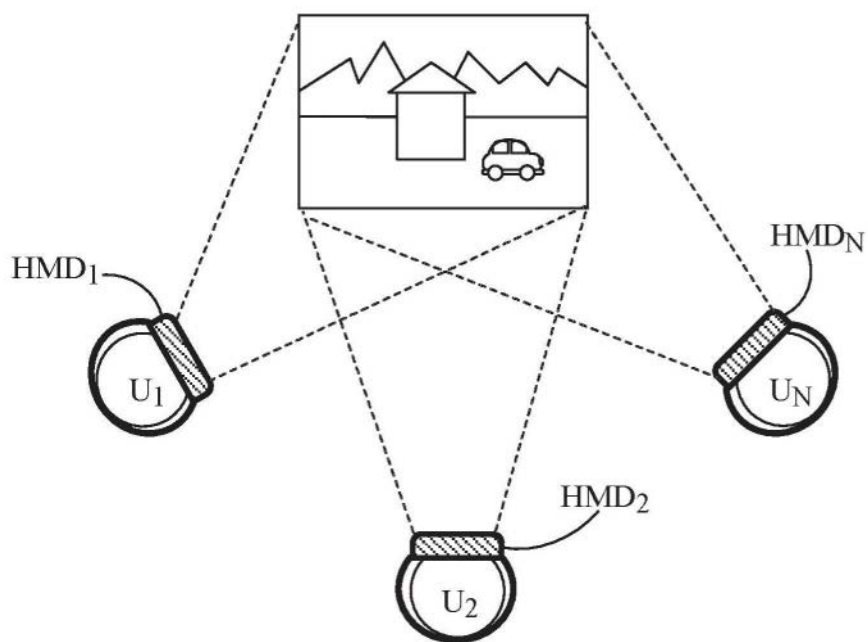


图8D

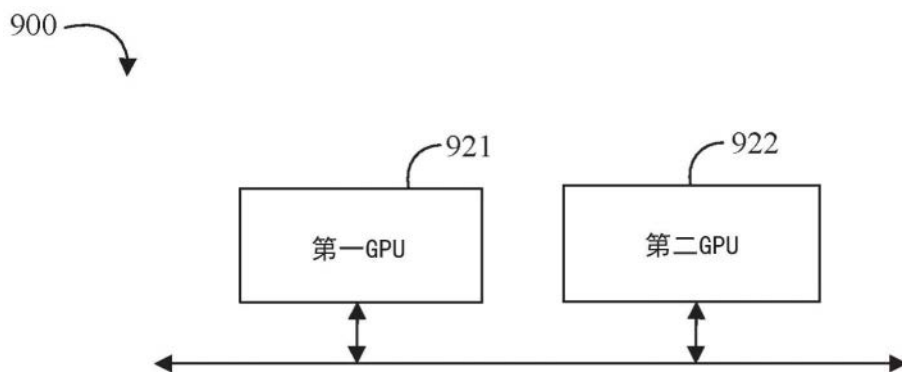


图9A

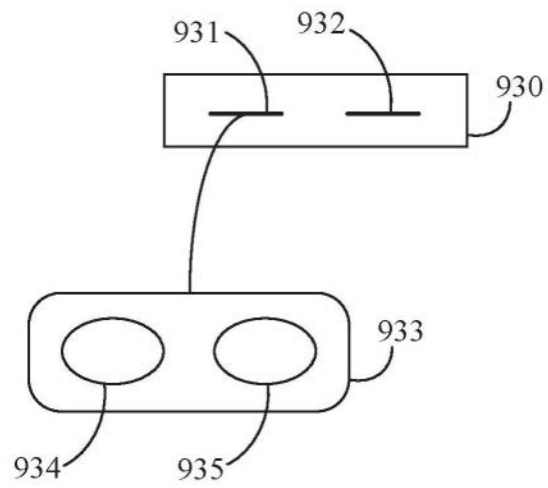


图9B

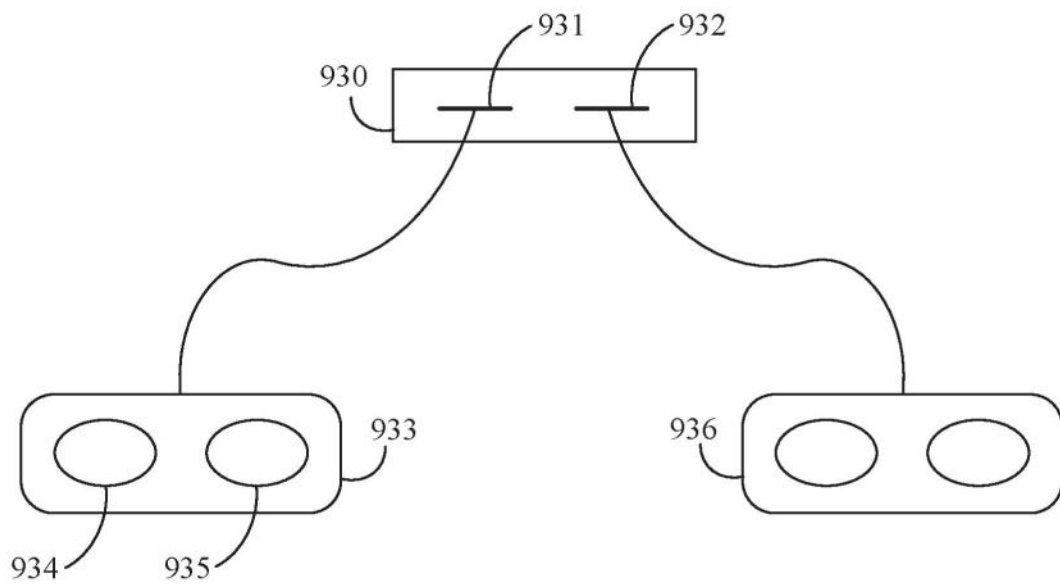


图9C

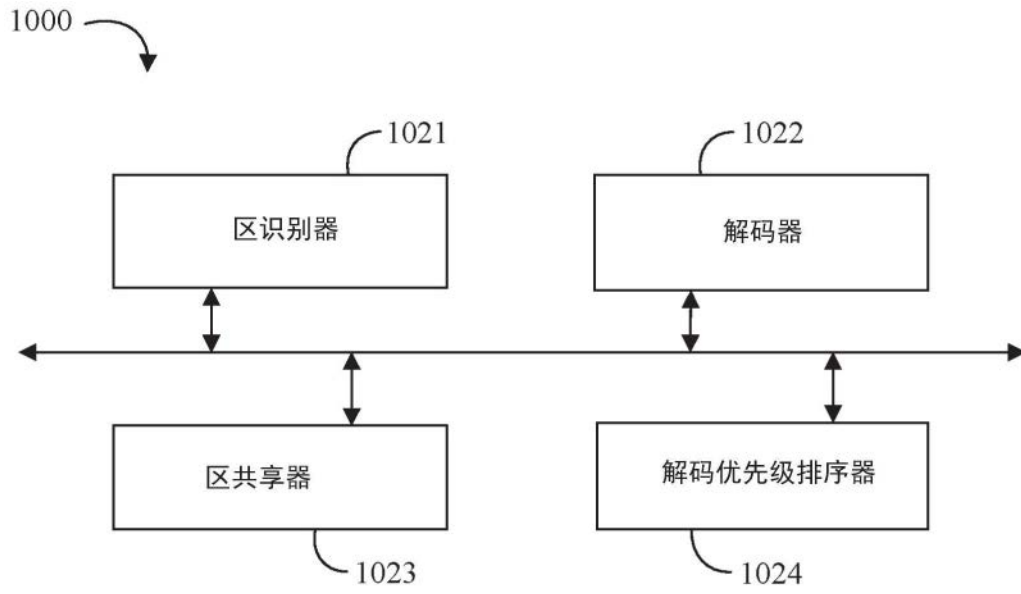


图10A

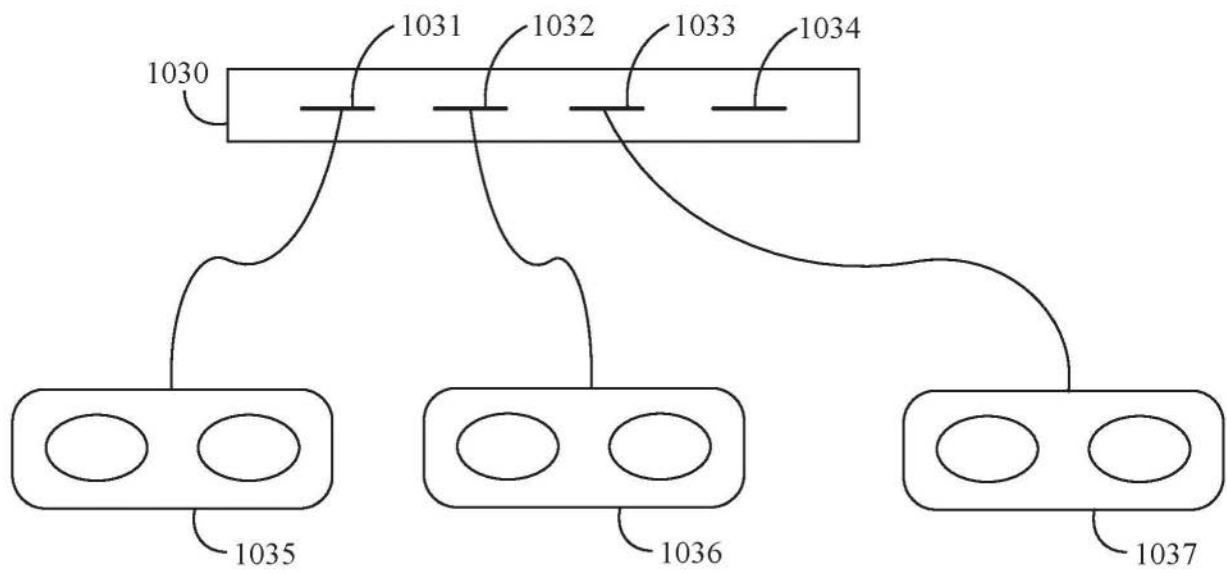


图10B

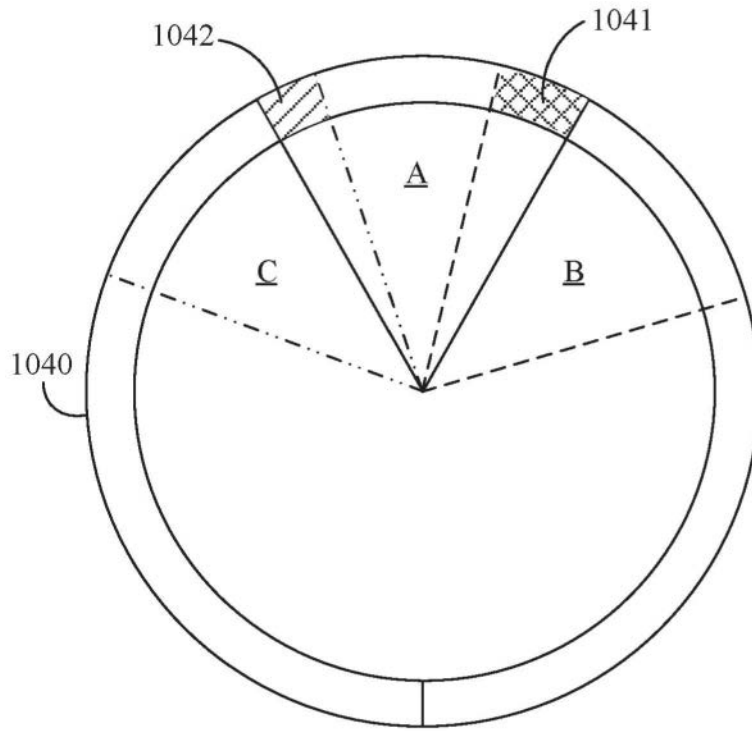


图10C

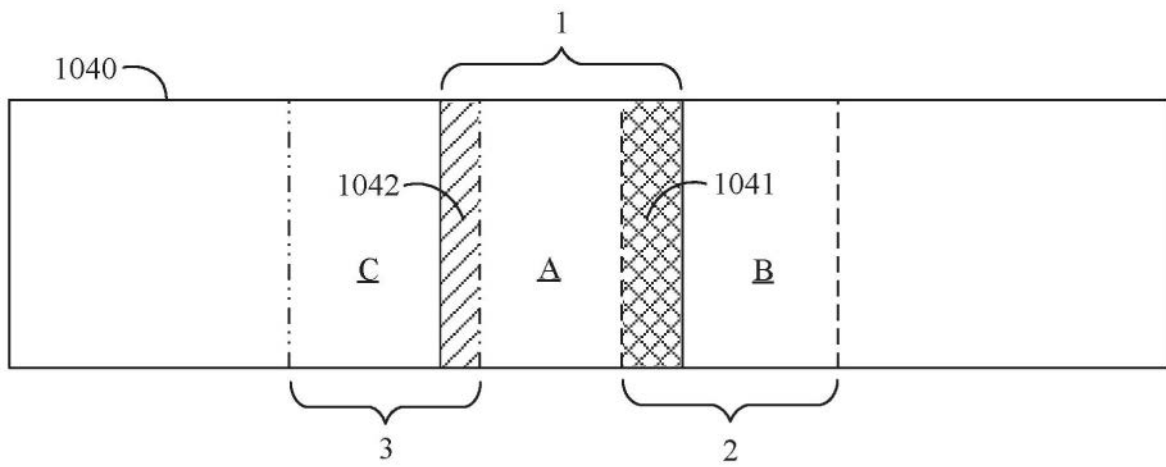


图10D

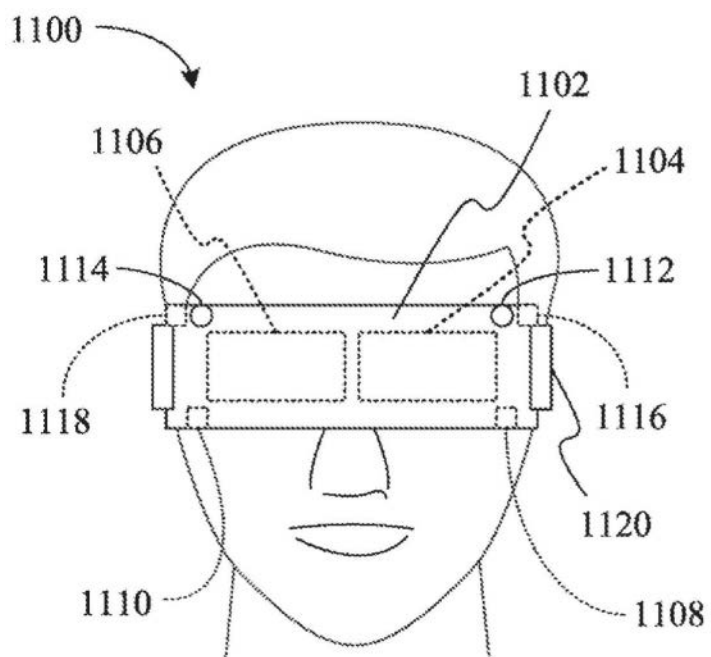


图11

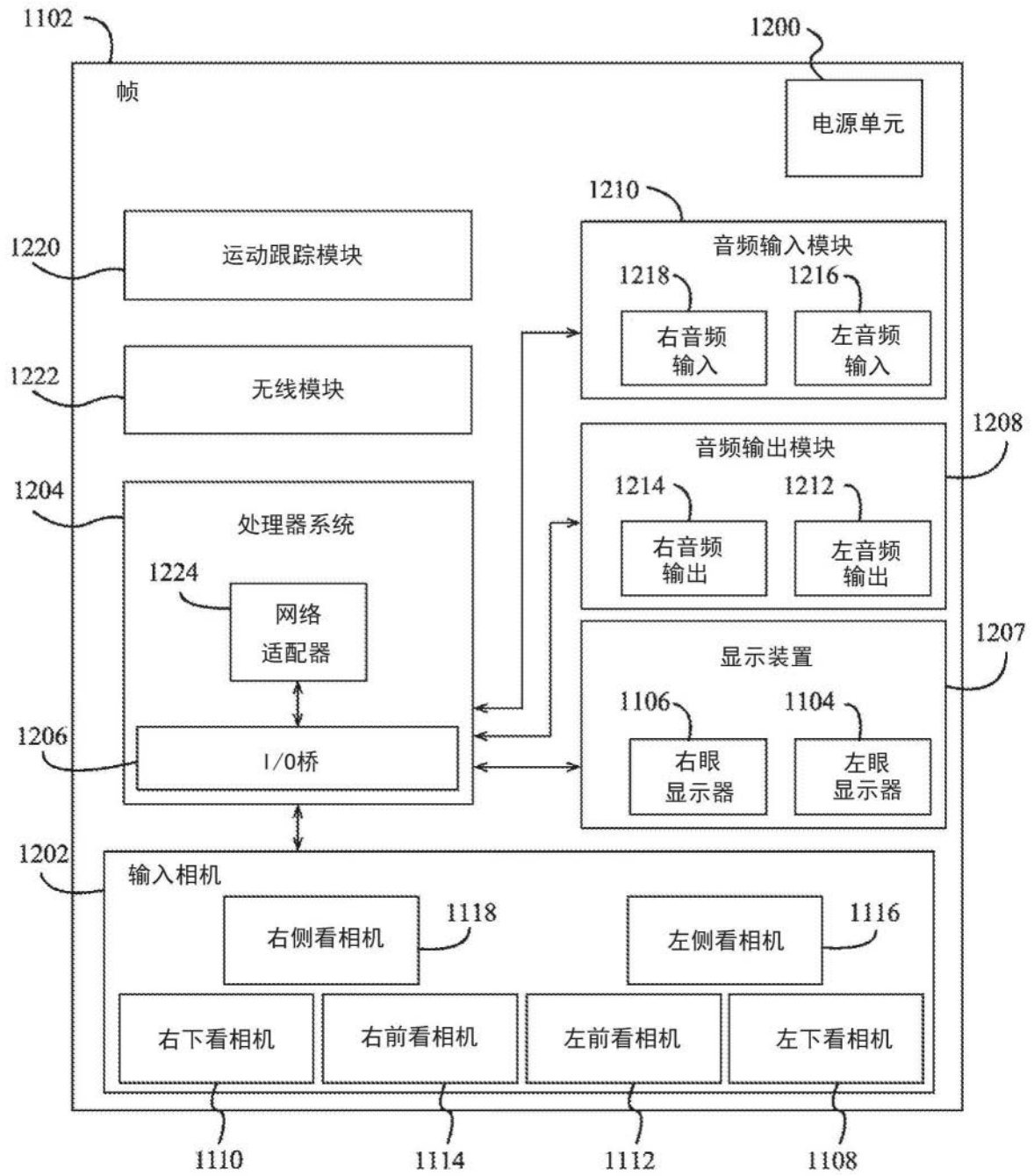


图12

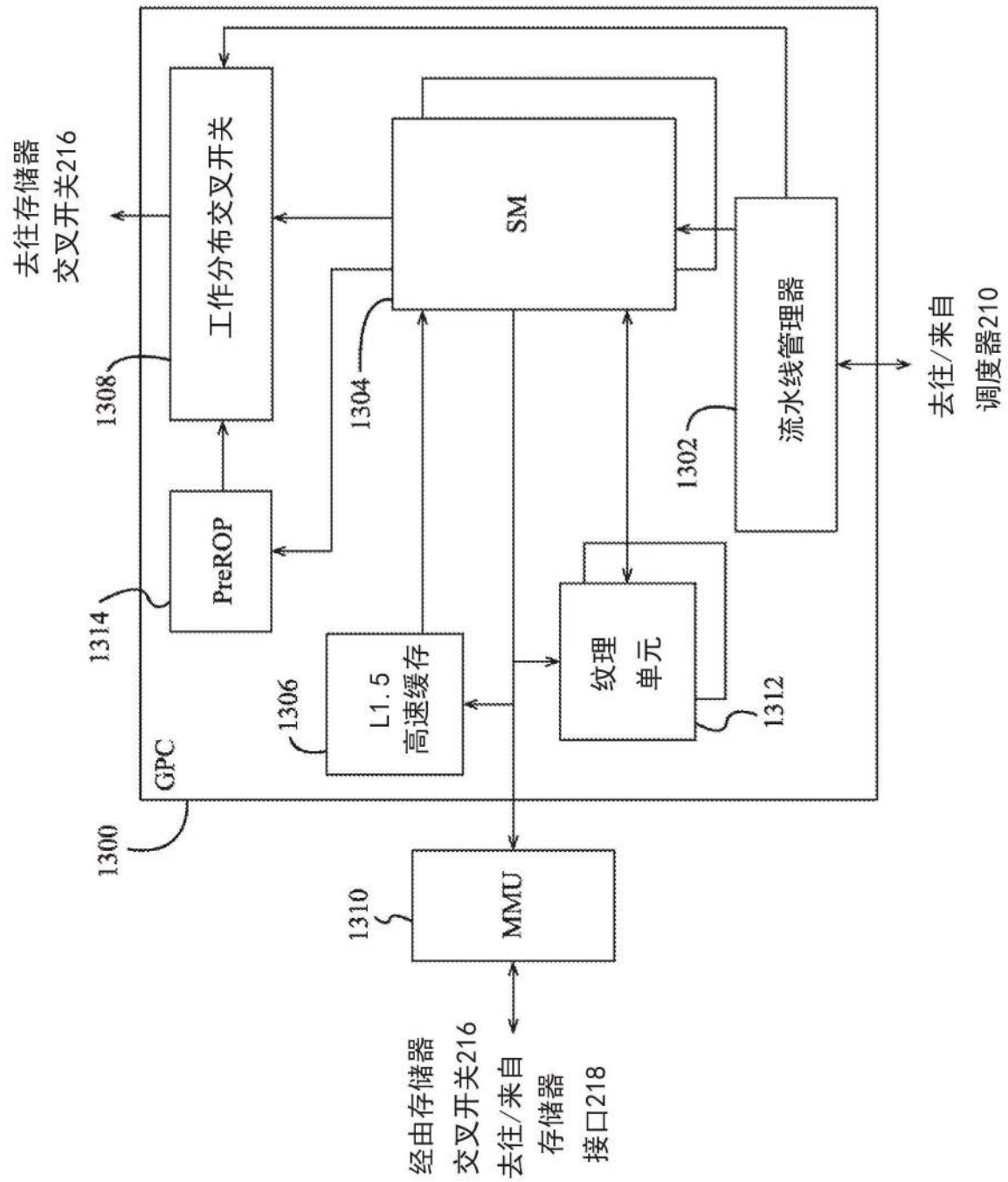


图13

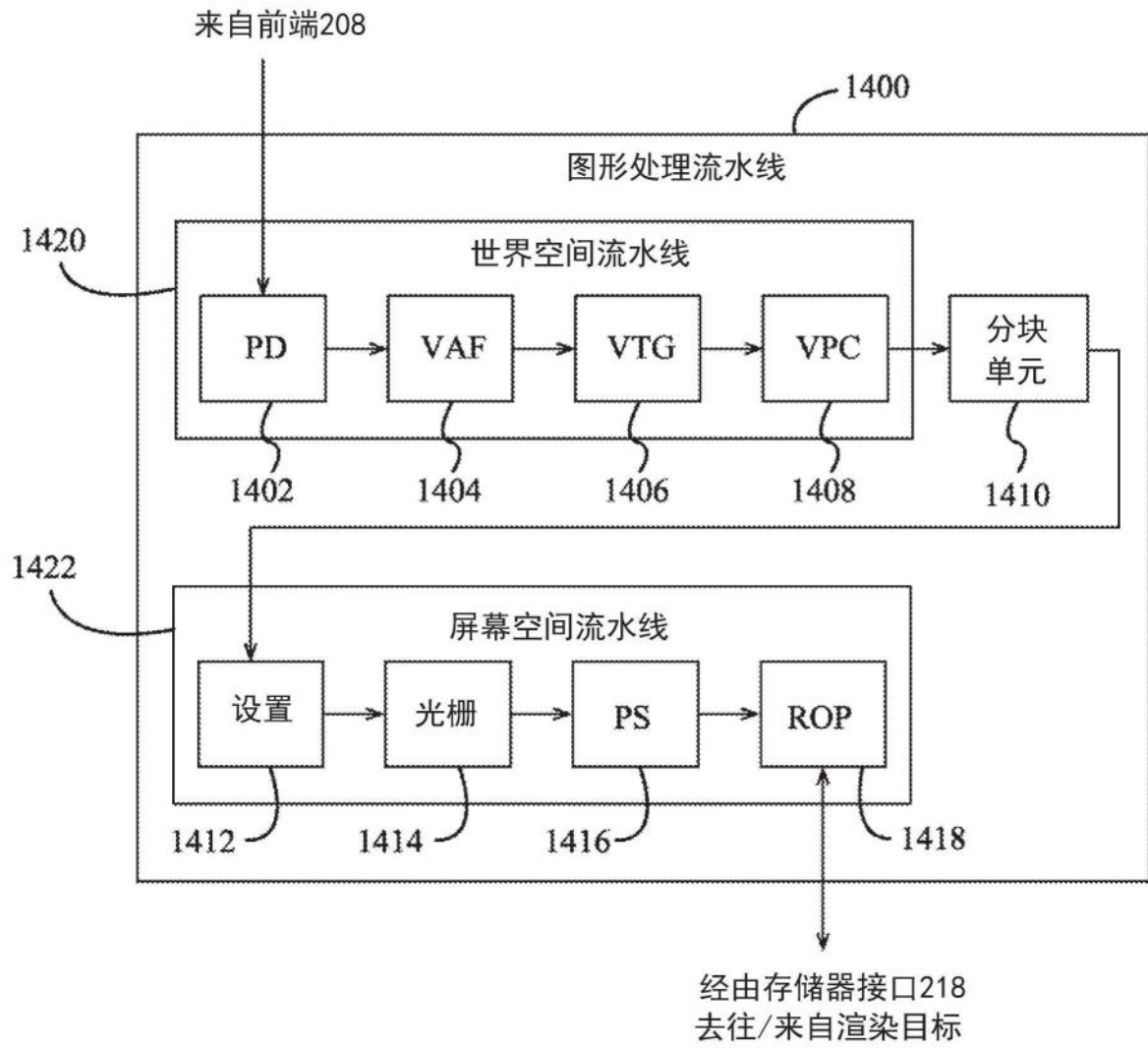


图14

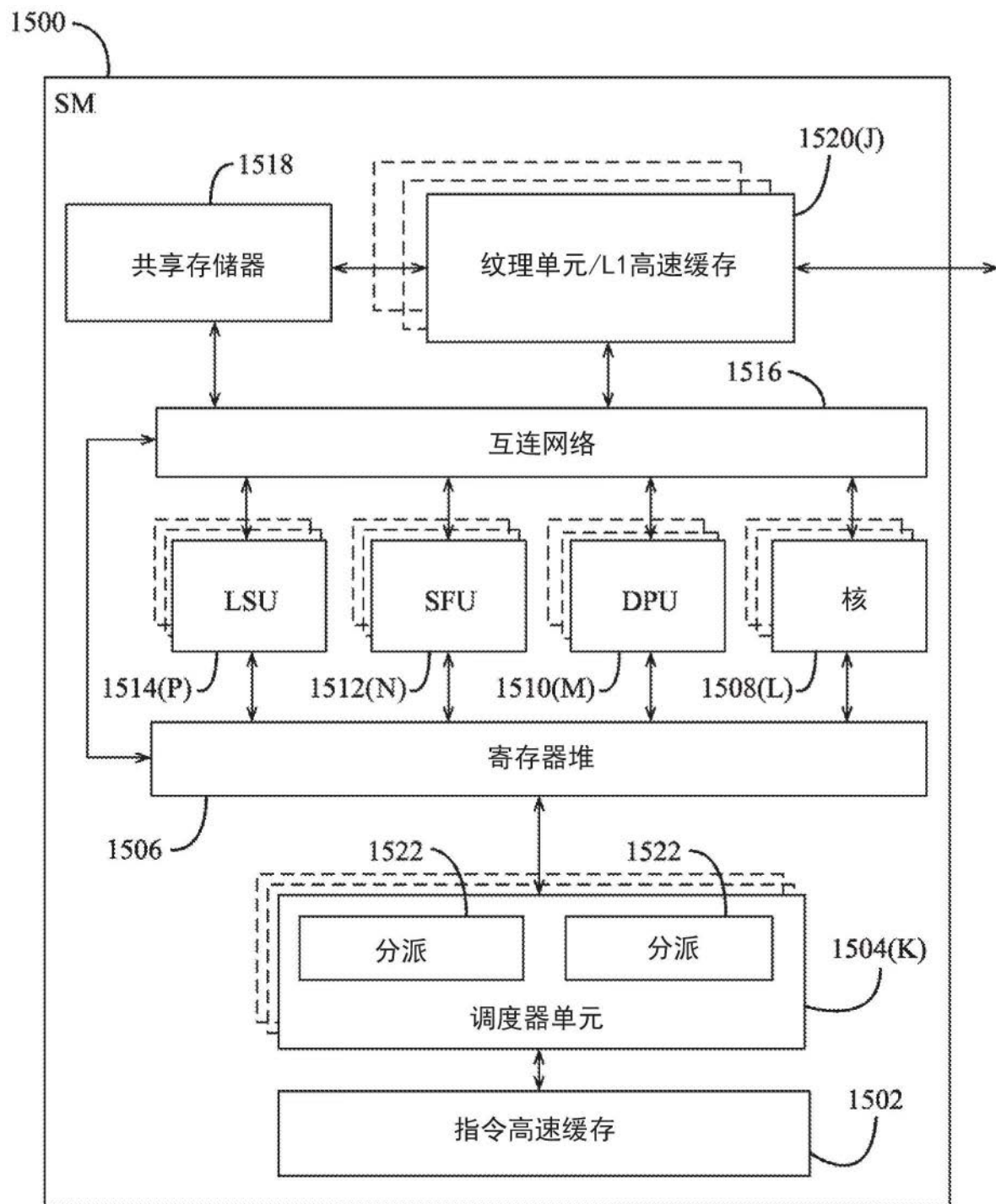


图15

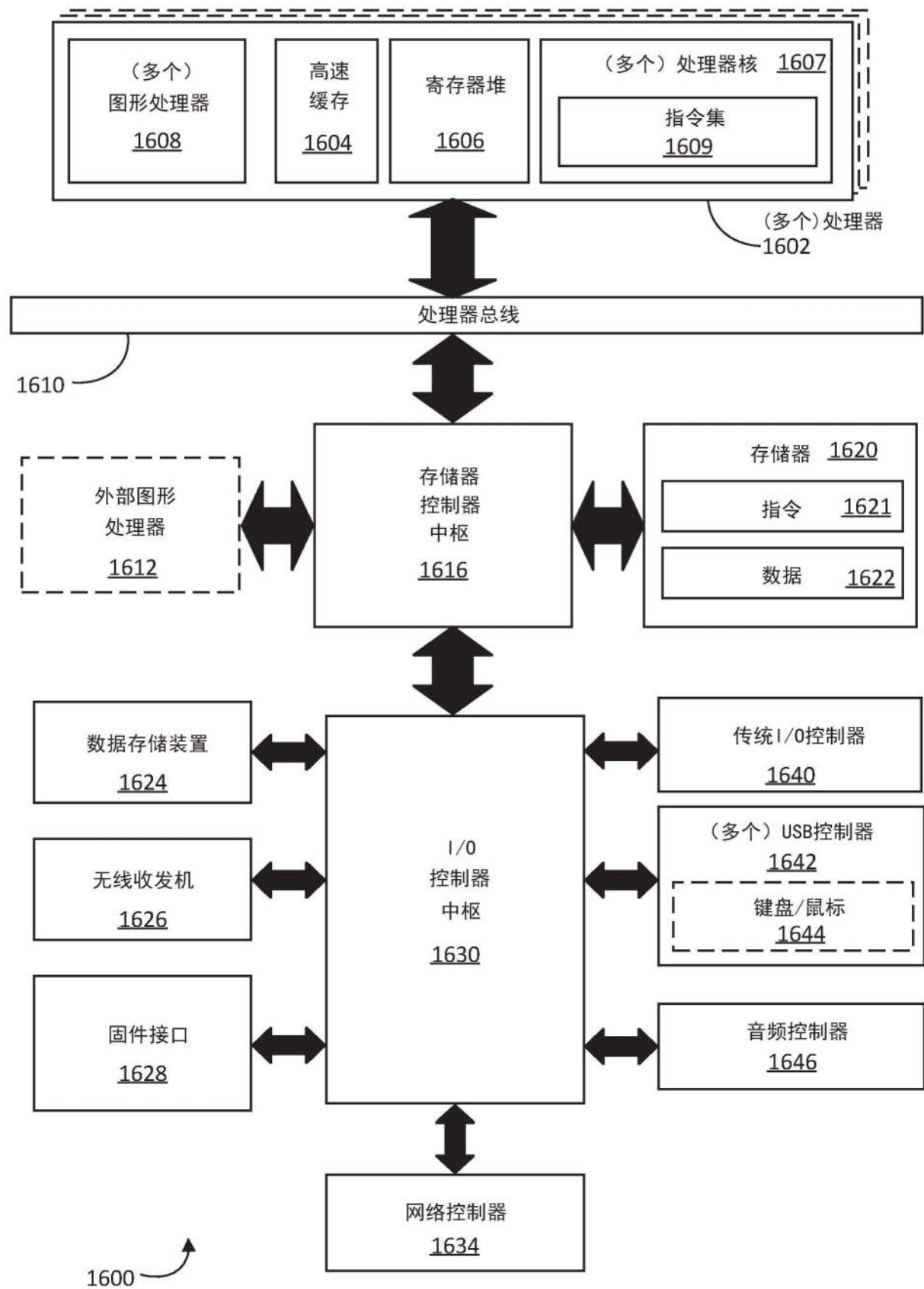


图16

处理器
1700

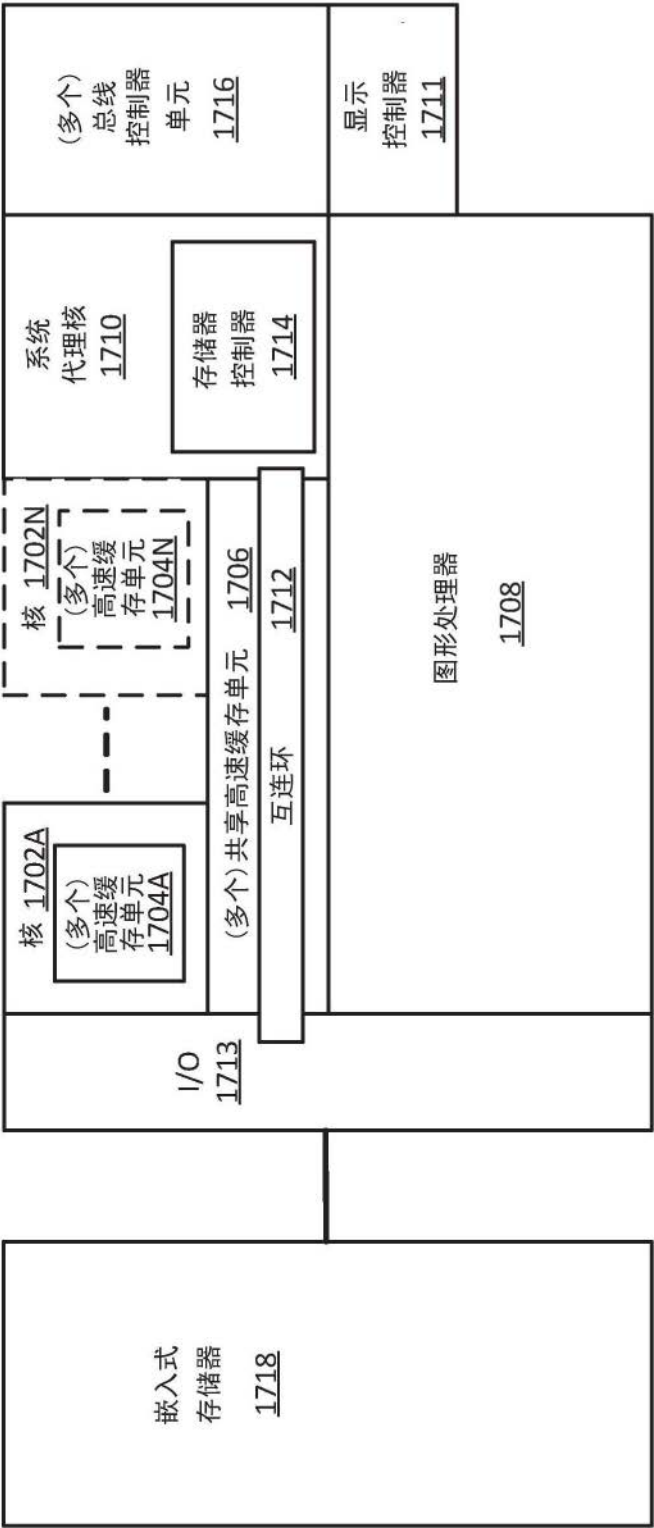


图17

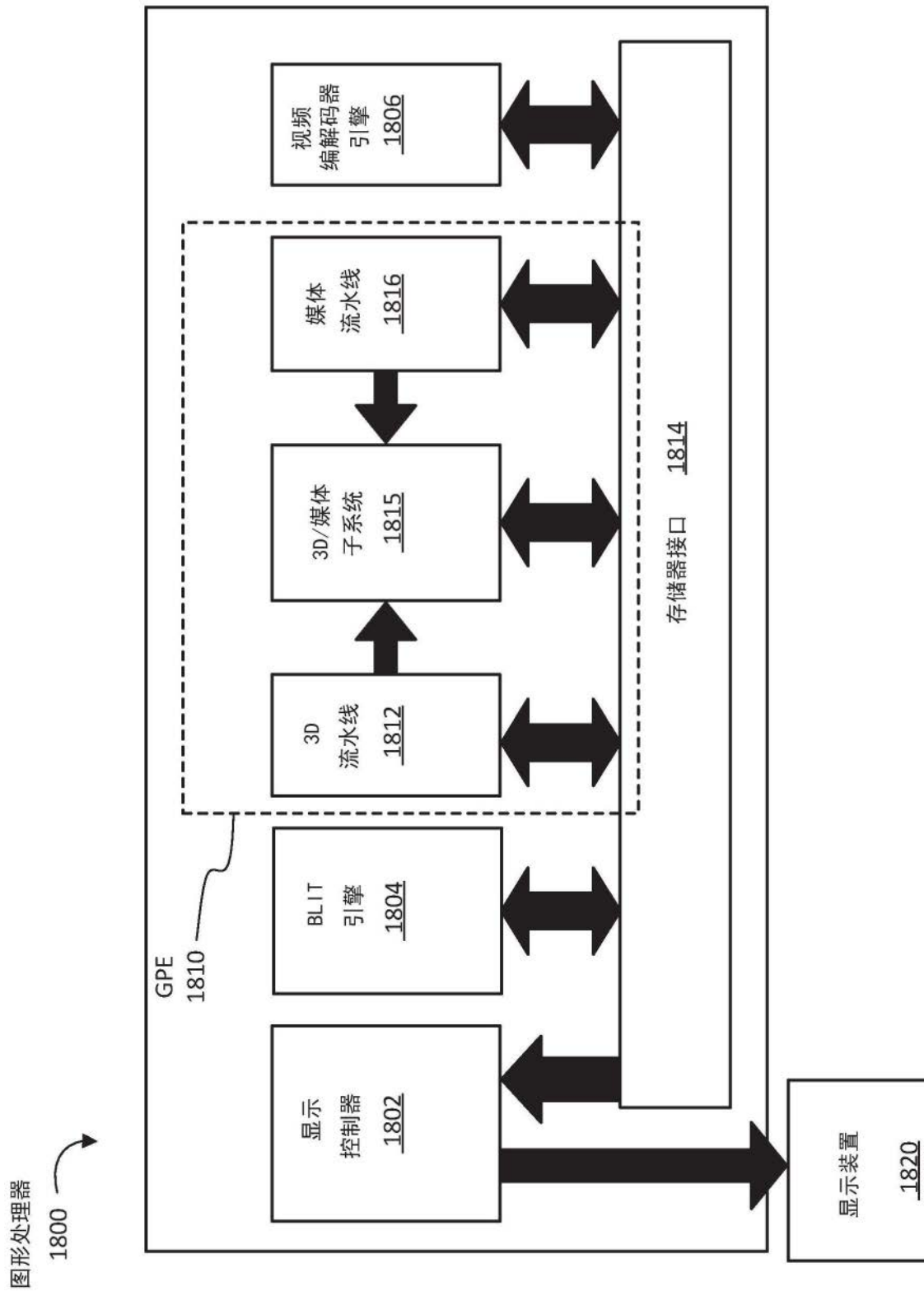


图18

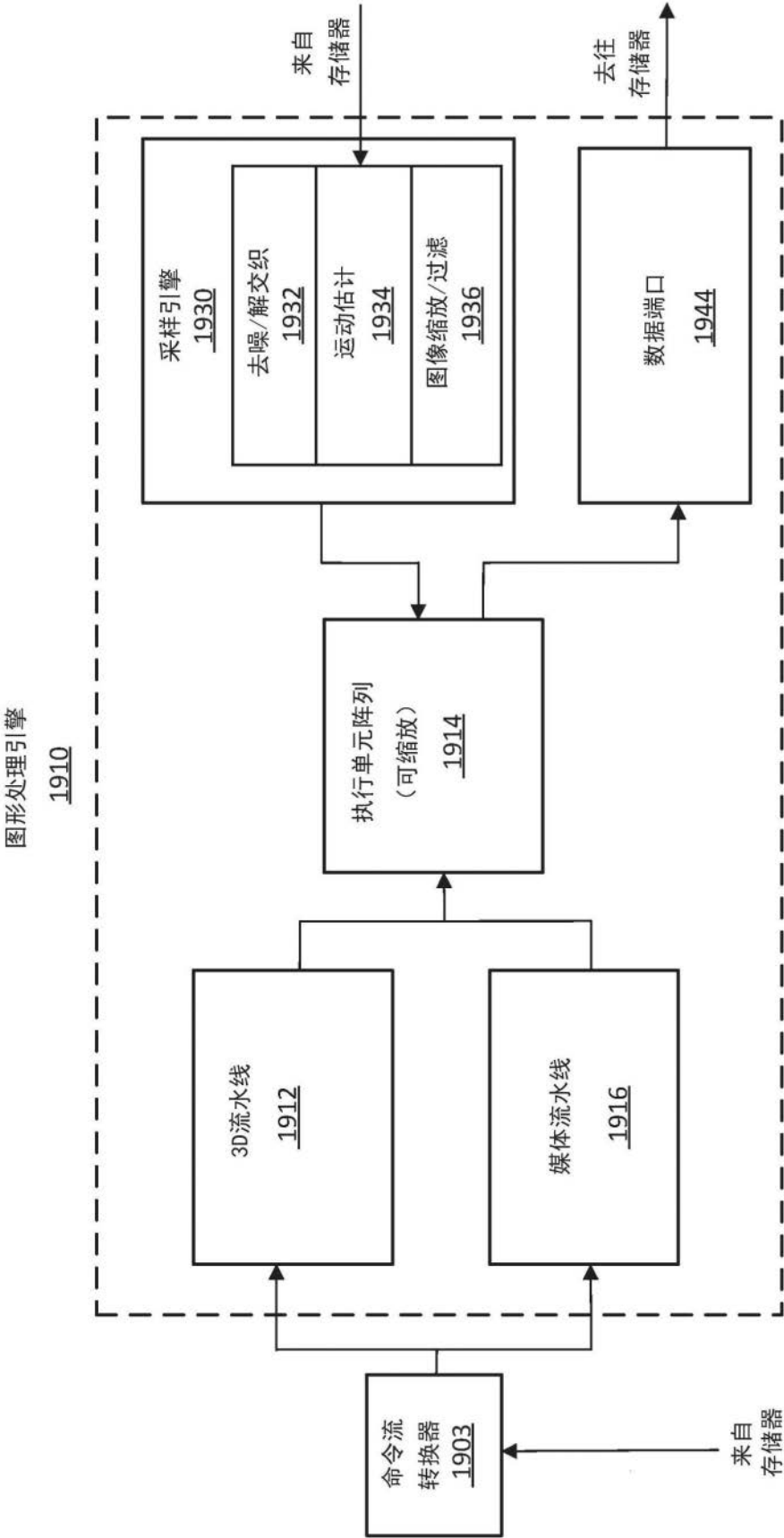


图19

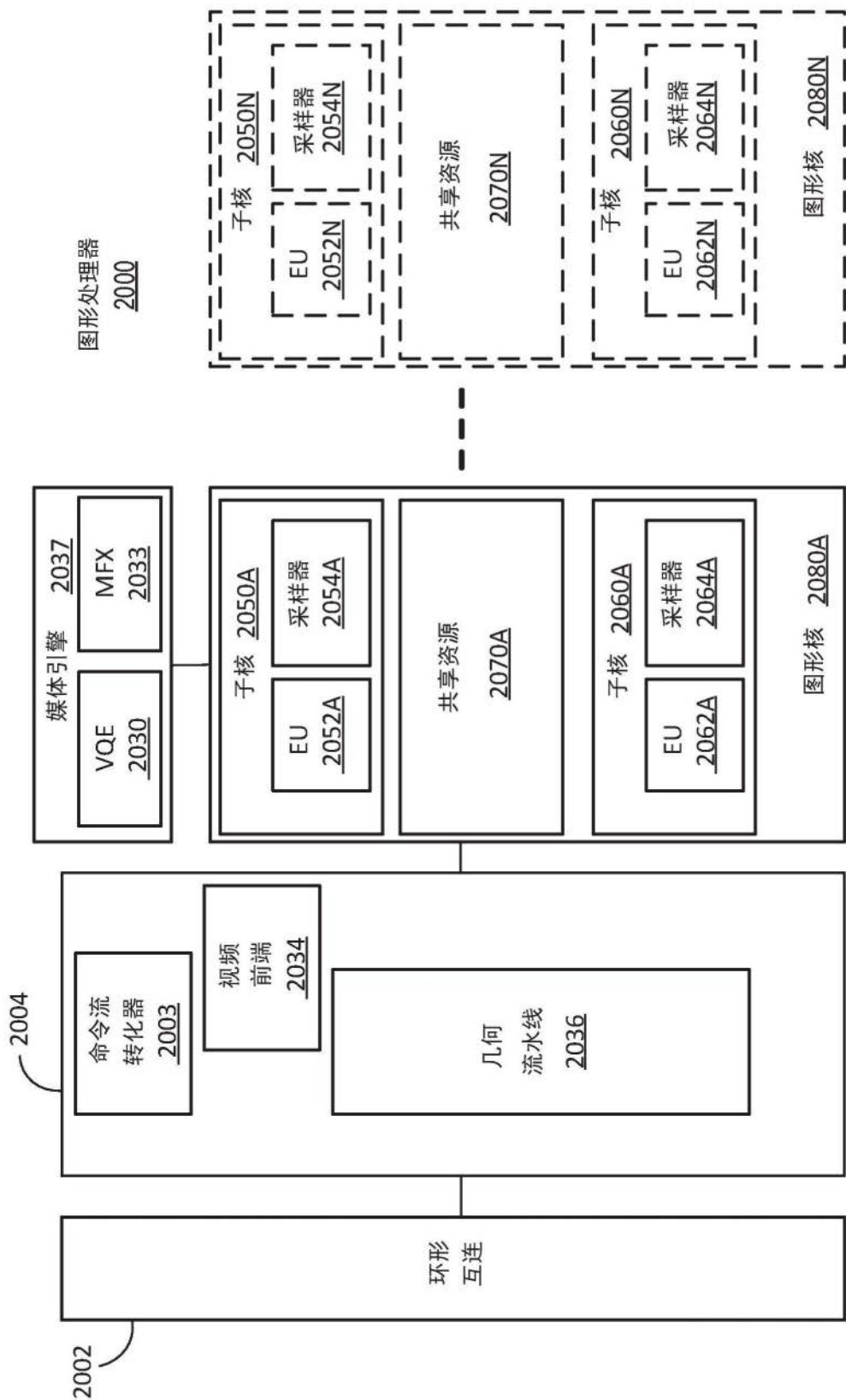


图20

2100

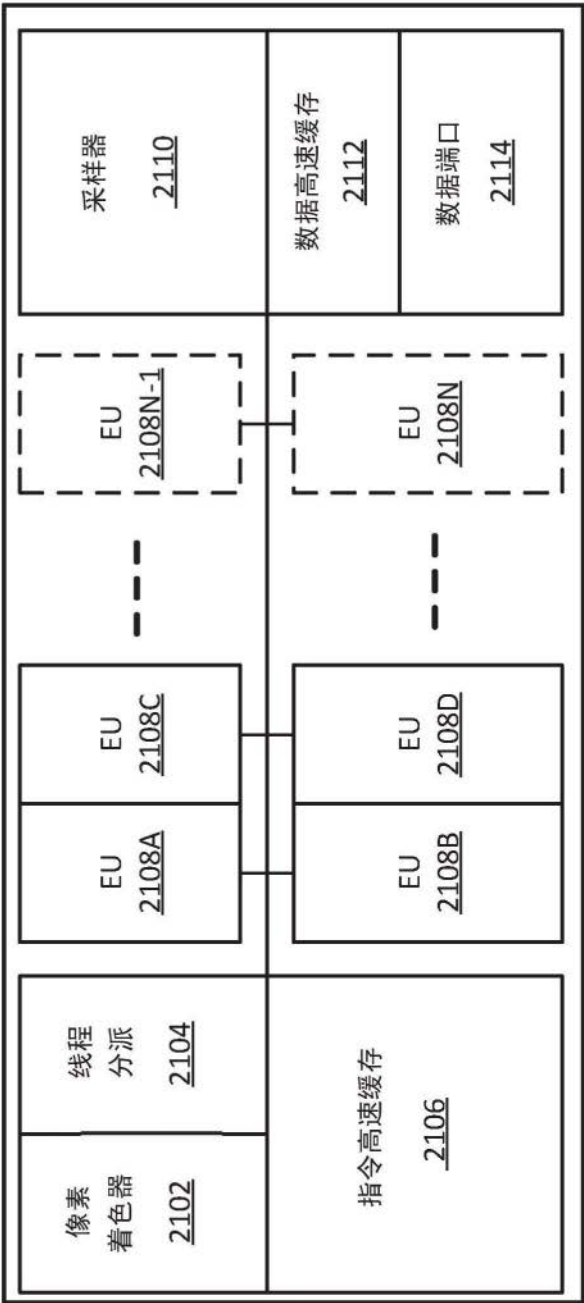


图21

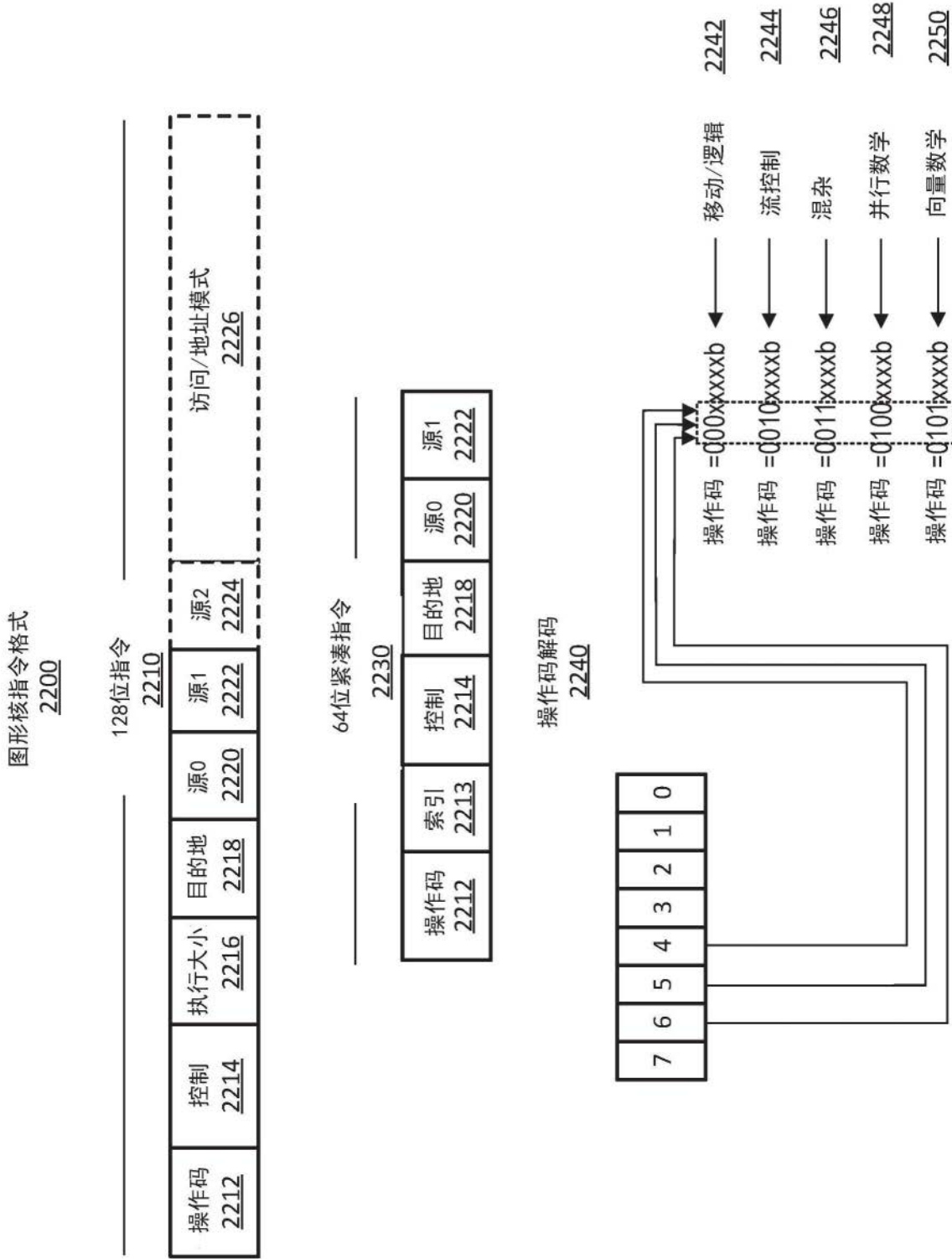


图22

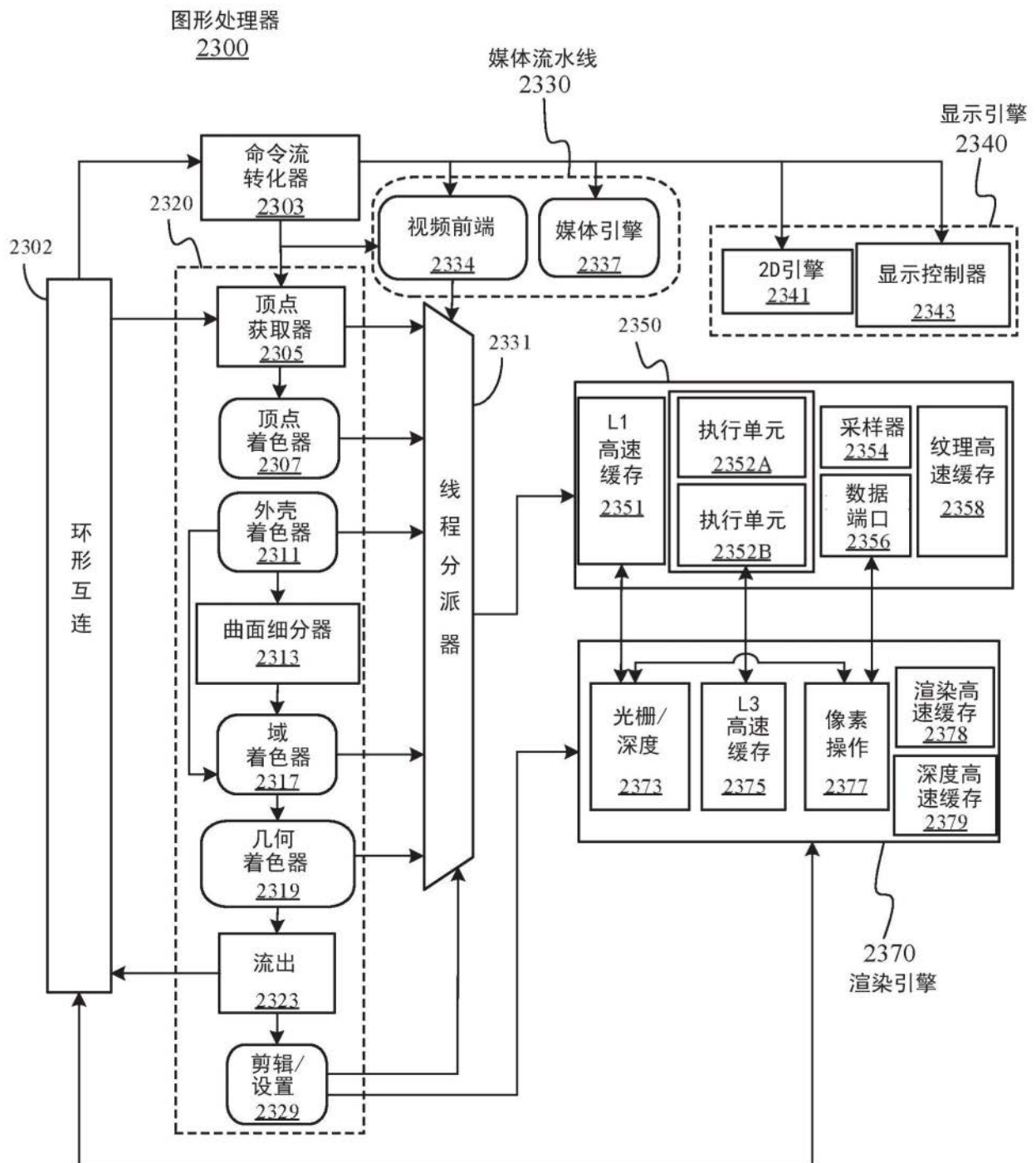


图23

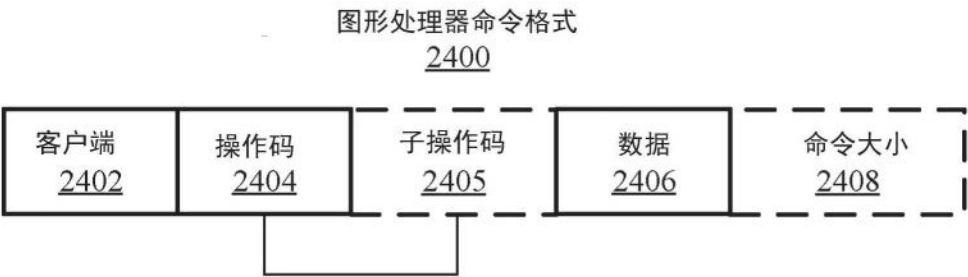


图24A

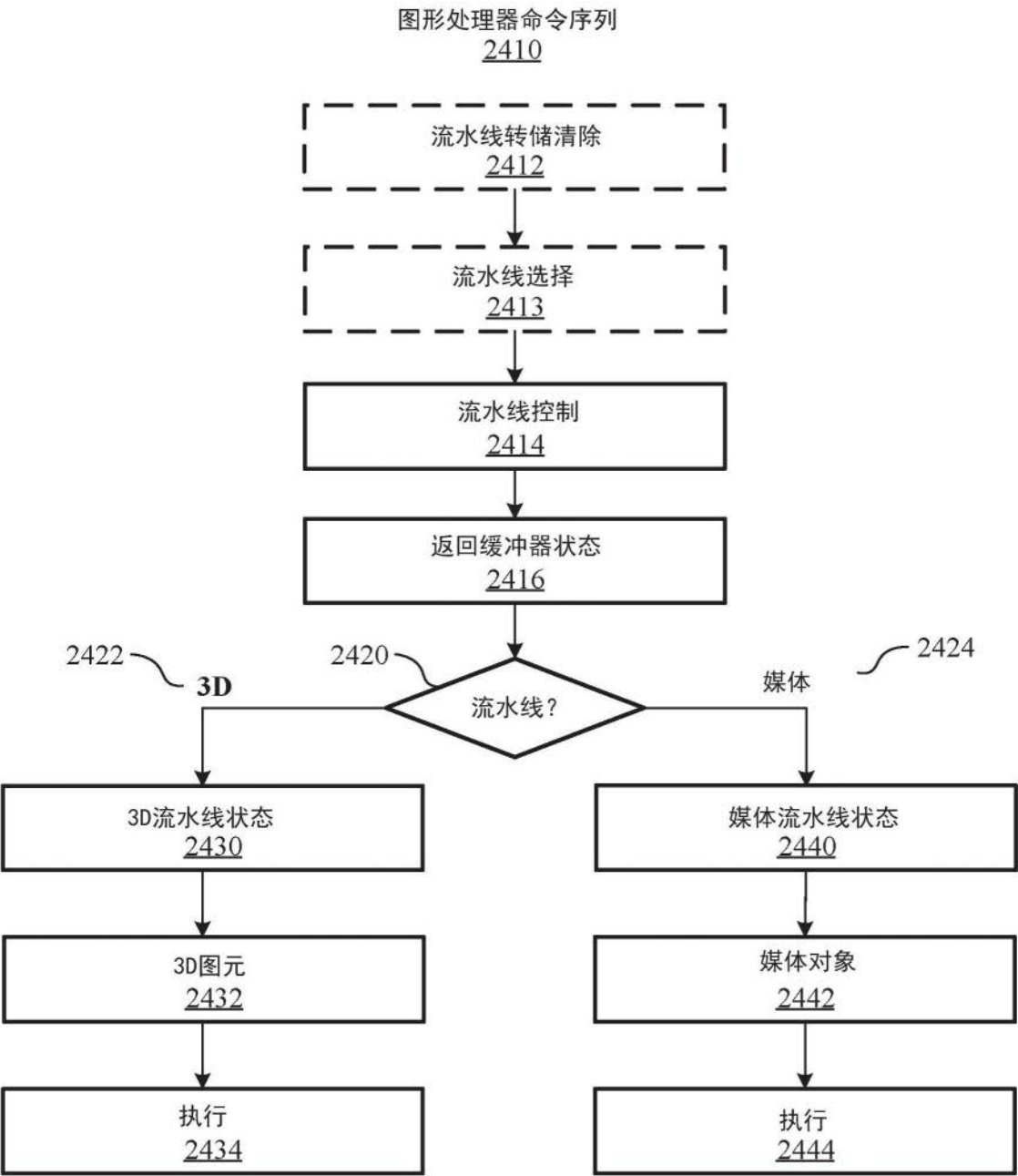


图24B

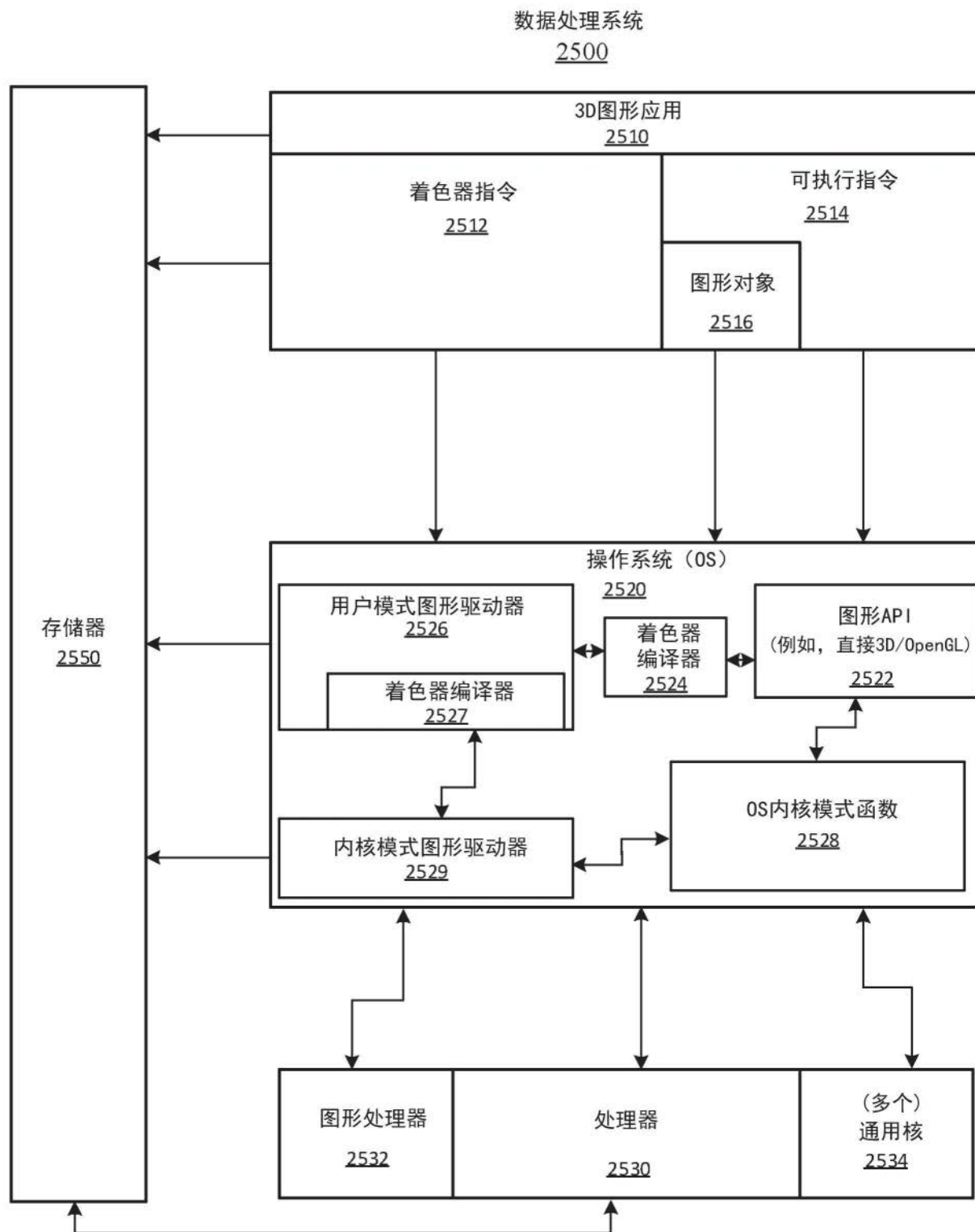


图25

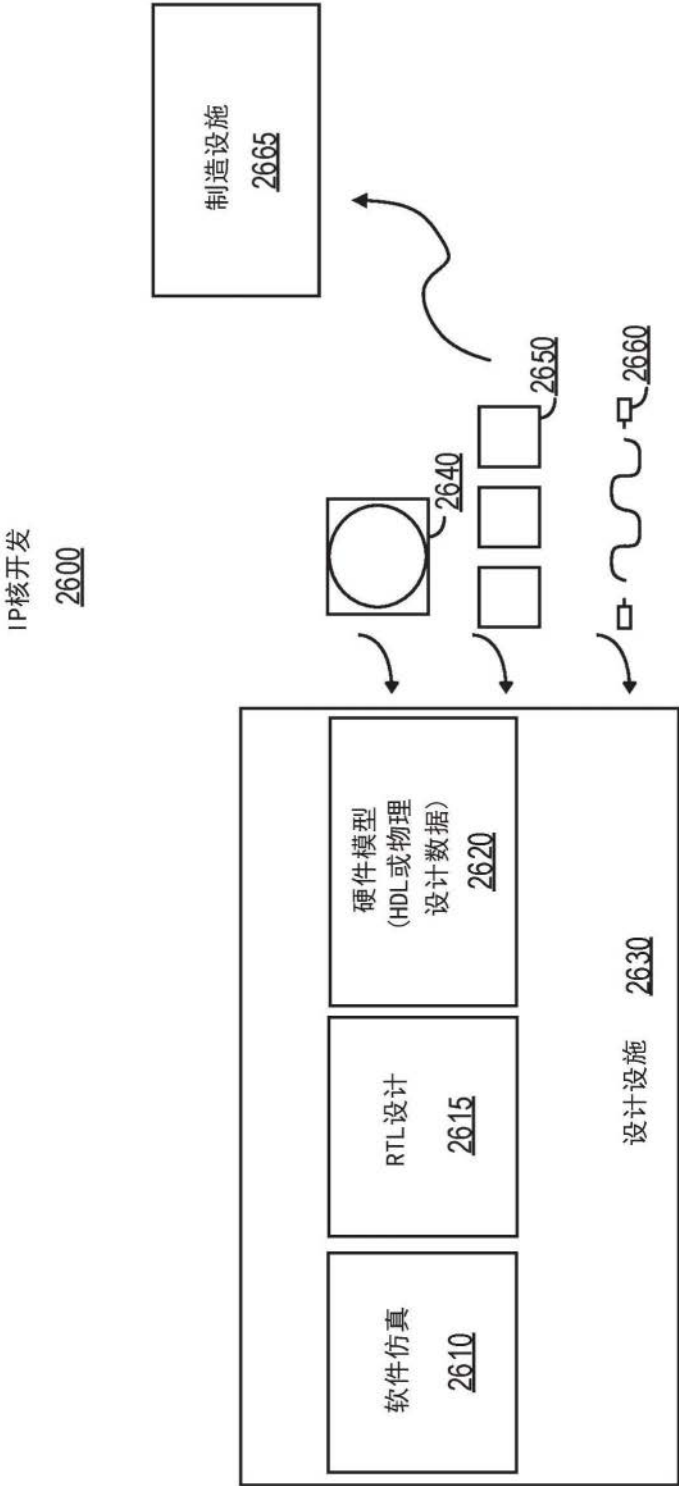


图26

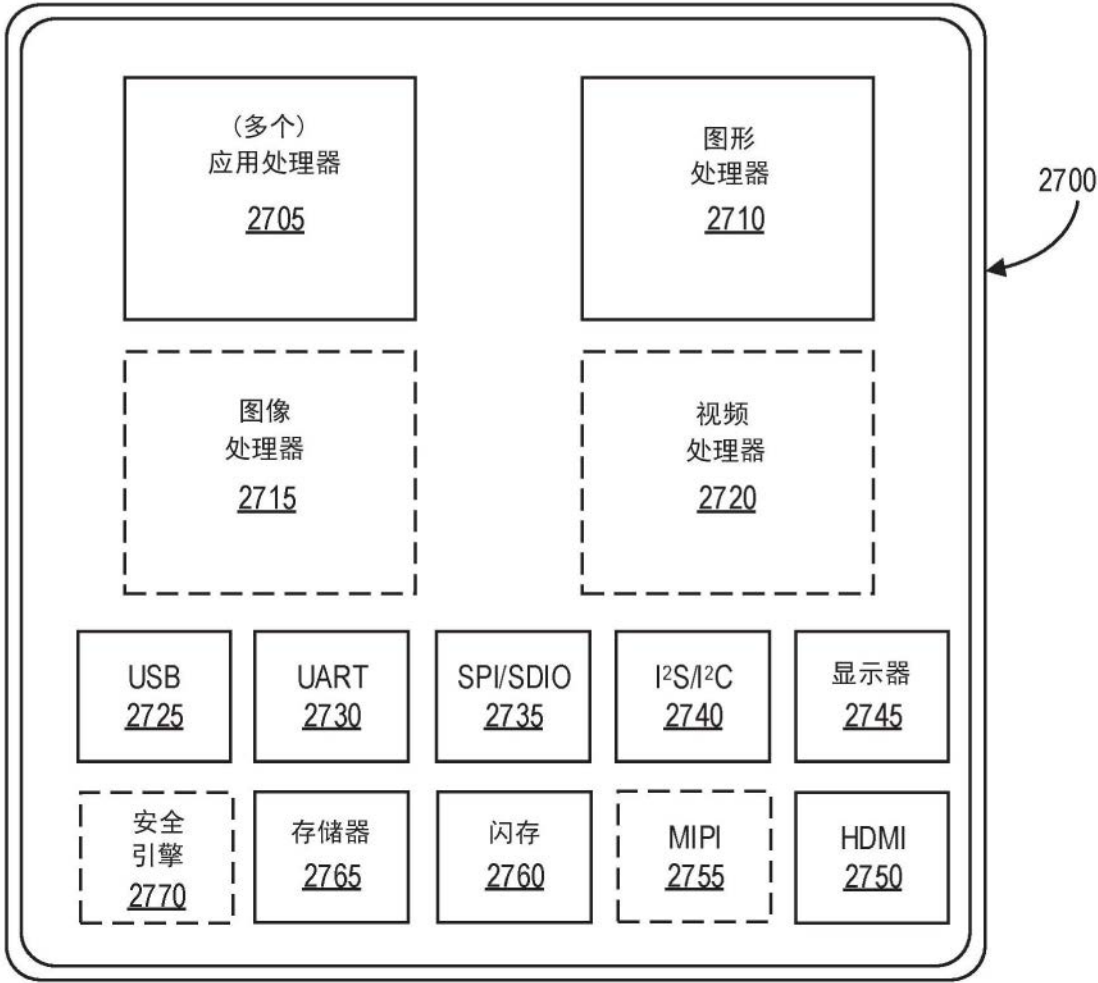


图27