



(10) 授权公告号 CN 110462602 B

(45) 授权公告日 2024. 04. 02

(21) 申请号 201780088118.6

(22) 申请日 2017.04.07

(65) 同一申请的已公布的文献号
申请公布号 CN 110462602 A

(43) 申请公布日 2019.11.15

(85) PCT国际申请进入国家阶段日
2019.09.06

(86) PCT国际申请的申请数据
PCT/CN2017/079726 2017.04.07

(87) PCT国际申请的公布数据
W02018/184208 EN 2018.10.11

(73) 专利权人 英特尔公司
地址 美国加利福尼亚州

(72) 发明人 杨柳 姚安邦

(74) 专利代理机构 中国专利代理(香港)有限公司 72001

专利代理师 郑瑾彤 申屠伟进

(51) Int.Cl.
G06F 15/76 (2006.01)

(56) 对比文件

CN 101339523 A, 2009.01.07

CN 102138138 A, 2011.07.27

CN 102681902 A, 2012.09.19

US 2010042809 A1, 2010.02.18

US 2012304194 A1, 2012.11.29

US 2013332939 A1, 2013.12.12

US 2016103901 A1, 2016.04.14

US 2016125316 A1, 2016.05.05

US 2016155049 A1, 2016.06.02

US 2016267397 A1, 2016.09.15

审查员 张钰柔

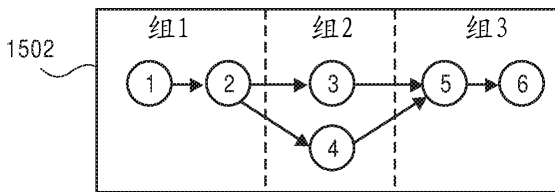
权利要求书2页 说明书45页 附图40页

(54) 发明名称

用于多处理器平台上的深度学习网络执行流水线的方法和装置

(57) 摘要

公开了在多处理器平台上使用执行流水线用于深度学习网络执行的方法和系统。在一个示例中,网络工作负荷分析器接收工作负荷、分析工作负荷的计算分发、并将网络节点分组成多组。网络执行器将每个组指派给多核平台的处理核,使得相应的处理核处理相应组的接收到的工作负荷的计算任务。



1. 一种用于在多核平台上运行具有多个网络节点的深度学习网络的流水线框架,所述流水线框架包括:

网络工作负荷分析器,其要接收工作负荷、要分析所述工作负荷的计算分发、并且要将所述网络节点分组成多组;以及

网络执行器,其要将每个组指派给所述多核平台的处理核,使得相应的处理核处理针对相应组的接收到的工作负荷的计算任务。

2. 根据权利要求1所述的框架,其中,所述网络工作负荷分析器将所述网络分组成多个组,所述组的数量等于所述多核平台的核的数量。

3. 根据权利要求1或2所述的框架,其中,所述网络工作负荷分析器基于每个组的计算量来对所述网络节点进行分组。

4. 根据权利要求3所述的框架,其中,所述网络工作负荷分析器向每个组分发类似量的工作负荷。

5. 根据权利要求1或2所述的框架,其中,所述深度学习网络被配置为非循环图,并且通过每个网络节点的数据流在单个方向上流过所述图,并且其中,通过每个组的数据流针对每个组在相同方向上。

6. 根据权利要求1或2所述的框架,其中,所述工作负荷包括连续图像的序列,并且其中,每个图像被指派给多个组。

7. 一种多处理器计算方法,包括:

接收工作负荷以供流水线框架在具有多个网络节点的深度学习网络上运行;

分析所述工作负荷跨所述网络节点的计算分发;

基于所述计算分发将所述网络节点分组成多组;

将每个组指派给所述流水线框架的相应处理资源;以及

基于所述指派在所述处理资源上执行所述工作负荷。

8. 根据权利要求7所述的方法,还包括将所述执行的输出提供给另外的过程。

9. 根据权利要求7或8所述的方法,其中,所述处理资源包括多核处理器的核。

10. 根据权利要求9所述的方法,其中,每个相应的处理核处理相应的网络节点组的接收到的工作负荷的计算任务。

11. 根据权利要求7或8所述的方法,其中,对所述网络节点进行分组包括对所述网络节点进行分组,以便跨所述组均匀地分发所述工作负荷。

12. 一种深度学习系统,包括:

传感器阵列,其要捕获图像序列;

输入缓冲器,其要接收所述图像序列;

具有多个核的网络执行器,其要在所述多个核上执行深度学习网络;以及

网络工作负荷分析器,其要接收所述图像序列作为工作负荷,要分析所述工作负荷的计算分发,并且要将所述网络节点分组成多组,其中所述网络执行器将每个组指派给相应的核并使用所述相应的核来执行所述组。

13. 根据权利要求12所述的深度学习系统,其中,所述网络执行器还要将共享存储器分配给所述多个核中的一个核作为输出并分配给所述相应核中的另一个核作为输入,以便顺序执行每个相应的网络节点组。

14. 根据权利要求12或13所述的深度学习系统,其中,所述网络工作负荷分析器基于每个组的计算量来对所述网络节点进行分组。

15. 根据权利要求12或13所述的深度学习系统,其中,所述深度学习网络被配置为非循环图,并且通过每个网络节点的数据流在单个方向上流过所述图,并且其中,通过每个组的数据流针对每个组在相同方向上。

16. 一种有指令存储于其上的机器可读介质,所述指令在被执行时引起计算设备实行根据权利要求7-11中的任一项所述的方法。

17. 一种用于多处理器计算的装置,包括:

用于接收工作负荷以供流水线框架在具有多个网络节点的深度学习网络上运行的部件;

用于分析所述工作负荷跨所述网络节点的计算分发的部件;

用于基于所述计算分发将所述网络节点分组成多组的部件;

用于将每个组指派给所述流水线框架的相应处理资源的部件;以及

用于基于所述指派在所述处理资源上执行所述工作负荷的部件。

18. 根据权利要求17所述的装置,还包括用于将所述执行的输出提供给另外的过程的部件。

19. 根据权利要求17或18所述的装置,其中,所述处理资源包括多核处理器的核。

20. 根据权利要求19所述的装置,其中,每个相应的处理核处理相应的网络节点组的接收到的工作负荷的计算任务。

21. 根据权利要求17或18所述的装置,其中,对所述网络节点进行分组包括对所述网络节点进行分组,以便跨所述组均匀地分发所述工作负荷。

用于多处理器平台上的深度学习网络执行流水线的方法和装置

技术领域

[0001] 实施例一般涉及数据处理并且更特别地涉及经由通用图形处理单元的数据处理。具体来说,实施例涉及用于在多处理器系统的流水线上执行深度学习网络的系统和方法。

背景技术

[0002] 当前的并行图形数据处理包括被开发以对图形数据执行特定操作的系统和方法,这些特定操作诸如例如线性内插、镶嵌、栅格化、纹理映射、深度测试等。传统上,图形处理器使用了固定功能计算单元来处理图形数据;然而,最近,图形处理器的多个部分已变得可编程,从而使得这样的处理器能够支持用于处理顶点和片段数据的更广泛种类的操作。

[0003] 为了进一步提高性能,图形处理器通常实现诸如流水线化之类的处理技术,这些处理技术试图贯穿图形流水线的不同部分来并行处理尽可能多的图形数据。具有单指令多线程(SIMT)架构的并行图形处理器被设计成最大化图形流水线中的并行处理量。在SIMT架构中,多组并行线程试图尽可能经常地一起同步执行程序指令,以提高处理效率。用于SIMT架构的软件和硬件的一般性概述可以在Shane Cook的CUDA编程(CUDA Programming),第3章,第37-51页(2013年)中找到。

[0004] 机器学习已成功解决了多种任务。在训练和使用机器学习算法(例如,神经网络)时出现的计算自然适合于高效的并行实现。因此,诸如通用图形处理单元(GPGPU)之类的并行处理器已在深度神经网络的实际实现中发挥了重要作用。具有单指令多线程(SIMT)架构的并行图形处理器被设计成最大化图形流水线中的并行处理量。在SIMT架构中,多组并行线程试图尽可能经常地一起同步执行程序指令,以提高处理效率。并行机器学习算法实现所提供的效率允许使用高容量网络,并且使得能够在更大的数据集上训练那些网络。

[0005] 深度学习神经网络(DNN)通常被构造为一种卷积神经网络,并且用于执行复杂的关联性任务。在使用已知输入的训练阶段之后,DNN能够识别与原始训练输入类似的新输入。这有助于对象检测技术、自动语音识别、用户认证、图像理解和机器视觉用途等。视频序列可以用于对象跟踪以及识别。

[0006] DNN对系统提出了大量的计算处理需求,但这些任务中的许多都是重复的。利用多核流水线,可以更快地处理DNN工作负荷。DNN通常被形成为在非循环图中具有多个节点的网络。数据在一个方向上从一个节点流向下一个节点,进行适当的分裂和结合,穿过该图。

附图说明

[0007] 附图图示了示例,并且因此是示例性实施例,而不被认为是对范围的限制。

[0008] 图1是图示了被配置成实现本文所述的示例性实施例的一个或多个方面的计算机系统的框图。

[0009] 图2A-2D图示了根据示例性实施例的并行处理器部件。

[0010] 图3A-3B是根据示例性实施例的图形多处理器的框图。

[0011] 图4A-4F图示了其中多个图形处理单元 (GPU) 通信地耦合至多个多核处理器的示例性架构。

[0012] 图5图示了根据示例性实施例的图形处理流水线。

[0013] 图6图示了根据示例性实施例的机器学习软件堆栈。

[0014] 图7图示了根据示例性实施例的高度并行的通用图形处理单元。

[0015] 图8图示了根据示例性实施例的多GPU计算系统。

[0016] 图9A-9B图示了示例性深度神经网络的各层。

[0017] 图10图示了示例性递归神经网络。

[0018] 图11图示了深度神经网络的训练和部署的示例性实施例。

[0019] 图12是图示了分布式学习的示例性框图。

[0020] 图13图示了适合于通过使用已训练的模型来执行推断的示例性进行推断的片上系统 (SOC)。

[0021] 图14是将RGB和CNN通道数据存储为深度通道图像的一部分的示例性图示,其是使用卷积神经网络 (CNN) 存储深度通道图像的图像捕获系统的示例性框图。

[0022] 图15是深度学习网络的有序节点的图示。

[0023] 图16是对图15的节点进行分组的图示。

[0024] 图17是被配置成执行深度学习网络的多核处理平台的图示。

[0025] 图18是多核处理平台上的深度学习网络执行的过程流程图。

[0026] 图19图示了根据示例性实施例的处理系统的框图。

[0027] 图20图示了具有一个或多个处理器核、集成存储器控制器和集成图形处理器的处理器的实施例的示例性框图。

[0028] 图21图示了图形处理器的示例性框图。

[0029] 图22图示了根据示例性实施例的图形处理器的图形处理引擎的框图。

[0030] 图23图示了图形处理器的另一示例性实施例的框图。

[0031] 图24图示了包括在图形处理引擎 (GPE) 的示例性实施例中采用的处理元件阵列的线程执行逻辑。

[0032] 图25图示了根据示例性实施例的图形处理器指令格式的框图。

[0033] 图26图示了图形处理器的示例性实施例的框图。

[0034] 图27A图示了根据示例性实施例的图形处理器命令格式的框图。

[0035] 图27B图示了根据示例性实施例的图形处理器命令序列的框图。

[0036] 图28图示了根据示例性实施例的数据处理系统的示例性图形软件架构。

[0037] 图29图示了根据示例性实施例的IP核开发系统的框图,该系统可以用于制造集成电路 (IC) 以执行操作。

[0038] 图30图示了根据示例性实施例的示例性片上系统IC的框图,该IC可以使用一个或多个IP核来制造。

[0039] 图31图示了根据示例性实施例的在片上系统IC上的示例性图形处理器的框图,其可以使用一个或多个IP核来制造。

[0040] 图32图示了根据示例性实施例的片上系统IC的示例性附加图形处理器的框图,其可以使用一个或多个IP核来制造。

具体实施方式

[0041] 在一些实施例中,图形处理单元(GPU)通信地耦合到主机/处理器核以加速图形操作、机器学习操作、模式分析操作、以及各种通用GPU(GPGPU)功能。GPU可以通过总线或另一互连(例如,诸如PCIe或NVLink之类的高速互连)通信地耦合到主机处理器/核。在其他实施例中,GPU可以与核一样集成在相同封装或芯片上,并通过内部处理器总线/互连(即,在封装或芯片内部)通信地耦合到核。不管GPU连接的方式如何,处理器核都可以以工作描述符中所包含的命令/指令的序列的形式将工作分配给GPU。GPU随后使用专用电路/逻辑以用于高效地处理这些命令/指令。

[0042] 在一些实施例中,图像捕获设备是独立的设备。然而,图像捕获设备可以是需要图像捕获能力的另一计算设备的一部分或子部件,该另一计算设备诸如便携式或手持式计算设备,其具有数字相机来捕获图像。

[0043] 在以下描述中,阐述了很多特定细节以提供更透彻的理解。然而显而易见的是,没有这些特定细节中的一个或多个也可以实践本文中所描述的实施例。在其他实例中,未描述公知的特征以避免使示例性实施例的细节变得模糊。

[0044] 计算系统概述

[0045] 图1是图示了被配置成实现本文所述的示例性实施例的一个或多个方面的计算系统100的框图。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,所述一个或多个处理器和所述系统存储器经由互连路径进行通信,所述互连路径可以包括存储器中枢105。存储器中枢105可以是芯片组部件内的单独的部件,或者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设备108接收输入。另外,I/O中枢107可以使得显示控制器(所述显示控制器可以被包括在一个或多个处理器102中)能够向一个或多个显示设备110A提供输出。在一个实施例中,与I/O中枢107耦合的一个或多个显示设备110A可以包括本地显示设备、内部显示设备或嵌入式显示设备。

[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,所述一个或多个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可以是任意数量的基于标准的通信链路技术或协议(诸如但不限于PCI Express)中的一个,或者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形成以计算为中心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,诸如集成众核(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向经由I/O中枢107耦合的一个或多个显示设备110A中的一个输出像素。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)以实现到一个或多个显示设备110B的直接连接。

[0047] 在I/O子系统111内,系统存储单元114可以连接至I/O中枢107来为计算系统100提供存储机制。I/O开关116可以用于提供接口机制以实现I/O中枢107和可以集成到平台中的其他部件(诸如网络适配器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设备120添加的各种其他设备之间的连接。网络适配器118可以是以太网适配器或另一种有线网络适配器。无线网络适配器119可以包括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设备中的一个或多个。

[0048] 计算系统100可以包括未显式示出的其他部件,这些部件包括USB或其他端口连接件、光存储驱动器、视频捕获设备等,也可以连接至I/O中枢107。图1中将各种部件互连的通信路径可以使用任何合适的协议(诸如基于PCI(外围部件互连)的协议(例如,PCI-Express))或(多个)任何其他总线或点对点通信接口和/或协议(诸如NV-Link高速互连或本领域中已知的互连协议)来实现。

[0049] 在一个实施例中,一个或多个并行处理器112并入有为进行图形和视频处理而优化的电路,包括例如视频输出电路,并且所述电路构成图形处理单元(GPU)。在另一实施例中,一个或多个并行处理器112并入有为进行通用处理而优化的电路,同时保留了本文更详细描述的基础计算架构。在又一个实施例中,计算系统100的各部件可以与一个或多个其他系统元件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、(多个)处理器102和I/O中枢107可以集成到片上系统(SoC)集成电路中。可替代地,计算系统100的各部件可以集成到单个封装中以形成系统级封装(SIP)配置。在一个实施例中,计算系统100的各部件的至少一部分可以集成到多芯片模块(MCM)中,所述多芯片模块可以与其他多芯片模块互连成模块化计算系统。

[0050] 应当理解,本文所示的计算系统100是例示性的并且变型和修改是可能的。连接拓扑可以根据需要进行修改,所述连接拓扑包括桥的数量和安排、(多个)处理器102的数量和(多个)并行处理器112的数量。例如,在一些实施例中,系统存储器104直接而不是通过桥连接至(多个)处理器102,而其他设备经由存储器中枢105和(多个)处理器102与系统存储器104进行通信。在其他替代性拓扑中,(多个)并行处理器112连接至I/O中枢107或直接连接至一个或多个处理器102中的一个,而不是连接至存储器中枢105。在其他实施例中,I/O中枢107和存储器中枢105可以集成到单个芯片中。一些实施例可以包括经由多个插座附接的(多个)处理器102的两个或更多个组,这两个或更多个组可以与(多个)并行处理器112的两个或更多个实例耦合。

[0051] 本文示出的一些特定部件是可选的并且可能不被包括在计算系统100的所有实现中。例如,可以支持任意数量的插入式卡或外围装置,或者可以省去一些部件。此外,一些架构可以使用针对与图1中所图示的那些类似的部件的不同术语。例如,存储器中枢105在一些架构中可以被称为北桥,而I/O中枢107可以被称为南桥。

[0052] 图2A图示了根据示例性实施例的并行处理器200。并行处理器200的各种部件可以使用诸如可编程处理器、专用集成电路(ASIC)或现场可编程门阵列(FPGA)之类的一个或多个集成电路设备来实现。根据示例性实施例,所图示的并行处理器200是图1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单元实现与包括并行处理单元202的其他实例的其他设备的通信。I/O单元204可以直接连接至其他设备。在一个实施例中,I/O单元204经由诸如存储器中枢105之类的中枢或开关接口的使用来与其他设备连接。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引导至前端208。在一个实施例中,前端208与调度器210耦合,所述调度器被配置成将命令或其他工作项目分发至处理集群阵列212。在一个实施例中,调度器210确保处理集群阵列212被正确配置,并且在将任务分发至处理集群阵列212的处理集群之前处于有效状态。在一个实施例中,经由微控制器上执行的固件逻辑来实现调度器210。微控制器实现的调度器210可配置成以粗和细粒度来执行复杂的调度和工作分发操作,从而实现在处理阵列212上执行的线程的迅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形处理门铃之一而证明针对处理阵列212上调度的工作负荷。然后可以通过调度器微控制器内的调度器210逻辑而跨处理阵列212自动分发工作负荷。

[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、一直到集群214N)。处理集群阵列212的每个集群214A至214N均可执行大量并发线程。调度器210可以使用各种调度和/或工作分发算法来向处理集群阵列212的集群214A至214N分配工作,这些算法可以依据每种类型的程序或计算引起的工作负荷而变化。调度可以由调度器210动态地处理,或者可以在编译被配置成由处理集群阵列212执行的程序逻辑的过程中由编译器逻辑部分地协助。在一个实施例中,处理集群阵列212的不同集群214A至214N可以被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] 处理集群阵列212可以被配置成执行各种类型的并行处理操作。在一个实施例中,处理集群阵列212被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,该处理任务包括:视频和/或音频数据的过滤;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可以包括用于支持此类图形处理操作的执行的附加逻辑,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相关的着色器程序,诸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从系统存储器传送数据以进行处理。在处理期间,可以在处理期间将经传送的数据存储到片上存储器(例如,并行处理器存储器222),然后写回到系统存储器。

[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分成大致相等大小的任务,以更好地使得图形处理操作能够分发到处理集群阵列212的多个集群214A至214N。在一些实施例中,处理集群阵列212的各部分可以被配置成执行不同类型的处理。例如,第一部分可以被配置成执行顶点着色和拓扑生成,第二部分可以被配置成执行曲面细分和几何着色,并且第三部分可以被配置成执行像素着色或其他屏幕空间操作,以产生渲染的图像进行显示。由集群214A至214N中的一个或多个产生的中间数据可以存储在缓冲器中以允许中间数据在集群214A至214N之间传输以用于进一步处理。

[0059] 在操作期间,处理集群阵列212可以接收将经由调度器210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。对于图形处理操作,处理任务可以包括要处理的数据(例如表面(补丁)数据、图元数据、顶点数据和/或像素数据以及定义如何处理数据

的状态参数和命令(例如,要执行哪个程序))的索引。调度器210可以被配置成获取对应于任务的索引或者可以从前端208接收索引。前端208可以被配置成确保处理集群阵列212在由传入命令缓冲器(例如,批处理缓冲器、入栈缓冲器等)指定的工作负荷被发起之前被配置成有效状态。

[0060] 并行处理单元202的一个或多个实例中的每一个均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,所述存储器交叉开关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、一直到分区单元220N),这些分区单元可以各自耦合至并行处理器存储器222的一部分(例如,存储器单元)。在一个实现中,分区单元220A至220N的数量被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,以及第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至220N的数量可能不等于存储器设备的数量。

[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设备,包括动态随机存取存储器(DRAM)或图形随机存取存储器,诸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还可以包括3D堆叠式存储器,包括但不限于高带宽存储器(HBM)。本领域技术人员将会理解,存储器单元224A至224N的具体实现可以变化,并且可以从各种常规设计之一进行选择。诸如帧缓冲器或纹理映射之类的渲染目标可以存储在存储器单元224A至224N上,从而允许分区单元220A至220N并行地写入每个渲染目标的各部分,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,为了支持利用系统存储器连同本地高速缓存存储器的统一存储器设计,可以将并行处理器存储器222的本地实例排除在外。

[0062] 在一个实施例中,处理集群阵列212的集群214A至214N中的任一个可以处理将被写入到并行处理器存储器222内的存储器单元224A至224N中的任一个的数据。存储器交叉开关216可以被配置成将每个集群214A至214N的输出传送到任何分区单元220A至220N或另一个集群214A至214N,这可以对所述输出执行附加处理操作。每个集群214A至214N均可以通过存储器交叉开关216与存储器接口218进行通信以针对各种外部存储器设备进行读取或写入操作。在一个实施例中,存储器交叉开关216可连接至存储器接口218以与I/O单元204通信,并且可连接至并行处理器存储器222的本地实例,从而使得不同处理集群214A至214N内的处理单元能够与系统存储器或对于并行处理单元202并非本地的其他存储器进行通信。在一个实施例中,存储器交叉开关216可以使用虚拟通道来分离集群214A至214N与分区单元220A至220N之间的业务流。

[0063] 虽然并行处理单元202的单个实例图示为在并行处理器200内,但并行处理单元202的任意数量的实例也可以被包括在内。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,或者可以使多个插入式卡互连。即使不同实例具有不同的处理核数量、不同的本地并行处理器存储量和/或其他配置差异,并行处理单元202的不同实例也可以被配置成交互操作。例如,以及在一个实施例中,并行处理单元202的一些实例可以包括相对于其他实例的较高精度的浮点单元。并入有并行处理单元202或并行处理器200的一个或多个实例的系统可以以各种配置和形状因数来实现,包括但不限于台式计算机、膝上型计算机

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是根据示例性实施例的分区单元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中的一个的实例。如所示出的,分区单元220包括L2高速缓存221、帧缓冲器接口225和ROP 226(栅格操作单元)。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向帧缓冲器发送更新以用于处理。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(诸如,图2的存储器单元224A至224N(例如,在并行处理器存储器222内))中的一个接口连接。

[0065] 在图形应用中,ROP 226是执行诸如模板印刷(stencil)、z测试、混合等等之类的栅格操作的处理单元。ROP 226随后输出经处理的图形数据,该图形数据被存储在图形存储器中。在一些实施例中,ROP 226包括压缩逻辑,所述压缩逻辑用于压缩被写入到存储器的深度或颜色数据,并对从存储器所读取的深度或颜色数据解压缩。压缩逻辑可以是利用多种压缩算法中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将被压缩的数据的统计特性而变化。例如,在一个实施例中,德尔塔(delta)颜色压缩在每图块的基础上在深度和颜色数据上被执行。

[0066] 在一些实施例中,ROP 226被包括在每个处理集群(例如,图2A的集群214A至214N)内而非被包括在分区单元220内。在这样的实施例中,通过存储器交叉开关216而非像素片段数据来传输针对像素数据的读取和写入请求。经处理图形数据可以显示在显示设备(诸如图1的一个或多个显示设备110A、110B中的一个)上,由(多个)处理器102路由以用于进一步处理,或者由图2A的并行处理器200内的处理实体中的一个路由以用于进一步处理。

[0067] 图2C是根据示例性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2A的处理集群214A至214N中的一个的实例。处理集群214可以被配置成并行地执行许多线程,其中术语“线程”是指在特定输入数据集上执行的特定程序的实例。在一些实施例中,使用单指令多数据(SIMD)指令发布技术来支持大量线程的并行执行,而无需提供多个独立的指令单元。在其他实施例中,使用单指令多线程(SIMT)技术来使用被配置成向处理集群中的每一个内的一组处理引擎发布指令的公共指令单元来支持大量大致同步线程的并行执行。与所有处理引擎通常执行相同指令的SIMD执行机制不同,SIMT执行允许不同线程更容易地遵循穿过给定线程程序的发散执行路径。本领域技术人员将会理解,SIMD处理机制表示SIMT处理机制的功能子集。

[0068] 处理集群214的操作可以经由向SIMT并行处理器分发处理任务的流水线管理器232来控制。流水线管理器232从图2A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来管理那些指令的执行。所图示的图形多处理器234是SIMT并行处理器的示例性实例。然而,不同架构的各种类型的SIMT并行处理器可以被包括在处理集群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可以用于将经处理数据分发到包括其他着色器单元的多个可能目的地中的一个。流水线管理器232可以通过为将经由数据交叉开关240分发的经处理数据指定目的地来促进经处理数据的分发。

[0069] 处理集群214内的每个图形多处理器234均可以包括相同的功能执行逻辑组(例

如,算术逻辑单元、加载存储单元等)。功能执行逻辑可以通过流水线化的方式进行配置,其中可以在先前的指令完成之前发布新的指令。功能执行逻辑支持各种运算,包括整数和浮点算术、比较运算、布尔运算、位移位和各种代数函数的计算。在一个实施例中,可以利用相同的功能单元硬件来执行不同的操作,并且可以存在功能单元的任意组合。

[0070] 传输到处理集群214的指令构成线程。在一组并行处理引擎上执行的一组线程是线程组。线程组在不同的输入数据上执行相同的程序。线程组内的每个线程均可以被指派给图形多处理器234内的不同处理引擎。线程组可以包括比图形多处理器234内的处理引擎的数量更少的线程。当线程组包括比处理引擎的数量更少的线程时,处理引擎中的一个或多个处理引擎可能在处理线程组的周期期间空闲。线程组还可以包括比图形多处理器234内的处理引擎的数量更多的线程。当线程组包括比图形多处理器234内的处理引擎的数量更多的线程时,可以在连续的时钟周期上执行处理。在一个实施例中,可以在图形多处理器234上同时执行多个线程组。

[0071] 在一个实施例中,图形多处理器234包括用于执行加载和存储操作的内部高速缓存存储器。在一个实施例中,图形多处理器234可以放弃内部高速缓存而在处理集群214内使用高速缓存存储器(例如,L1高速缓存248)。每个图形多处理器234还可以访问在所有处理集群214之间共享的分区单元(例如,图2A的分区单元220A至220N)内的L2高速缓存,并且可以用于在线程之间传送数据。图形多处理器234还可以访问片外全局存储器,所述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个或多个。并行处理单元202外部的任何存储器可以用作全局存储器。其中处理集群214包括图形多处理器234的多个实例的实施例可以共享可以在L1高速缓存248中存储的公共指令和数据。

[0072] 每个处理集群214均可以包括被配置成将虚拟地址映射到物理地址的MMU 245(存储器管理单元)。在其他实施例中,MMU 245的一个或多个实例可以驻留在图2A的存储器接口218内。MMU 245包括用于将虚拟地址映射到图块(tile)的物理地址(更多地提及分块)和可选地高速缓存行索引的一组页表项(PTE)。MMU 245可以包括可以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。对物理地址进行处理以分发表面数据访问局部性以实现分区单元之间的高效请求交错。可以使用高速缓存行索引来确定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可以被配置成使得每个图形多处理器234均耦合至纹理单元236以执行纹理映射操作,例如确定纹理样本位置、读取纹理数据和过滤纹理数据。纹理数据是从内部纹理L1高速缓存(未示出)或者在一些实施例中从图形多处理器234内的L1高速缓存读取的,并且是根据需要从L2高速缓存、本地并行处理器存储器或系统存储器获取的。每个图形多处理器234向数据交叉开关240输出经处理任务以向另一个处理集群214提供经处理任务以用于进一步处理或经由存储器交叉开关216在L2高速缓存、本地并行处理器存储器或系统存储器中存储经处理任务。preROP 242(预先栅格操作单元)被配置成从图形多处理器234接收数据,将数据引导到ROP单元,这些ROP单元可以如本文所述的那样用分区单元(例如,图2A的分区单元220A至220N)定位。preROP 242单元可以对颜色混合进行优化、组织像素颜色数据并执行地址转换。

[0074] 应当理解,本文所述的核架构是例示性的并且变型和修改是可能的。例如图形多处理器234、纹理单元236、preROP 242等任意数量的处理单元可以被包括在处理集群214

内。此外,虽然仅示出一个处理集群214,但如本文所述的并行处理单元可以包括处理集群214的任意数量的实例。在一个实施例中,每个处理集群214均可以被配置成使用单独的和不同的处理单元、L1高速缓存等来独立于其他处理集群214而操作。

[0075] 图2D示出了根据一个示例性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,所述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接收要执行的指令流。将这些指令高速缓存在指令高速缓存252中并分派用于由指令单元254执行。指令单元254可以将指令作为线程组(例如,经线)进行分派,其中线程组的每个线程均被指派给GPGPU核262内的不同执行单元。指令可以通过在统一地址空间内指定地址来访问本地、共享或全局地址空间中的任一个。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由加载/存储单元266访问的不同存储器地址。

[0077] 寄存器堆258为图形多处理器234的功能单元提供一组寄存器。寄存器堆258为连接至图形多处理器234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,寄存器堆258在功能单元中的每一个之间进行划分,使得每个功能单元均被分配寄存器堆258的专用部分。在一个实施例中,寄存器堆258在正由图形多处理器234执行的不同经线之间进行划分。

[0078] GPGPU核262可以各自包括用于执行图形多处理器234的指令的浮点单元(FPU)和/或整数算术逻辑单元(ALU)。根据实施例,GPGPU核262的架构可以类似,或者可以不同。例如,以及在一个实施例中,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可以实现IEEE 754-2008浮点算术标准或启用可变精度浮点算术。另外,图形多处理器234还可以包括用于执行诸如复制矩形或像素混合操作之类的特定功能的一个或多个固定功能或特殊功能单元。在一个实施例中,GPGPU核中的一个或多个还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SIMD逻辑,所述SIMD逻辑能够在多个数据集合上执行单个指令。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指令可以在编译时间由着色器编译器生成或者在执行为了单程序多数据(SPMD)或SIMT架构所编写和编译的程序时自动地被生成。被配置用于SIMT执行模型的程序的多个线程可以经由单个SIMD指令来被执行。例如并且在一个实施例中,执行相同或类似操作的八个SIMT线程可以经由单个SIMD8逻辑单元并行地被执行。

[0080] 存储器和高速缓存互连268是互连网络,所述互连网络将图形多处理器234的功能单元中的每一个连接至寄存器堆258和共享存储器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的交叉开关互连。寄存器堆258可以以与GPGPU核262相同的频率操作,因此GPGPU核262与寄存器堆258之间的数据传送具有非常低的等待时间。共享存储器270可以用于实现在图

形多处理器234内的功能单元上执行的线程之间的通信。例如,高速缓存存储器272可以用作数据高速缓存,以高速缓存在功能单元与纹理单元236之间通信的纹理数据。共享存储器270也可以用作经高速缓存的受管理的程序。除了在高速缓存存储器272内存储的经自动高速缓存的数据之外,在GPGPU核262上执行的线程还可以在共享存储器内以编程方式存储数据。

[0081] 图3A至图3B图示了根据示例性实施例的附加图形多处理器。所图示的图形多处理器325、350是图2C的图形多处理器234的变体。所图示的图形多处理器325、350可以被配置为能够同时执行大量执行线程的流式多处理器(SM)。

[0082] 图3A示出了根据附加的示例性实施例的图形多处理器325。图形多处理器325包括相对于图2D的图形多处理器234的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至334B和(多个)纹理单元344A至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,执行资源单元具有公共指令高速缓存330、纹理和/或数据高速缓存存储器342和共享存储器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实现在图形多处理器325的各种部件之间的通信。在一个实施例中,互连结构327是分离的高速网络结构层,在其上堆叠图形多处理器325的每个部件。图形多处理器325的部件经由互连结构327而与远程部件通信。例如,GPGPU核336A-336B、337A-337B以及3378A-338B可以各自经由互连结构327而与共享存储器346通信。互连结构327可以对图形多处理器325内的通信进行仲裁以确保部件之间公平的带宽分配。

[0084] 图3B示出了根据附加的示例性实施例的图形多处理器350。如图2D和图3A所示,图形处理器包括多组执行资源356A至356D,其中每组执行资源均包括多个指令单元、寄存器堆、GPGPU核和加载存储单元。执行资源356A至356D可以与(多个)纹理单元360A至360D一起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。各种部件可以经由与图3A的互连结构327类似的互连结构352进行通信。

[0085] 本领域技术人员将理解,图1、图2A至图2D和图3A至图3B中所述的架构是描述性的,而不限本发明的实施例的范围,本发明的实施例是示例性的。因此,本文所述的技术可以在任何适当配置的处理单元上实现,该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,诸如图2A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,而不脱离本文所述的实施例的范围。

[0086] 在一些实施例中,如本文所述的并行处理器或GPGPU通信地耦合至主机/处理器核以加快图形操作、机器学习操作、模式分析操作和各种通用GPU(GPGPU)功能。GPU可以通过总线或其他互连(例如,诸如PCIe或NVLink之类的高速互连)通信地耦合至主机处理器/核。在其他实施例中,GPU可以与核一样集成在相同的封装或芯片上并且通过内部处理器总线/互连(即,在封装或芯片内部)通信地耦合至所述核。不管GPU连接的方式如何,处理器核都

可以以工作描述符中包含的命令/指令的序列的形式向GPU分配工作。然后,GPU使用专用电路/逻辑来高效地处理这些命令/指令。

[0087] 用于GPU到主机处理器互连的技术

[0088] 图4A图示了其中多个GPU 410至413通过高速链路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的示例性架构。在一个实施例中,高速链路440、429、442和443支持4GB/s、30GB/s、80GB/s或更高的通信吞吐量,这取决于实现。可以使用各种互连协议,包括但不限于PCIe 4.0或5.0和NVLink 2.0。然而,本发明的基本原理不限于任何特定的通信协议或吞吐量。

[0089] 此外,并且在一个实施例中,GPU 410至413中的两个或更多个通过高速链路444和449互连,这可以使用与用于高速链路440、429、442和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中的两个或更多个可以通过高速链路433连接,所述高速链路可以是以20GB/s、30GB/s、120GB/s或更高的速度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均可以使用相同的协议/链路(例如,通过公共互连结构)来完成。然而,如所提及的,本发明的基本原理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分别通过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和450至453可以利用相同或不同的存储器访问技术。作为示例而不是作为限制,处理器存储器401至402和GPU存储器420至423可以是诸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)之类的易失性存储器,和/或可以是诸如3D XPoint或Nano-Ram之类的非易失性存储器。在一个实施例中,存储器的某个部分可以是易失性存储器,而另一个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] 如下所述,尽管各种处理器405至406和GPU 410至413均可以分别物理地耦合至特定存储器401至402、420至423,但可以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)分发在所有各种物理存储器当中。例如,处理器存储器401至402可以各自包括64GB的系统存储器地址空间,并且GPU存储器420至423可以各自包括32GB的系统存储器地址空间(导致在该示例中产生总共256GB的可寻址存储空间)。

[0092] 图4B图示了根据一个示例性实施例的多核处理器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可以包括集成在经由高速链路440耦合至处理器407的线卡上的一个或多个GPU芯片。可替代地,图形加速模块446可以与处理器407一样集成在相同的封装或芯片上。

[0093] 所图示的处理器407包括多个核460A至460D,这些核各自具有转换后备缓冲器461A至461D和一个或多个高速缓存462A至462D。这些核可以包括用于执行指令和处理未图示的数据以避免模糊本发明的基本原理的各种其他部件(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可以包括1级(L1)和2级(L2)高速缓存。此外,一个或多个共享高速缓存426可以被包括在高速缓存层级结构中并由各组核460A至460D共享。例如,处理器407的一个实施例包括24个核,这些核各自具有它自己的L1高速缓存、12个共享L2高速缓存和12个共享L3高速缓存。在这个实施例中,L2高速缓存和

L3高速缓存中的一个由两个相邻核共享。处理器407和图形加速器集成模块446与系统存储器441连接,所述系统存储器可以包括处理器存储器401至402。

[0094] 通过一致性总线464经由核间通信来为各种高速缓存462A至462D、456和系统存储器441中存储的数据和指令保持一致性。例如,每个高速缓存均可以具有与其关联的高速缓存一致性逻辑/电路,以响应于所检测的对特定高速缓存行的读取或写入而通过一致性总线464进行通信。在一个实现中,通过一致性总线464实现高速缓存窥探协议以窥探高速缓存访问。本领域技术人员很好理解高速缓存窥探/一致性技术,并且这里不会详细描述该高速缓存窥探/一致性技术以避免模糊本发明的基本原理。

[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作为核的对等体参与高速缓存一致性协议。具体地,接口435通过高速链路440(例如,PCIe总线、NVLink等)向代理电路425提供连接性,并且接口437将图形加速模块446连接至链路440。

[0096] 在一个实现中,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、43N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、43N可以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、43N可以在GPU内包括不同类型的图形处理引擎,诸如图形执行单元、媒体处理引擎(例如,视频编码器/解码器)、采样器和块图像传输引擎。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、43N的GPU,或图形处理引擎431至432、43N可以是集成在公共包、线卡或芯片上的单独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,所述存储器管理单元用于执行诸如虚拟到物理存储器转换(也称为有效到实际存储器转换)之类的各种存储器管理功能和用于访问系统存储器441的存储器访问协议。MMU 439还可以包括用于高速缓存虚拟/有效到物理/实际地址转换的转换后备缓冲器(TLB)(未示出)。在一个实现中,高速缓存438存储用于由图形处理引擎431至432、43N高效访问的命令和数据。在一个实施例中,使高速缓存438和图形存储器433至434、43N中存储的数据与核高速缓存462A至462D、456和系统存储器411保持一致。如所提及的,这可以经由代理电路425来完成,所述代理电路代表高速缓存438和存储器433至434、43N参与高速缓存一致性机制(例如,向高速缓存438发送与处理器高速缓存462A至462D、456上的高速缓存行的修改/访问相关的更新并从高速缓存438接收更新)。

[0098] 一组寄存器445存储由图形处理引擎431至432、43N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可以执行保存和恢复操作以在上下文切换期间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,使得第二线程可以由图形处理引擎执行)。例如,在上下文切换时,上下文管理电路448可以将当前寄存器值存储到存储器中的指定区域(例如,由上下文指针标识)。然后,所述上下文管理电路可以在返回上下文时恢复寄存器值。在一个实施例中,中断管理电路447接收并处理从系统设备所接收的中断。

[0099] 在一个实现中,由MMU 439将来自图形处理引擎431的虚拟/有效地址转换为系统存储器411中的实际/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个、8个、16个)图形加速器模块446和/或其他加速器设备。图形加速器模块446可以专用于在处

理器407上执行的单个应用,或者可以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,其中图形处理引擎431至432、43N的资源与多个应用或虚拟机 (VM) 共享。资源可以被细分为基于与VM和/或应用相关联的处理要求和优先级而分配给不同的VM和/或应用的“分片”。

[0100] 因此,加速器集成电路充当图形加速模块446的系统的桥,并提供地址转换和系统存储器高速缓存服务。此外,加速器集成电路436可以为主机处理器提供虚拟化设施以管理图形处理引擎、中断和存储器管理的虚拟化。

[0101] 由于图形处理引擎431至432、43N的硬件资源显式地映射到由主机处理器407看到的实际地址空间,因此任何主机处理器都可以使用有效地址值来为这些资源直接寻址。在一个实施例中,加速器集成电路436的一个功能是图形处理引擎431至432、43N的物理分离,使得它们作为独立单元出现在系统上。

[0102] 如所提及的,在所图示的实施例中,一个或多个图形存储器433至434、43M分别耦合至图形处理引擎431至432、43N中的每一个。图形存储器433至434、43M存储正由图形处理引擎431至432、43N中的每一个处理的指令和数据。图形存储器433至434、43M可以是诸如DRAM(包括堆叠式DRAM)、GDDR存储器(例如,GDDR5、GDDR6)或HBM之类的易失性存储器,和/或可以是诸如3D XPoint或Nano-Ram之类的非易失性存储器。

[0103] 在一个实施例中,为了减少链路440上的数据流量,使用偏置技术来确保图形存储器433至434、43M中存储的数据是图形处理引擎431至432、43N最频繁使用且核460A至460D优选不使用(至少不频繁使用)的数据。类似地,偏置机制试图使核(并且优选地不是图形处理引擎431至432、43N)所需的数据保持在核和系统存储器411的高速缓存462A至462D、456内。

[0104] 图4C图示了其中加速器集成电路436集成在处理器407内的另一示例性实施例。在这个实施例中,图形处理引擎431至432、43N经由接口437和接口435来直接通过高速链路440与加速器集成电路436进行通信(这也可以利用任何形式的总线或接口协议)。加速器集成电路436可以执行与关于图4B所描述的操作相同的操作,但考虑到其与一致性总线464和高速缓存462A至462D、426紧密接近,可能以较高的吞吐量执行操作。

[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(不具有图形加速模块虚拟化)和共享编程模型(具有虚拟化)。共享编程模型可以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、43N在单个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集中到图形引擎431至432、43N,从而在VM/分区内提供虚拟化。

[0107] 在专用进程编程模型中,图形处理引擎431至432、43N可以由多个VM/应用分区共享。共享模型要求系统管理程序以将图形处理引擎431至432、43N虚拟化,以允许由每个操作系统进行访问。对于没有管理程序的分分区系统,图形处理引擎431至432、43N由操作系统拥有。在这两种情况下,操作系统都可以将图形处理引擎431至432、43N虚拟化以提供对每个进程或应用的访问。

[0108] 对于共享编程模型,图形加速模块446或单独图形处理引擎431至432、43N使用进程句柄来选择处理元件。在一个实施例中,处理元件被存储在系统存储器411中并且可使用

本文所述的有效地址到实际地址转换技术来寻址。所述进程句柄可以是在向图形处理引擎431至432、43N注册它的上下文(即,调用系统软件以向处理元件链接表添加处理元件)时向主机进程提供的特定于实现的值。所述进程句柄的低16位可以是处理元件链接表内的处理元件的偏移量。

[0109] 图4D图示了示例性加速器集成分片490。如本文所用,“分片”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存储处理元件483。在一个实施例中,处理元件483响应于来自在处理器407上执行的应用480的GPU调用481而被存储。处理元件483包含对应应用480的处理状态。处理元件483中包含的工作描述符(WD)484可以是应用所请求的单个作业,或者可以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/或单独图形处理引擎431至432、43N可以由系统中的全部进程或进程子集共享。本发明的实施例包括用于建立处理状态并向图形加速模块446发送WD 484以在虚拟化环境中开始作业的基础结构。

[0111] 在一个实现中,专用进程编程模型是特定于实现的。在这个模型中,单个进程拥有图形加速模块446或单独的图形处理引擎431。由于图形加速模块446由单个进程拥有,因此管理程序初始化加速器集成电路436以获得所属分区,并且操作系统在图形加速模块446被指派时初始化加速器集成电路436以获得所属进程。

[0112] 在操作中,加速器集成分片490中的WD获取单元491获取下一个WD 484,所述下一个WD包括将由图形加速模块446的图形处理引擎之一进行的工作的指示。如图所示,来自WD 484的数据可以被存储在寄存器445中并由MMU 439、中断管理电路447和/或上下文管理电路448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/页步行(walk)电路。中断管理电路447可以处理从图形加速模块446所接收的中断事件492。当执行图形操作时,由图形处理引擎431至432、43N生成的有效地址493由MMU 439转换为实际地址。

[0113] 在一个实施例中,针对每个图形处理引擎431至432、43N和/或图形加速模块446复制同一组寄存器445,并且可以由管理程序或操作系统初始化这一组寄存器。这些复制的寄存器中的每一个均可以被包括在加速器集成分片490中。表1中示出了可以由管理程序初始化的示例性寄存器。

[0114] 表1-管理程序初始化寄存器

[0115]

1	分片控制寄存器
2	实际地址 (RA) 调度进程区域指针
3	授权掩码覆盖寄存器
4	中断向量表项偏移
5	中断向量表项极限
6	状态寄存器
7	逻辑分区ID
8	实际地址 (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中示出了可以由操作系统初始化的示例性寄存器。

[0117] 表2-操作系统初始化寄存器

[0118]	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	授权掩码
	6	工作描述符

[0119] 在一个实施例中,每个WD 484均特定于特定图形加速模块446和/或图形处理引擎431至432、43N。所述WD包含图形处理引擎431至432、43N完成其工作所需的所有信息,或者所述WD可以是指向应用已经建立了要完成的工作命令队列的存储器位置的指针。

[0120] 图4E图示了共享模型的一个示例性实施例的附加细节。所述实施例包括其中存储了处理元件列表499的管理程序实际地址空间498。管理程序实际地址空间498可经由管理程序496来访问,所述管理程序将操作系统495的图形加速模块引擎虚拟化。

[0121] 共享编程模型允许来自系统中的全部分区或分区子集的全部进程或进程子集使用图形加速模块446。有两种编程模型,其中图形加速模块446由多个进程和分区共享:时间分片共享和图形直接共享。

[0122] 在这个模型中,系统管理程序496拥有图形加速模块446并且使其功能对所有操作系统495可用。为使图形加速模块446支持系统管理程序496的虚拟化,图形加速模块446可以遵守以下要求:1) 应用作业请求必须是自主的(即,不需要维持作业之间的状态),或者图形加速模块446必须提供上下文保存和恢复机制。2) 图形加速模块446保证在指定时间量内完成应用作业请求,包括任何转换错误,或者图形加速模块446提供抢占作业处理的能力。3) 当以直接共享编程模型操作时,必须为图形加速模块446保证进程之间的公平性。

[0123] 在一个实施例中,对于共享模型,要求应用480以利用图形加速模块446类型、工作描述符(WD)、授权掩码寄存器(AMR)值以及上下文保存/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类型描述了系统调用的目标加速功能。图形加速模块446类型可以是特定于系统的值。所述WD专门针对图形加速模块446来格式化,并且可以呈以下形式:图形加速模块446命令;指向用户定义结构的有效地址指针;指向命令队列的有效地址指针;或用于描述将由图形加速模块446进行的工作的任何其他数据结构。在一个实施例中,AMR值是用于当前进程的AMR状态。传递给操作系统的值与设置AMR的应用类似。如果加速器集成电路436和图形加速模块446的实现不支持用户授权掩码覆盖寄存器(UAMOR),则操作系统可以在管理程序调用中传递AMR之前向AMR值应用当前UAMOR值。在将AMR置于处理元件483之前,管理程序496可以可选地应用当前授权掩码覆盖寄存器(AMOR)值。在一个实施例中,CSRP是包含应用地址空间482中供图形加速模块446保存和恢复上下文状态的区域的有效地址的寄存器445中的一个。如果不要要求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。所述上下文保存/恢复区域可以是插接的系统存储器。

[0124] 在接收到系统调用时,操作系统495可以验证应用480已注册并被授权使用图形加速模块446。操作系统495然后利用表3中所示的信息来调用管理程序496。

[0125] 表3-OS对管理程序的调用参数

[0126]	1	工作描述符 (WD)
	2	授权掩码寄存器 (AMR) 值 (可能已掩蔽)
	3	有效地址 (EA) 上下文保存/恢复区域指针 (CSRP)
	4	进程 ID (PID) 和可选的线程 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已注册并被授权使用图形加速模块446。管理程序496然后将处理元件483针对对应图形加速模块446类型放入处理元件链接表中。处理元件可以包括表4中所示的信息。

[0128] 表4-处理元件信息

[0129]	1	工作描述符 (WD)
	2	授权掩码寄存器 (AMR) 值 (可能已掩蔽)
	3	有效地址 (EA) 上下文保存/恢复区域指针 (CSRP)
	4	进程 ID (PID) 和可选的线程 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
[0130]	7	逻辑中断服务号 (LISN)
	8	中断向量表, 从管理程序调用参数导出
	9	状态寄存器 (SR) 值
	10	逻辑分区 ID (LPID)
	11	实际地址 (RA) 管理程序加速器利用记录指针
	12	存储描述符寄存器 (SDR)

[0131] 在一个实施例中,管理程序将寄存器445的多个加速器集成分片490初始化。

[0132] 如图4F所示,本发明的一个示例性实施例采用可经由用于访问物理处理器存储器401至402和GPU存储器420至423的公共虚拟存储器地址空间来寻址的统一存储器。在这个实现中,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,反之亦然,由此简化可编程性。在一个实施例中,将虚拟/有效地址空间的第一部分分配给处理器存储器401,将第二部分分配给第二处理器存储器402,将第三部分分配给GPU存储器420,以此类推。整个虚拟/有效存储器空间 (有时称为有效地址空间) 由此分布在处理器存储器401至402和GPU存储器420至423中的每一个上,从而允许任何处理器或GPU访问具有映射到该存储器的虚拟地址的任何物理存储器。

[0133] 在一个实施例中,MMU 439A至439E中的一个或多个内的偏置/一致性管理电路494A至494E确保了主机处理器 (例如,405) 与GPU 410至413的高速缓存之间的高速缓存一致性,并实现指示其中应当存储某些类型的数据的物理存储器的偏置技术。尽管在图4F中图示了偏置/一致性管理电路494A至494E的多个实例,但偏置/一致性电路也可以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。

[0134] 一个实施例允许将GPU附接的存储器420至423映射为系统存储器的一部分,并使

用共享虚拟存储器 (SVM) 技术进行访问,但不会遭受与全系统高速缓存一致性相关联的典型性能缺陷。GPU附接的存储器420至423作为系统存储器来访问的能力不会造成繁重的高速缓存一致性开销,这为GPU卸载提供了有利的操作环境。这种安排允许主机处理器405软件设置操作数并访问计算结果,而不具有传统I/ODMA数据拷贝的开销。这些传统拷贝涉及驱动器调用、中断和存储器映射I/O (MMIO) 访问,这些访问相对于简单内存访问来说都是低效的。同时,在不具有高速缓存一致性开销的情况下访问GPU附接存储器420至423的能力对于卸载计算的执行时间可能是关键的。例如,在具有大量流式写入存储器业务的情况下,高速缓存一致性开销可以显著降低由GPU 410至413看到的有效写入带宽。操作数设置的效率、结果访问的效率以及GPU计算的效率都在确定GPU卸载的有效性方面发挥着重要作用。

[0135] 在一个实现中,GPU偏置与主机处理器偏置之间的选择由偏置跟踪器数据结构驱动。例如,可以使用偏置表,所述偏置表可以是每个GPU附接存储器页包括1或2个位的页面粒度结构(即,以存储器页的粒度来控制)。偏置表可以在一个或多个GPU附接存储器420至423的被盗存储器范围内实现,在GPU 410至413中具有或不具有偏置高速缓存(例如,以高速缓存频繁/最近使用的偏置表的项)。可替代地,整个偏置表均可以保持在GPU内。

[0136] 在一个实现中,在实际访问GPU存储器之前访问与对GPU附接存储器420至423的每次访问相关联的偏置表项,从而导致以下操作。首先,将来自GPU 410至413的在GPU偏置中发现其页面的本地请求直接转发到对应的GPU存储器420至423。将来自GPU的在主机偏置中发现其页面的本地请求转发给处理器405(例如,如上所讨论通过高速链路)。在一个实施例中,来自处理器405的在主机处理器偏置中发现所请求的页面的请求完成了像正常存储器读取那样的请求。可替代地,可以将针对GPU偏置页面的请求转发给GPU 410至413。然后,如果GPU当前未使用所述页面,则GPU可以将所述页面转变到主机处理器偏置。

[0137] 页面的偏置状态可以通过基于软件的机制、基于硬件辅助软件的机制或者对于一组有限的情况基于仅硬件的机制来改变。

[0138] 一种用于改变偏置状态的机制采用API调用(例如OpenCL),所述API调用继而调用GPU设备驱动器,所述设备驱动器继而向GPU发送消息(或将命令描述符入队),从而引导所述GPU改变偏置状态,并且对于某些转变,在主机中执行高速缓存转储清除操作。所述高速缓存转储清除操作是从主机处理器405偏置到GPU偏置的转变所必需的,而对于相反转变则不是必需的。

[0139] 在一个实施例中,通过暂时呈现主机处理器405不可高速缓存的GPU偏置页面来保持缓存一致性。为了访问这些页面,处理器405可以请求来自GPU 410的访问,取决于实现,GPU 410立即可以授权访问或者可以不授权访问。因此,为了减少处理器405与GPU 410之间的通信,有利的是确保GPU偏置页面是GPU所需但不是主机处理器405所需的页面,反之亦然。

[0140] 图形处理流水线

[0141] 图5图示了根据示例性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图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所述的并行处理子系统内,所述并行处理子系统诸如是图2A的并行处理器200,在一个实施例中,所述并行处理器200是图1的(多个)并行处理器112的变体。如本文所述,各种并行处理系统可以经由并行处理单元(例如,图2A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2D的图形多处理器234)可以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中的一个或多个的功能。数据汇编器502、图元汇编器506、514、518、曲面细分单元510、栅格器522和栅格操作单元526的功能还可以由处理集群(例如,图2C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2A的分区单元220A至220N)执行。图形处理流水线500还可以使用一个或多个功能的专用处理单元来实现。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由通用处理器(例如,CPU)内的并行处理逻辑执行。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访问片上存储器(例如,如图2A中所示的并行处理器存储器222),所述存储器接口可以是图2A的存储器接口218的实例。

[0142] 在一个实施例中,数据汇编器502是收集表面和图元的顶点数据的处理单元。数据汇编器502然后向顶点处理单元504输出包括顶点属性的顶点数据。顶点处理单元504是可编程执行单元,所述可编程执行单元执行顶点着色器程序,从而照明和变换如顶点着色器程序所指定的顶点数据。顶点处理单元504读取高速缓存、本地或系统存储器中存储的用于处理顶点数据的数据,并且可以被编程为将顶点数据从基于对象的坐标表示变换为世界空间坐标空间或归一化设备坐标空间。

[0143] 图元汇编器506的第一实例从顶点处理单元50接收顶点属性。图元汇编器506根据需要读取所存储的顶点属性并构造图形图元以由曲面细分控制处理单元508进行处理。图形图元包括如各种图形处理应用编程接口(API)所支持的三角形、线段、点、补丁等等。

[0144] 曲面细分控制处理单元508将输入顶点视为几何补丁的控制点。这些控制点来自补丁的输入表示(例如,补丁的基础)变换为适用于由曲面细分评估处理单元512进行表面评估的表示。曲面细分控制处理单元508还可以计算几何补丁的边缘的曲面细分因子。曲面细分因子适用于单个边缘,并量化与边缘相关联的视点相关细节水平。曲面细分单元510被配置成接收补丁的边缘的曲面细分因子并将补丁曲面细分为多个几何图元,诸如线、三角形或四边形图元,所述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对细分的补丁的参数化坐标进行操作以生成与几何图元相关联的每个顶点的表面表示和顶点属性。

[0145] 图元汇编器514的第二实例从曲面细分评估处理单元512接收顶点属性,根据需要读取所存储的顶点属性,并构造图形图元以由几何处理单元516处理。几何处理单元516是可编程执行单元,所述可编程执行单元执行几何着色器程序,以变换如几何着色器程序所指定的从图元汇编器514所接收的图形图元。在一个实施例中,几何处理单元516被编程为将图形图元细分为一个或多个新的图形图元并且计算用于将新的图形图元栅格化的参数。

[0146] 在一些实施例中,几何处理单元516可以在几何流中添加或删除元素。几何处理单元516向图元汇编器518输出指定新图形图元的参数和顶点。图元汇编器518从几何处理单元516接收参数和顶点,并构建图形图元以供视口缩放、剔除和裁剪单元520进行处理。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用于处理几何数据。视口缩放、剔除和裁剪单元520执行裁剪、剔除和视口缩放,并且将已处理的图形图元输出到栅格器522。

[0147] 栅格器522可以执行深度剔除和其他基于深度的优化。栅格器522还对新图形图元

执行扫描转换以生成片段并向片段/像素处理单元524输出这些片段和关联的覆盖数据。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变换从栅格器522所接收的片段或像素,如片段或像素着色器程序所指定的。例如,片段/像素处理单元524可以被编程为执行包括但不限于纹理映射、着色、混合、纹理校正和透视校正的操作,以产生输出到栅格操作单元526的着色片段或像素。片段/像素处理单元524可以读取并行处理器存储器或系统存储器中存储的数据,以在处理片段数据时使用。片段或像素着色器程序可以被配置成依据针对处理单元进行配置的采样速率以样本、像素、图块或其他粒度着色。

[0148] 栅格操作单元526是执行包括但不限于模板印刷、z测试、混合等的栅格操作的处理单元,并且将像素数据作为经处理图形数据输出以存储在图形存储器(例如,如图2A中的并行处理器存储器222和/或如图1中的系统存储器104)中,以在一个或多个显示设备110A、110B上进行显示或者由一个或多个处理器102或(多个)并行处理器112中的一个进一步处理。在一些实施例中,栅格操作单元526被配置成压缩被写入到存储器的z或颜色数据并解压缩从存储器读取的z或颜色数据。

[0149] 机器学习概述

[0150] 机器学习算法是可以基于一组数据来学习的算法。机器学习算法的实施例可以被设计成对数据集内的高级抽象进行建模。例如,图像识别算法可以用于确定给定的输入属于若干种类别中的哪一种;回归算法可以在给定输入的情况下输出数值;并且模式识别算法可以用于生成翻译文本或执行文本至语音和/或语音识别。

[0151] 一种示例性类型的机器学习算法是神经网络。存在许多类型的神经网络;一种简单类型的神经网络是前馈网络。可以将前馈网络实现为无环图,其中节点布置在层中。通常,前馈网络拓扑包括输入层和输出层,输入层和输出层通过至少一个隐藏层分开。隐藏层将由输入层接收到的输入变换为对在输出层中生成输出有用的表示。网络节点经由边缘全连接至相邻层中的节点,但每个层内的节点之间不存在边缘。在前馈网络的输入层的节点处接收的数据经由激活函数被传播(即,“前馈”)至输出层的节点,所述激活函数基于系数(“权重”)来计算网络中的每个连续层的节点的状态,所述系数分别与连接这些层的边缘中的每一个相关联。取决于由执行的算法所表示的特定模型,来自神经网络算法的输出可以采用各种形式。

[0152] 在可以使用机器学习算法来对具体问题建模之前,使用训练数据集来训练所述算法。训练神经网络涉及:选择网络拓扑;使用表示被网络建模的问题的一组训练数据;以及调节权重,直到网络模型针对训练数据集的所有实例表现为具有最小误差。例如,在用于神经网络的监督式学习训练过程期间,将由网络响应于表示训练数据集中的实例的输入所产生的输出与该实例的“正确”的已标记输出相比较;计算表示所述输出与已标记输出之间的差异的误差信号;以及当将误差信号向后传播穿过网络的层时,调节与所述连接相关联的权重以最小化该误差。当从训练数据集的实例中生成的每个输出的误差被最小化时,网络被视为“已经过训练”。

[0153] 机器学习算法的准确度会受到用于训练所述算法的数据集的质量的很大影响。训练过程可以是计算密集型的,并且在常规通用处理器上可能需要大量的时间。因此,使用并行处理硬件来训练许多类型的机器学习算法。这对于优化神经网络的训练是特别有用的,

因为在调节神经网络中的系数时执行的计算本身自然地适于并行实现方式。具体地,许多机器学习算法和软件应用已被适配成在通用图形处理设备内使用并行处理硬件。

[0154] 图6是机器学习软件堆叠600的广义图。机器学习应用602可以被配置成使用训练数据集来训练神经网络或使用已训练的深度神经网络来实现机器智能。机器学习应用602可以包括神经网络和/或专用软件的训练和推断功能,所述功能可以用于在部署之前训练神经网络。机器学习应用602可以实现任何类型的机器智能,包括但不限于:图像识别、映射和定位、自主导航、语音合成、医学成像或语言翻译。

[0155] 可以经由机器学习框架604来实现针对机器学习应用602的硬件加速。机器学习框架604可以提供机器学习图元的库。机器学习图元是机器学习算法通常执行的基本操作。在没有机器学习框架604的情况下,将需要机器学习算法的开发者创建和优化与机器学习算法相关联的主要计算逻辑,然后在开发出新的并行处理器时重新优化所述计算逻辑。相反,机器学习应用可以被配置成使用由机器学习框架604提供的图元来执行必要的计算。示例性图元包括张量卷积、激活函数和池化,它们是在训练卷积神经网络(CNN)时执行的计算操作。机器学习框架604还可以提供图元以用于实现由许多机器学习算法执行的基本线性代数子程序,诸如矩阵和向量运算。

[0156] 机器学习框架604可以处理从机器学习应用602接收的输入数据,并生成至计算框架606的适当输入。计算框架606可以使提供给GPGPU驱动器608的底层指令抽象化,以使得机器学习框架604能够经由GPGPU硬件610来利用硬件加速而无需机器学习框架604非常熟悉GPGPU硬件610的架构。另外,计算框架606可以跨越多种类型和各代GPGPU硬件610来实现针对机器学习框架604的硬件加速。

[0157] GPGPU机器学习加速

[0158] 图7图示了根据示例性实施例的高度并行的通用图形处理单元700。在一个实施例中,通用处理单元(GPGPU)700可以被配置成在处理与训练深度神经网络相关联的这种类型的计算工作负荷中特别高效。另外,GPGPU 700可以直接链接至GPGPU的其他实例以用于创建多GPU集群,从而改进特别深的神经网络的训练速度。

[0159] GPGPU 700包括主机接口702以用于实现与主机处理器的连接。在一个实施例中,主机接口702是PCI Express接口。然而,主机接口还可以是供应方特定的通信接口或通信结构。GPGPU 700从主机处理器接收命令,并使用全局调度器704以将与那些命令相关联的执行线程分发至一组计算集群706A至H。计算集群706A至H共享高速缓存存储器708。高速缓存存储器708可以充当计算集群706A至H内的高速缓存存储器中的较高级高速缓存。

[0160] GPGPU 700包括存储器714A至B,所述存储器经由一组存储器控制器712A至B与计算集群706A至H耦合。在各种实施例中,存储器714A至B可以包括各种类型的存储器设备,包括动态随机存取存储器(DRAM)或图形随机存取存储器(诸如,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N还可以包括3D堆叠式存储器,包括但不限于高带宽存储器(HBM)。

[0161] 在一个实施例中,每个计算集群706A至H包括一组图形多处理器,诸如图2D的图形多处理器234。计算集群的图形多处理器包括多种类型的整数和浮点逻辑单元,这些单元可以在一系列精度(包括适合于机器学习计算的精度)下执行计算操作。例如且在一个实施例中,计算集群706A至H中的每一个中的浮点单元的至少子集可以被配置成执行16位或32位

浮点运算,而浮点单元的不同子集可以被配置成执行64位浮点运算。

[0162] GPGPU 700的多个实例可以被配置成作为计算集群来操作。由计算集群用于同步和数据交换的通信机制跨实施例变化。在一个实施例中,GPGPU 700的多个实例通过主机接口702来通信。在一个实施例中,GPGPU 700包括使GPGPU 700与GPU链路710耦合的I/O中枢709,所述GPU链路实现至GPGPU的其他实例的直接连接。在一个实施例中,GPU链路710耦合至专用GPU-GPU桥,所述GPU-GPU桥实现GPGPU 700的多个实例之间的通信和同步。在一个实施例中,GPU链路710与高速互连耦合,以用于将数据传输和接收至其他GPGPU或并行处理器。在一个实施例中,GPGPU 700的多个实例位于单独的数据处理系统中并且经由网络设备来通信,所述网络设备可经由主机接口702来访问。在一个实施例中,除主机接口702之外或作为主机接口702的替代例,GPU链路710可以被配置成使得能够连接至主机处理器。

[0163] 虽然GPGPU 700的所图示配置可以被配置成训练神经网络,但是一个实施例提供了GPGPU 700的替代性配置,其可以被配置成用于部署在高性能或低功率推断平台内。在推断配置中,GPGPU 700包括相对于训练配置更少的计算集群706A至H。另外,与存储器714A至B相关联的存储器技术可以在推断和训练配置之间有所不同。在一个实施例中,GPGPU 700的推断配置可以支持推断特定的指令。例如,推断配置可以提供对一个或多个8位整数点积指令的支持,这些指令通常在用于已部署神经网络的推断操作期间使用。

[0164] 图8图示了根据示例性实施例的多GPU计算系统800。多GPU计算系统800可以包括处理器802,所述处理器经由主机接口开关804耦合至多个GPGPU 806A至D。在一个实施例中,主机接口开关804是将处理器802耦合至PCI Express总线的PCI Express开关设备,处理器802可以通过所述PCI Express总线与这组GPGPU 806A至D通信。多个GPGPU 806A至D中的每一个可以是图7的GPGPU 700的实例。GPGPU 806A至D可以经由一组高速点对点GPU-GPU链路816互连。高速GPU-GPU链路可以经由专用GPU链路(诸如,如图7中的GPU链路710)连接至GPGPU 806A至D中的每一个。P2P GPU链路816使得GPGPU 806A至D中的每一个之间能够直接通信,而无需通过主机接口总线(处理器802连接至所述主机接口总线)来通信。在GPU-GPU业务针对P2P GPU链路的情况下,主机接口总线仍然可用于系统存储器访问或与多GPU计算系统800的其他实例通信(例如,经由一个或多个网络设备)。虽然在所图示的实施例中GPGPU 806A至D经由主机接口开关804连接至处理器802,但是在一个实施例中,处理器802包括对P2P GPU链路816的直接支持并且可以直接连接至GPGPU 806A至D。

[0165] 机器学习神经网络实现由本文描述的实施例提供的计算架构可以被配置成执行特别适合于训练和部署用于机器学习的神经网络的这些类型的并行处理。可以将神经网络一般化为具有图表关系的函数的网络。如本领域中众所周知的,存在机器学习中所使用的多种类型的神经网络实现方式。一种示例性类型的神经网络是如先前描述的前馈网络。

[0166] 第二种示例性类型的神经网络是卷积神经网络(CNN)。CNN是用于处理具有已知的、网格状拓扑的数据(诸如,图像数据)的专用前馈神经网络。因此,CNN通常用于计算机视觉和图像识别应用,但它们也可以用于其他类型的模式识别,诸如语音和语言处理。CNN输入层中的节点被组织为一组“滤波器”(受视网膜中发现的感受野启发的特征检测器),并且每一组滤波器的输出被传播至网络的连续层中的节点。用于CNN的计算包括将卷积数学运算应用于每个滤波器以产生该滤波器的输出。卷积是由两个函数执行以产生第三个函数的一种专门的数学运算,所述第三个函数是两个原始函数中的一个的修改版本。在卷积网络

术语中,关于卷积的第一个函数可以被称为输入,而第二个函数可以被称为卷积内核。输出可以被称为特征图。例如,至卷积层的输入可以是多维数据阵列,其定义输入图像的各种颜色分量。卷积内核可以是多维参数阵列,其中通过针对神经网络的训练过程来适配所述参数。

[0167] 递归神经网络(RNN)是一类前馈神经网络,其包括层之间的反馈连接。RNN使得能够通过跨神经网络的不同部分共享参数数据来对序列数据进行建模。RNN的架构包括循环。这些循环表示变量的当前值在未来的时间对其自身值的影响,因为来自RNN的输出数据的至少一部分被用作反馈以用于处理序列中的后续输入。由于语言数据可被组成的可变本质,这个特征使RNN变得对语言处理特别有用。

[0168] 下文描述的图呈现了示例性前馈、CNN和RNN网络,以及描述了用于分别训练和部署那些类型的网络中的每一种的通用过程。将理解,这些描述就本文描述的任何特定实施例而论是示例性且非限制性的,并且一般说来可以通常将所图示的概念应用于深度神经网络和机器学习技术。

[0169] 上文描述的示例性神经网络可以用于执行深度学习。深度学习是使用深度神经网络进行的机器学习。与仅包括单个隐藏层的浅层神经网络相反,深度学习中使用的深度神经网络是由多个隐藏层组成的人工神经网络。更具深度的神经网络通常训练起来更具计算密集性。然而,网络的附加隐藏层实现了多步模式识别,所述多步模式识别相对于浅层机器学习技术导致减少的输出误差。

[0170] 深度学习中使用的深度神经网络通常包括前端网络以用于执行耦合至表示数学模型的后端网络的特征识别,所述数学模型可以基于提供给所述模型的特征表示来执行操作(例如,目标分类、语音识别等)。深度学习使得能够执行机器学习,而无需针对所述模型执行手工特征工程。相反,深度神经网络可以基于输入数据内的统计结构或相关性来学习特征。所学习的特征可以提供给数学模型,所述数学模型可以将所检测的特征映射至输出。由网络使用的数学模型通常专用于待执行的特定任务,并且不同的模型将用于执行不同的任务。

[0171] 一旦将神经网络结构化,就可以将学习模型应用于网络以将网络训练成执行特定任务。学习模型描述如何在模型内调节权重以减少网络的输出误差。反向传播误差是一种用于训练神经网络的常用方法。向网络呈现输入向量以供处理。使用损失函数将网络的输出与期望的输出相比较,并且为输出层中的每个神经元计算误差值。然后,向后传播这些误差值,直到每个神经元具有粗略地表示其对原始输出的贡献的相关误差值。然后,网络可以使用算法(诸如,随机梯度下降算法)从那些误差中学习,以更新神经网络的权重。

[0172] 图9A至B图示了示例性卷积神经网络。图9A图示了CNN内的各种层。如图9A中所示,用于对图像处理进行建模的示例性CNN可以接收输入902,所述输入描述输入图像的红、绿和蓝(RGB)分量。输入902可以由多个卷积层(例如,卷积层904、卷积层906)处理。可选地,来自所述多个卷积层的输出可以由一组全连接层908处理。全连接层中的神经元具有至前一层中的所有激活的全连接,如先前针对前馈网络所描述的。来自全连接层908的输出可以用于从网络中生成输出结果。可以使用矩阵乘法而非卷积来计算全连接层908内的激活。并非所有的CNN实现方式都使用全连接层908。例如,在一些实现方式中,卷积层906可以生成CNN的输出。

[0173] 卷积层被稀疏地连接,这不同于全连接层908中发现的传统神经网络配置。传统神经网络层被全连接,使得每个输出单元与每个输入单元相互作用。然而,卷积层被稀疏地连接,这是因为感受野的卷积的输出(而非感受野中的每个节点的相应状态值)被输入至后续层的节点,如所图示。与卷积层相关联的内核执行卷积运算,所述卷积运算的输出被发送至下一个层。在卷积层内执行的降维是使得CNN能够进行缩放以处理大图像的一个方面。

[0174] 图9B图示了在CNN的卷积层内的示例性计算阶段。可以在卷积层914的三个阶段中处理至CNN的卷积层的输入912。这三个阶段可以包括卷积阶段916、检测器阶段918和池化阶段920。然后,卷积层914可以将数据输出至连续的卷积层。网络的最后一个卷积层可以生成输出特征图数据或提供至全连接层的输入,例如以生成至CNN的输入的分类值。

[0175] 在卷积阶段916中,卷积层914可以并行执行若干个卷积,以产生一组线性激活。卷积阶段916可以包括仿射变换,所述仿射变换是可以被指定为线性变换外加平移的任何变换。仿射变换包括旋转、平移、缩放和这些变换的组合。卷积阶段计算连接至输入中特定区域的函数的输出(例如,神经元),所述特定区域可以被确定为与神经元相关联的局部区域。神经元计算神经元的权重与局部输入(神经元连接至所述局部输入)中的区域之间的点积。来自卷积阶段916的输出定义由卷积层914的连续阶段处理的一组线性激活。

[0176] 线性激活可以由检测器阶段918处理。在检测器阶段918中,每个线性激活由非线性激活函数处理。非线性激活函数增加整体网络的非线性性质,而不影响卷积层的感受野。可以使用若干种类型的非线性激活函数。一个具体的类型是线性修正单元(ReLU),其使用被定义为 $f(x) = \max(0, x)$ 的激活函数,使得激活在零处被阈值化。

[0177] 池化阶段920使用池化函数,所述池化函数用附近输出的概括统计数值来代替卷积层906的输出。池化函数可以用于将平移不变性引入到神经网络中,使得至输入的轻微平移不改变池化输出。局部平移的不变性在输入数据中的特征存在性比特征的精确位置更加重要的场景中可以是有益的。可以在池化阶段920期间使用各种类型的池化函数,包括最大池化、平均池化和L2范数池化。另外,一些CNN实现方式不包括池化阶段。相反,这样的实现方式代用附加的卷积阶段,所述附加的卷积阶段相对于先前的卷积阶段具有增大的步幅。

[0178] 然后,来自卷积层914的输出可以由下一个层922处理。下一个层922可以是附加的卷积层或是全连接层908中的一个。例如,图9A的第一卷积层904可以输出至第二卷积层906,而第二卷积层可以输出至全连接层908中的第一层。

[0179] 图10图示了示例性递归神经网络1000。在递归神经网络(RNN)中,网络的先前状态影响网络的当前状态的输出。可以使用各种各样的函数以各种各样的方式来构建RNN。RNN的使用通常围绕使用数学模型以基于先前的输入序列来预测未来。例如,RNN可以用于执行统计语言建模以在给定先前的字序列的情况下预测即将来临的字。可以将所图示的RNN 1000描述为具有以下各项:输入层1002,其接收输入向量;隐藏层1004,用于实现递归函数;反馈机构1005,用于实现先前状态的“存储器”;以及输出层1006,用于输出结果。RNN 1000基于时间步长来操作。经由反馈机构1005基于先前的时间步长来影响RNN在给定的时间步长的状态。针对给定的时间步长,由先前状态和在当前时间步长的输入来定义隐藏层1004的状态。在第一时间步长的初始输入(x_1)可以由隐藏层1004处理。第二输入(x_2)可以由隐藏层1004使用在处理初始输入(x_1)期间所确定的状态信息来处理。可以将给定的状态计算为 $s_t = f(Ux_t + Ws_{t-1})$,其中,U和W是参数矩阵。函数f通常为非线性,诸如双曲正切函数(Tanh)或

修正函数 $f(x) = \max(0, x)$ 的变体。然而,隐藏层1004中使用的特定数学函数可以取决于RNN 1000的特定实现方式细节而变化。

[0180] 除所描述的基本CNN和RNN网络之外,还可以实现那些网络的变化。一个示例RNN变体是长短期记忆(LSTM)RNN。LSTM RNN能够学习对于处理更长的语言序列来说可能有必要的长期依赖性。CNN的变体是卷积深度置信网络,所述卷积深度置信网络具有类似于CNN的结构并且以类似于深度置信网络的方式受训练。深度置信网络(DBN)是由随机性(随机)变量的多个层组成的生成式神经网络。可以使用贪婪式无监督式学习来逐层训练DBN。然后,DBN的学习权重可以用于通过确定用于神经网络的一组最佳初始权重来提供预训练神经网络。

[0181] 图11图示了深度神经网络的示例性训练和部署。一旦已针对任务将给定的网络结构化,就使用训练数据集1102来训练神经网络。已开发出各种训练框架1104以用于实现对训练过程的硬件加速。例如,图6的机器学习框架604可以被配置为训练框架1104。训练框架1104可以跟未训练的神经网络1106挂钩,并且使得能够使用本文描述的并行处理资源来训练未训练的神经网络以生成已训练的神经网络1108。

[0182] 为了开始训练过程,可以随机地或通过使用深度置信网络进行预训练来选择初始权重。然后,可以以监督或无监督的方式来执行训练循环。

[0183] 监督式学习是一种学习方法,其中将训练作为仲裁操作来执行,诸如当训练数据集1102包括输入(其与所述输入的期望输出成对)时,或在训练数据集包括具有已知的输出的输入并且神经网络的输出被手动地分级的情况下。网络处理输入,并且将所得输出与一组预期或期望的输出相比较。然后,通过系统反向传播误差。训练框架1104可以进行调节,以调节控制未训练的神经网络1106的权重。训练框架1104可以提供工具以用于监视未训练的神经网络1106在多大程度上收敛于适合基于已知的输入数据生成正确的答案的模型。当调节网络的权重以改善由神经网络生成的输出时,反复地出现训练过程。训练过程可以继续,直到神经网络达到与已训练的神经网络1108相关联的统计上期望的准确度。然后,可以部署已训练的神经网络1108以实现任何数量的机器学习操作。

[0184] 无监督式学习是一种学习方法,其中网络试图使用未标记数据来训练其自身。因此,针对无监督式学习,训练数据集1102将包括输入数据而无任何关联的输出数据。未训练的神经网络1106可以学习未标记输入内的分组,并且可以确定个别输入如何与整体数据集相关。无监督式训练可以用于生成自组织映射,所述自组织映射是能够执行在数据降维中有用的操作的一种类型的已训练神经网络1108。无监督式训练还可以用于执行异常检测,所述异常检测允许识别输入数据集中偏离数据正常模式的数据点。

[0185] 还可以采用监督式和无监督式训练的变化。半监督式学习是一种技术,其中训练数据集1102包括相同分布的已标记数据和未标记数据的混合。增量学习是监督式学习的变体,其中连续地使用输入数据以用于进一步训练模型。增量学习使得已训练的神经网络1108能够适配于新数据1112,而不忘记在初始训练期间根植在网络内的知识。

[0186] 不管是监督式还是无监督式,用于特别深的神经网络的训练过程对于单个计算节点而言可能是过于计算密集的。可以使用计算节点的分布式网络而非使用单个计算节点来加速训练过程。

[0187] 图12是图示了分布式学习的示例性框图。分布式学习是训练模型,其使用多个分

布式计算节点来执行神经网络的监督式或无监督式训练。所述分布式计算节点可以各自包括一个或多个主机处理器以及通用处理节点中的一个或多个,诸如如图7中的高度并行的通用图形处理单元700。如所图示,分布式学习可以执行模型并行化1202、数据并行化1204或模型和数据并行化的组合1206。

[0188] 在模型并行化1202中,分布式系统中的不同计算节点可以针对单个网络的不同部分执行训练计算。例如,可以由分布式系统的不同处理节点来训练神经网络的每个层。模型并行化的益处包括缩放到特别大的模型的能力。分裂与神经网络的不同层相关联的计算使得能够训练非常大的神经网络,其中所有层的权重将不纳入(fit into)单个计算节点的存储器中。在一些实例中,模型并行化在执行大型神经网络的无监督式训练中可以是特别有用的。

[0189] 在数据并行化1204中,分布式网络的不同节点具有模型的完整实例,并且每个节点接收数据的不同部分。然后,组合来自不同节点的结果。虽然用于数据并行化的不同方法是有可能的,但是数据并行训练方法都需要一种组合结果并使每个节点之间的模型参数同步的技术。组合数据的示例性方法包括参数求平均和基于更新的数据并行化。参数求平均训练在训练数据的子集上的每个节点,并且将全局参数(例如,权重、偏置)设定至来自每个节点的参数的平均值。参数求平均使用保持参数数据的中心参数服务器。基于更新的数据并行化类似于参数求平均,除了以下情况之外:传递对模型的更新而非将来自节点的数据传递到参数服务器。另外,可以以分散的方式执行基于更新的数据并行化,其中更新被压缩并且在节点之间传递。

[0190] 例如,可以在分布式系统中实现经组合的模型和数据并行化1206,在所述分布式系统中,每个计算节点包括多个GPU。每个节点可以具有模型的完整实例,其中每个节点内的单独GPU用于训练模型的不同部分。

[0191] 分布式训练相对于单个机器上的训练具有增加的开销。然而,本文描述的并行处理器和GPGPU可以各自实现各种技术以用于减少分布式训练的开销,包括用于实现高带宽GPU-GPU数据传递和加速的远程数据同步的技术。

[0192] 示例性机器学习应用

[0193] 可以应用机器学习以解决多种技术问题,包括但不限于计算机视觉、自动驾驶和导航、语音识别以及语言处理。计算机视觉传统上已是机器学习应用的最活跃研究领域之一。计算机视觉的应用范围为从重现人类视觉能力(诸如,识别人脸)到创建新类别的视觉能力。例如,计算机视觉应用可以被配置成从视频中可见的物体中所诱导的振动来识别声波。并行处理器加速的机器学习使得能够使用明显大于先前可行的训练数据集的训练数据集来训练计算机视觉应用,并且使得能够使用低功率并行处理器来部署推断系统。

[0194] 并行处理器加速的机器学习具有自动驾驶应用,包括车道和道路标志识别、障碍物回避、导航和驾驶控制。加速的机器学习技术可以用于基于数据集来训练驾驶模型,所述数据集定义对特定训练输入的适当响应。本文描述的并行处理器可以使得能够快速训练用于自动驾驶解决方案的日益复杂的神经网络,并且使得能够将低功率推断处理器部署在适合于集成到自主车辆中的移动平台中。

[0195] 并行处理器加速的深度神经网络已实现自动语音识别(ASR)的机器学习方法。ASR包括创建在给定的输入声序列的情况下计算最可能的语言序列的函数。使用深度神经网络

的加速的机器学习已实现代替先前用于ASR的隐马尔可夫模型(HMM)和高斯混合模型(GMM)。

[0196] 并行处理器加速的机器学习还可以用于加速自然语言处理。自动学习程序可以使用统计推断算法以产生对于误差的或不熟悉的输入具有鲁棒性的模型。示例性自然语言处理器应用包括人类语言之间的自动机器翻译。

[0197] 可以将用于机器学习的并行处理平台划分为训练平台和部署平台。训练平台通常高度并行,并且包括优化以用于加速多GPU单节点训练和多节点多GPU训练。适合于训练的示例性并行处理器包括图7的高度并行的通用图形处理单元700和图8的多GPU计算系统800。相反,部署的机器学习平台通常包括适合于用在诸如相机、自主机器人和自主车辆之类的产品中的较低功率并行处理器。

[0198] 图13图示了适合于使用训练模型执行推断的示例性推断片上系统(SOC)1300。SOC 1300可以集成处理部件,包括媒体处理器1302、视觉处理器1304、GPGPU 1306和多核处理器1308。SOC 1300可以另外包括片上存储器1305,所述片上存储器1305可以实现可由所述处理部件中的每一个访问的共享片上数据池。所述处理部件可以针对低功率操作被优化,以用于使得能够部署至各种各样的机器学习平台(包括自主车辆和自主机器人)。例如,可以将SOC 1300的一种实现方式用作用于自主车辆的主控制系统的一部分。在SOC 1300被配置成用于自主车辆中的情况下,SOC被设计和配置成用于符合部署管辖权的相关功能安全标准。

[0199] 在操作期间,媒体处理器1302和视觉处理器1304可以一致地工作以加速计算机视觉操作。媒体处理器1302可以使得能够对多个高分辨率(例如,4K、8K)视频流进行低等待时间解码。可以将已解码的视频流写入到片上存储器1305中的缓冲器。然后,视觉处理器1304可以解析已解码的视频,并且对已解码视频的帧执行初步处理操作以准备使用已训练的图像识别模型来处理帧。例如,视觉处理器1304可以加速用于CNN(用于对高分辨率视频数据执行图像识别)的卷积运算,而后端模型计算由GPGPU 1306执行。

[0200] 多核处理器1308可以包括控制逻辑,以用于有助于数据传递的排序和同步以及由媒体处理器1302和视觉处理器1304执行的共享存储器操作。多核处理器1308还可以充当应用处理器,以用于执行可以使用GPGPU 1306的推断计算能力的软件应用。例如,可以在多核处理器1308上执行的软件实现导航和驾驶逻辑的至少一部分。这样的软件可以直接将计算工作负荷发布给GPGPU 1306,或可以将计算工作负荷发布给多核处理器1308,所述多核处理器可以将那些操作的至少一部分卸载到GPGPU 1306。

[0201] GPGPU 1306可以包括计算集群,诸如高度并行的通用图形处理单元700内的计算集群706A至706H的低功率配置。GPGPU 1306内的计算集群可以支持被具体地优化以用于对已训练的神经网络执行推断计算的指令。例如,GPGPU 1306可以支持用于执行低精度计算(诸如,8位和4位整数向量运算)的指令。

[0202] 图14图示了用于图像捕获设备的示例性图像捕获系统。在一些实施例中,图像捕获设备是独立的设备,诸如数字相机。在其他实施例中,图像捕获设备是需要图像捕获能力的另一计算设备的一部分或子部件,该另一计算设备诸如便携式或手持式计算设备,其具有数字相机来捕获图像。在示例性实施例中,将图像捕获设备与卷积神经网络(CNN)计算处理集成,所述CNN诸如深度学习神经网络(DNN),其可以实时地、或与图像捕获设备进行图像

捕获和存储同时地、或在图像捕获设备进行图像捕获和存储时提供DNN数据。因此,本文公开的图像捕获设备可以提供图像和颜色信息以及深度机器学习信息,如使用CNN的特征检测。

[0203] 图14图示了将RGB和CNN通道数据存储为深度通道图像的一部分的示例性图示。示例性系统被示出为具有深度通道相机1400和深度通道存储器1420。深度通道相机1400包括颜色传感器1402和FPGA单元1404。颜色传感器1402包括以阵列布置的多个传感器,以捕获图像1403A的颜色数据。图像1403A可以是针对RGB图像或YUV图像。颜色传感器1402将图像值或数据输出到深度通道存储器1420和FPGA单元1404。

[0204] 颜色传感器可以是任何类型的光传感器、光检测器、CMOS传感器或具有光传感器阵列的像素阵列,以检测形成由颜色传感器阵列捕获的彩色图像的光能。可替代地,颜色传感器可以被配置成检测非彩色图像。每个颜色分量的数据或值形成单独的输出通道,如RGB颜色通道1414。

[0205] 在一个实施例中,颜色传感器阵列1402的输出是到被编程在FPGA 1404中的神经网络1406的输入。网络可以是任何类型的网络,其具有可以执行神经网络处理的任何数量的卷积层和每层节点,如关于图9A和9B的示例性实施例中公开的。在其他示例中,可以使用硬接线电路或可编程逻辑阵列(PLA)来代替FPGA 1404,其中神经网络以相同的方式被编程或硬接线在相应的电路或阵列中。在其他实施例中,FPGA 1404可以被编程有如图6中关于机器学习应用602、机器学习框架604和计算框架606所述的硬件加速。

[0206] 在一个实施例中,FPGA单元1404被编程有标准超深CNN,如标记为标准VGG网络1406的视觉几何组(VGG)CNN。在其他实施例中,可以在如本文公开的FPGA单元1404中编程其他类型的CNN,如DNN。在该示例中,VGG网络1406被编程有标记为“Conv1”至“Conv5”的五个卷积层。每个卷积层Conv2至Conv5包括一“池”数据,从而提供附近输出的汇总统计。在其他实施例中,可以使用任何数量的卷积层,包括单个卷积层。每个Conv1至Conv5层执行卷积运算,该运算提供标记为“Conv1的数据”至“Conv5的数据”的经滤波的图像或特征图数据。在该前馈CNN网络中,每个卷积层的输出传递到下一层直到最终预测层1425。在预测层1425处,可以进行分类1430,诸如基于CNN处理和CNN数据来确定图像的类型。

[0207] 深度通道存储器1420存储从图像1403A导出的用于RGB通道1414的图像1403B。Conv1(1410A)至Conv5(1410E)的数据被存储用于CNN通道1418。在该实施例中,每个卷积通道Conv1至Conv5表示具有用于CNN通道1418的五个通道的CNN通道。Conv1(1410A)至Conv5(1410E)的数据可以是图像1403A的不同的滤波图像或特征图,这取决于在每一层执行的卷积处理。在深度通道存储器1420中,将用于RGB颜色通道1414和CNN通道的图像存储在相邻的存储器中或者将它们连结在一起。

[0208] 多处理器平台上的DNN执行流水线

[0209] 如在图12和14的上下文中所述,深度学习网络具有多个节点。图15是具有标记为1至6的六个节点的深度学习神经网络1502的示例的框图。虽然示出了六个节点,但这是为了图示的目的。可能有多得多的节点,这取决于具体实现。节点之间的连接也可能比示例中所示的更复杂。在许多深度学习网络中,原始网络是非循环图。在此示例中,数据流仅遵循一个方向,被示为从左到右。

[0210] 如本文所述,流水线框架能够以比用任务自动调度系统更高效的方式在多核平台

上运行深度学习网络。在所描述的多处理器或线程系统中,通过在处理资源之间平衡工作负荷来提高效率。这防止了一个处理资源中的积压、等待时间或延迟影响整个工作流。如本文所述,使用网络节点中的计算分发来分发工作流。使用两个部件来增强流水线框架。第一个部件是网络工作负荷分析器1708,并且第二个部件是网络执行器1712,如图17中所示。

[0211] 图16是由网络分析器对网络节点进行分组的图示。将相同的六个网络节点1-6聚集成三组,标记为组1、组2、组3。这些组对应于如下所述的处理资源。假设目标平台具有N个核,则在给定深度学习网络的情况下,网络工作负荷分析器分析计算分发并将网络节点正确地分组为N个组。进行分组以使每个组具有几乎相同的计算量。

[0212] 可以使用其输入数据的大小来精确地确定已知网络的计算量。深度学习网络的每个节点执行已知的深度学习操作,如卷积、批量归一化、softmax等,因此可以基于第一节点的输入大小精确地计算每个节点的计算量。同时,深度学习网络具有以下惰性(noble)特征,即一旦确定了网络的输入大小,那么就可以确定所有网络节点的输入大小,因为它们都是其输入大小和输出大小由标准公式来确定的标准运算。

[0213] 因此,对于任何特定网络和特定任务或工作负荷类型,都可以确定每个节点的计算复杂度。例如,第一节点的计算复杂度可以是5,第二节点是3,并且第三节点是2,这在本文中称为计算分发。在这个简单的三节点情况下,可以有两个组,组1仅具有第一节点,组2具有第二和第三节点。由于第二和第三节点的计算复杂度分别为3和2,因此组2的总复杂度为 $3+2=5$ 。这与组1相同,因此网络分析器的组策略遵循计算分发测量的指令。

[0214] 在所图示的示例中,存在三个组,每个组用于三个处理核中的每一个,其中 $N=3$ 。第一组具有网络节点1和2,第二组具有网络节点3和4,并且第三组具有网络节点5和6。由于原始网络是非循环图,并且数据流仅遵循一个方向,因此数据流以相同的方向穿过每个组内以及各组之间。

[0215] 在该示例中,由网络分析器在各组之间均匀地分发网络节点,使得每个组中具有相同数量的节点。在许多情况下,这可能是最佳或几乎最佳的解决方案。在其他情况下,可能存在一些节点需要比其他节点更多的计算。对于这样的网络,一些组中可以有比其他组中更少的节点。分组并非是基于节点数而进行的,而是基于计算分发而进行的。

[0216] 另外,将节点顺序分发至每个组。如图所示,节点1的输入去往第一组。节点1的输出去往节点2,然后去往第二组。分组建立了按顺序从节点1到节点6以及按顺序从组1到组3的处理流方向。这提供了流过网络的更顺畅的数据流,然而,对于一些网络,节点可能不是顺序分发的。

[0217] 网络工作负荷分析器1708可以是多核处理器处的部件、线程、模块或例程,其在接收到工作负荷时实时分析传入工作负荷。这允许修改分组以适应不同的工作负荷。在一些应用中,工作负荷是视频图像序列。该序列的图像可以被标准化为固定大小。该大小可以是捕获大小或经缩减或缩小的大小。在每个图像的大小相同的情况下,每个图像的计算分发将大致相同。在这种情况下,可以进行一次分组并然后将其固定,直到接收到不同尺寸的图像或不同类型的图像作为工作负荷。均匀地分发计算降低或避免了工作流中的空隙。

[0218] 图17是具有网络分析器1708和网络执行器1712的多核处理系统的图示。在对节点进行分组之后,然后如图所示,由网络执行器将每个组指派给相应的处理器,该处理器尤其处理针对该组的计算任务。处理器可以是多核管芯的核。处理器也可以包括多线程系统的

线程。处理器还可以是在不止一个管芯上的处理核。在将各组指派给各处理器的情况下,对于连续的图像序列,这N个处理器形成流水线。如果单个核针对一个图像本应花费时间t,那么具有N个核的流水线仅需要时间t/N。

[0219] 如图所示,多核处理器1712具有三个处理核。将三个组中的一个指派给每个核。每个核还耦合到存储器1714。如图所示,将工作负荷施加于存储器或通过输入缓冲器1704。输入缓冲器可以位于存储器1714的所分配的部分中。还将输入施加于网络工作负荷分析器1708,以用于确定计算分发。例如在该图中,如图所示,网络分析器将网络节点分发到各个核。

[0220] 在网络工作负荷分析器对节点进行分组之后,然后将开始网络执行配置阶段。网络执行器将每个组指派给特定的处理器或核,并为每个处理器分配一个输出存储器块,在该示例中被标识为P1、P2和P3。然后,下一个处理器或核访问该输出存储器以执行一系列阶段中的下一阶段。网络执行器还可以为输入工作负荷分配存储器块1704。这可以是来自应用1702或其他源的输出。所述应用或其他源然后被命令或配置以将新数据写入到所分配的输入存储器中。

[0221] 在该示例中,执行器还命令三个处理器1、2、3中的每一个,其中,应使用所分配的存储器空间P1、P2、P3来读取和写入相应的输入和输出。在该示例中,执行组1的节点的处理器1从输入存储器缓冲器1704读取其输入,并将其输出写入到输出存储器P1的所配置的部分。执行组2的节点的处理器2从P1读取其输入并将其输出写入输出存储器P2。处理器3从P2读取其输入并将其输出写入输出存储器P3。在DNN的有序图的每种情况下,一个处理器或组的输出用作下一个处理器或组的输入。网络的最终结果由处理器3写入P3。针对P3的存储器分配可以与输出缓冲器1706相同或不同。输出缓冲器将结果提供给其他下游过程。

[0222] 在通过网络执行器进行配置之后,执行器开始执行。应用1702或其他源连续地生成输入到输入缓冲器或存储器1704中,并且整个流水线保持运行并针对每个输入数据片(patch)连续地将对应输出写入到P3和输出缓冲器1706。

[0223] 每个处理核为其被指派的网络节点执行计算,并然后将结果存储在存储器中。第二和第三处理核从存储器获得它们的输入并将中间结果和输出存储在存储器中。提供最终结果作为来自多核处理器的输出。多核处理器可以具有多得多的核,它们被指派给不同的任务,或者被指派给比图中所示的三个节点更多的深度神经网络的节点。

[0224] 图18是由如上所述的分析器和执行器执行的操作的过程流程图。在1822处,接收工作负荷以供流水线框架在深度学习网络上运行。流水线框架具有例如在多核平台上的多个处理资源。深度学习网络具有多个节点。

[0225] 在1824处,网络工作负荷分析器分析工作负荷跨网络节点的计算分发。在1826处,网络工作负荷分析器将网络节点分组成多组。该分组是基于计算分发,以便跨各组均匀地分发工作负荷。在1828处,网络执行器或其他部件将每个组指派给处理核或多核平台的其他类型的资源。

[0226] 在配置了网络节点之后,在1830处,网络执行器执行工作负荷。每个相应的处理核处理相应的网络节点组的接收到的工作负荷的计算任务。在1832处,然后可以将输出应用于另一任务,如检测、识别和跟踪技术、自动语音识别、用户认证、图像理解和机器视觉等。

[0227] 图形系统概述

[0228] 图19是根据示例性实施例的处理系统1900的框图。在各种实施例中,系统1900包括一个或多个处理器1902以及一个或多个图形处理器1908,并且可以是单处理器台式系统、多处理器工作站系统或具有大量处理器1902或处理器核1907的服务器系统。在一个实施例中,系统1900是被并入到用于在移动设备、手持式设备或嵌入式设备中使用的片上系统(SoC)集成电路内的处理平台。

[0229] 系统1900的实施例可以包括下述各项或被并入到下述各项内: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1900是移动电话、智能电话、平板计算设备或移动互联网设备。数据处理系统1900还可以包括可穿戴设备(诸如智能手表可穿戴设备、智能眼镜设备、增强现实设备、或虚拟现实设备)、与所述可穿戴设备耦合、或者集成在所述可穿戴设备内。在一些实施例中,数据处理系统1900是电视或机顶盒设备,所述电视或机顶盒设备具有一个或多个处理器1902以及由一个或多个图形处理器1908生成的图形界面。

[0230] 在一些实施例中,一个或多个处理器1902各自包括用于处理指令的一个或多个处理器核1907,所述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1907中的每个处理器核被配置成处理特定的指令集1909。在一些实施例中,指令集1909可以促进复杂指令集计算(CISC)、精简指令集计算(RISC)、或经由甚长指令字(VLIW)的计算。多个处理器核1907可以各自处理不同的指令集1909,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1907还可以包括其他处理设备,诸如数字信号处理器(DSP)。

[0231] 在一些实施例中,处理器1902包括高速缓存存储器1904。取决于架构,处理器1902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1902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1902还使用外部高速缓存(例如,3级(L3)高速缓存或未级高速缓存(LLC))(未示出),可以使用已知的高速缓存一致性技术来在处理器核1907当中共享该外部高速缓存。另外,寄存器堆1906包括在处理器1902中,所述处理器可以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,而其他寄存器可以特定于处理器1902的设计。

[0232] 在一些实施例中,处理器1902与处理器总线1910耦合,所述处理器总线1910用于在处理器1902与系统1900中的其他部件之间传输通信信号,诸如地址、数据、或控制信号。在一个实施例中,系统1900使用示例性“中枢”系统架构,包括存储器控制器中枢1916和输入输出(I/O)控制器中枢1930。存储器控制器中枢1916促进存储器设备与系统1900的其他部件之间的通信,而I/O控制器中枢(ICH)1930经由本地I/O总线提供到I/O设备的连接。在一个实施例中,存储器控制器中枢1916的逻辑集成在处理器内。

[0233] 存储器设备1920可以是动态随机存取存储器(DRAM)设备、静态随机存取存储器(SRAM)设备、闪存设备、相变存储器设备、或具有合适的性能以充当处理存储器的某个其他存储器设备。在一个实施例中,存储器设备1920可以作为系统1900的系统存储器进行操作,以存储数据1922和指令1921,以供在一个或多个处理器1902执行应用或进程时使用。存储器控制器中枢1916还与可选的外部图形处理器1912耦合,所述可选的外部图形处理器可以与处理器1902中的一个或多个图形处理器1908通信,从而执行图形和媒体操作。

[0234] 在一些实施例中, ICH 1930使得外设能够经由高速I/O总线连接至存储器设备1920和处理器1902。I/O外设包括但不限于: 音频控制器1946、固件接口1928、无线收发机1926(例如, Wi-Fi、蓝牙)、数据存储设备1924(例如, 硬盘驱动器、闪存等)、以及用于将传统(例如, 个人系统2 (PS/2))设备耦合至所述系统的传统I/O控制器1940。一个或多个通用串行总线(USB)控制器1942连接输入设备, 诸如键盘和鼠标的组合1944。网络控制器1934还可以与ICH 1930耦合。在一些实施例中, 高性能网络控制器(未示出)与处理器总线1910耦合。应当理解, 所示出的系统1900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1930可以集成在一个或多个处理器1902内, 或者存储器控制器中枢1916和I/O控制器中枢1930可以集成到分立式外部图形处理器(诸如外部图形处理器1912)中。

[0235] 图20是处理器2000的示例性实施例的框图, 所述处理器具有一个或多个处理器核2002A至2002N、集成存储器控制器2014、以及集成图形处理器2008。图20的具有与本文中任何其他附图的元件相同的附图标记(或名称)的那些元件可以采用与在本文中其他地方描述的方式类似的任何方式进行操作或起作用, 但不限于这些。处理器2000可以包括直到且包括由虚线框表示的附加核2002N的附加核。处理器核2002A至2002N中的每一个包括一个或多个内部高速缓存单元2004A至2004N。在一些实施例中, 每个处理器核还可访问一个或多个共享的高速缓存单元2006。

[0236] 内部高速缓存单元2004A至2004N和共享高速缓存单元2006表示处理器2000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以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一级或多级共享中级高速缓存, 诸如2级(L2)、3级(L3)、4级(L4)、或其他级的高速缓存, 其中, 外部存储器前的最高级的高速缓存被分类为LLC。在一些实施例中, 高速缓存一致性逻辑维持各种高速缓存单元2006与2004A至2004N之间的一致性。

[0237] 在一些实施例中, 处理器2000还可以包括一组一个或多个总线控制器单元2016和系统代理核2010。一个或多个总线控制器单元2016管理一组外围总线, 诸如一个或多个外围部件互连总线(例如, PCI、PCI Express)。系统代理核2010提供对各种处理器部件的管理功能。在一些实施例中, 系统代理核2010包括一个或多个集成存储器控制器2014以管理对各种外部存储器设备(未示出)的访问。

[0238] 在一些实施例中, 处理器核2002A至2002N中的一个或多个包括对同时多线程的支持。在这种实施例中, 系统代理核2010包括用于在多线程处理期间协调和操作核2002A至2002N的部件。另外, 系统代理核2010还可以包括功率控制单元(PCU), 所述功率控制单元包括用于调节处理器核2002A至2002N以及图形处理器2008的功率状态的逻辑和部件。

[0239] 在一些实施例中, 另外, 处理器2000还包括用于执行图形处理操作的图形处理器2008。在一些实施例中, 图形处理器2008与共享高速缓存单元2006集以及系统代理核2010耦合, 所述系统代理核包括一个或多个集成存储器控制器2014。在一些实施例中, 显示控制器2011与图形处理器2008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2011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2008或系统代理核2010内。

[0240] 在一些实施例中, 基于环的互连单元2012用于耦合处理器2000的内部部件。然而, 可以使用替代性互连单元, 诸如点到点互连、切换式互连、或其他技术, 包括本领域中众所

周知的技术。在一些实施例中,图形处理器2008经由I/O链路2013与环形互连2012耦合。

[0241] 示例性I/O链路2013表示I/O互连的多个品种中的至少一种,包括促进各种处理器部件与高性能嵌入式存储器模块2018(诸如eDRAM模块)之间的通信的封装体I/O互连。在一些实施例中,处理器核2002A至2002N中的每个处理器核以及图形处理器2008将嵌入式存储器模块2018用作共享末级高速缓存。

[0242] 在一些实施例中,处理器核2002A至2002N是执行相同指令集架构的均质核。在另一实施例中,处理器核2002A至2002N就指令集架构(ISA)而言是异构的,其中,处理器核2002A至2002N中的一个或多个执行第一指令集,而其他核中的至少一个执行所述第一指令集的子集或不同的指令集。在一个实施例中,处理器核2002A至2002N就微架构而言是异构的,其中,具有相对较高功耗的一个或多个核与具有较低功耗的一个或多个功率核耦合。另外,处理器2000可以被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图示的部件的SoC集成电路。

[0243] 图21是图形处理器2100的框图,所述图形处理器可以是分立式图形处理单元、或者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存储器映射I/O接口并且利用被放置在处理器存储器中的命令进行通信。在一些实施例中,图形处理器2100包括用于访问存储器的存储器接口2114。存储器接口2114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。

[0244] 在一些实施例中,图形处理器2100还包括显示控制器2102,所述显示控制器用于将显示输出数据驱动到显示设备2120。显示控制器2102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2100包括用于将媒体编码到一个或多个媒体编码格式、从一个或多个媒体编码格式解码媒体、或者在一个或多个媒体编码格式之间对媒体进行代码转换的视频编解码器引擎2106,该一个或多个媒体编码格式包括但不限于:运动图像专家组(MPEG)格式(诸如MPEG-2)、高级视频编码(AVC)格式(诸如H.264/MPEG-4AVC)、以及电影和电视工程师协会(SMPTE)421M/VC-1、和联合图像专家组(JPEG)格式(诸如JPEG)、和运动JPEG(MJPEG)格式。

[0245] 在一些实施例中,图形处理器2100包括用于执行二维(2D)栅格器操作(包括例如位边界块传送)的块图像传送(BLIT)引擎2104。然而,在一个实施例中,使用图形处理引擎(GPE)2110的一个或多个部件执行2D图形操作。在一些实施例中,GPE 2110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。

[0246] 在一些实施例中,GPE 2110包括用于执行3D操作的3D流水线2112,所述3D操作诸如是使用作用于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2112包括可编程且固定的功能元件,所述可编程且固定的功能元件在元件内执行各种任务和/或生成到3D/媒体子系统2115的执行线程。虽然3D流水线2112可以用于执行媒体操作,但是GPE 2110的实施例还包括媒体流水线2116,所述媒体流水线具体地用于执行媒体操作,诸如视频后处理和图像增强。

[0247] 在一些实施例中,媒体流水线2116包括固定功能或可编程逻辑单元以便代替或代表视频编解码器引擎2106来执行一个或多个专门的媒体操作,诸如视频解码加速、视频解交织、以及视频编码加速。在一些实施例中,另外,媒体流水线2116还包括线程生成单元以

便生成用于在3D/媒体子系统2115上执行的线程。所生成的线程对3D/媒体子系统2115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。

[0248] 在一些实施例中,3D/媒体子系统2115包括用于执行3D流水线2112和媒体流水线2116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2115发送线程执行请求,所述3D/媒体子系统包括用于仲裁各种请求并将各种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阵列。在一些实施例中,3D/媒体子系统2115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。

[0249] 图形处理引擎

[0250] 图22是根据一些实施例的图形处理器的图形处理引擎2210的框图。在一个实施例中,图形处理引擎(GPE)2210是图21中所示的GPE 2110的版本。图22的具有与本文中任何其他附图的元件相同的附图标记(或名称)的元件可以采用与在本文中其他地方描述的方式类似的任何方式进行操作或起作用,但不限于这些。例如,图示了图21的3D流水线2112和媒体流水线2116。媒体流水线2116在GPE 2210的一些实施例中是可选的,并且可以不显式地包括在GPE 2210内。例如以及在至少一个实施例中,单独的媒体和/或图像处理器耦合至GPE 2210。

[0251] 在一些实施例中,GPE 2210与命令流送器2203耦合或包括命令流送器2203,所述命令流送器向3D流水线2112和/或媒体流水线2116提供命令流。在一些实施例中,命令流送器2203与存储器耦合,所述存储器可以是系统存储器、或者内部高速缓存存储器和共享高速缓存存储器中的一个或多个高速缓存存储器。在一些实施例中,命令流送器2203从存储器接收命令并将这些命令发送至3D流水线2112和/或媒体流水线2116。所述命令是从存储器用于3D流水线2112和媒体流水线2116的命令的环形缓冲器获取的指示。在一个实施例中,另外,环形缓冲器还可以包括存储多批多个命令的批命令缓冲器。用于3D流水线2112的命令还可以包括对在存储器中存储的数据的引用,该数据诸如但不限于用于3D流水线2112的顶点和几何数据2230和/或用于媒体流水线2116的图像数据和存储器对象2232。3D流水线2112和媒体流水线2116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图形核阵列2214来处理命令和数据。

[0252] 在各种实施例中,3D流水线2112可以通过处理指令并将执行线程分派给图形核阵列2214来执行一个或多个着色器程序,诸如顶点着色器、几何着色器、像素着色器、片段着色器、计算着色器或其他着色器程序。图形核阵列2214提供统一的执行资源块。图形核阵列2214内的多用途执行逻辑(例如,执行单元)包括对各种3D API着色器语言的支持,并且可以执行与多个着色器相关联的多个同时执行线程。

[0253] 在一些实施例中,图形核阵列2214还包括用于执行诸如视频和/或图像处理之类的媒体功能的执行逻辑。在一个实施例中,执行单元还包括可编程以除图形处理操作外还执行并行通用计算操作的通用逻辑。通用逻辑可以与图19的(多个)处理器核1907或如图20中的核2002A至2002N内的通用逻辑并行地或结合地执行处理操作。

[0254] 由在图形核阵列2214上执行的线程生成的输出数据可以将数据输出到统一返回缓冲器(URB)2218中的存储器。URB 2218可以存储多个线程的数据。在一些实施例中,URB

2218可以用于在图形核阵列2214上执行的不同线程之间发送数据。在一些实施例中,URB 2218可以另外用于图形核阵列上的线程与共享功能逻辑2220内的固定功能逻辑之间的同步。

[0255] 在一些实施例中,图形核阵列2214是可缩放的,使得所述阵列包括可变数量的图形核,这些图形核各自具有基于GPE 2210的目标功率和性能等级的可变数量的执行单元。在一个实施例中,执行资源是动态可缩放的,从而可以根据需要启用或禁用执行资源。

[0256] 图形核阵列2214与共享功能逻辑2220耦合,所述共享功能逻辑包括在图形核阵列中的图形核之间共享的多个资源。共享功能逻辑2220内的共享功能是向图形核阵列2214提供专用补充功能的硬件逻辑单元。在各种实施例中,共享功能逻辑2220包括但不限于采样器2221、数学2222和线程间通信(ITC) 2223逻辑。另外,一些实施例实现共享功能逻辑2220内的一个或多个高速缓存2225。在针对给定专用功能的需求不足以包括在图形核阵列2214内的情况下实现共享功能。取而代之,该专用功能的单个实例化被实现为共享功能逻辑2220中的独立实体并且在图形核阵列2214内的执行资源之间共享。在图形核阵列2214之间共享并包括在图形核阵列2214内的精确的一组功能在实施例之间变化。

[0257] 图23是图形处理器2300的另一示例性实施例的框图。图23的具有与本文中任何其他附图的元件相同的附图标记(或名称)的元件可以采用与在本文中其他地方描述的方式类似的任何方式进行操作或起作用,但不限于这些。

[0258] 在一些实施例中,图形处理器2300包括环形互连2302、流水线前端2304、媒体引擎2337、以及图形核2380A至2380N。在一些实施例中,环形互连2302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许多处理器之一。

[0259] 在一些实施例中,图形处理器2300经由环形互连2302接收多批命令。传入命令由流水线前端2304中的命令流送器2303来解译。在一些实施例中,图形处理器2300包括用于经由(多个)图形核2380A至2380N执行3D几何处理和媒体处理的可缩放执行逻辑。对于3D几何处理命令,命令流送器2303将命令供应至几何流水线2336。针对至少一些媒体处理命令,命令流送器2303将命令供应至视频前端2334,所述视频前端与媒体引擎2337耦合。在一些实施例中,媒体引擎2337包括用于视频和图像后处理的视频质量引擎(VQE) 2330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) 2333引擎。在一些实施例中,几何流水线2336和媒体引擎2337各自生成执行线程,所述执行线程用于由至少一个图形核2380A提供的线程执行资源。

[0260] 在一些实施例中,图形处理器2300包括可缩放线程执行资源表征模块化核2380A至2380N(有时被称为核分片),每一个可缩放线程执行资源表征模块化核具有多个子核2350A至2350N、2360A至2360N(有时被称为核子分片)。在一些实施例中,图形处理器2300可以具有任何数量的图形核2380A至2380N。在一些实施例中,图形处理器2300包括图形核2380A,所述图形核2380A至少具有第一子核2350A和第二子核2360A。在其他实施例中,图形处理器是具有单个子核(例如,2350A)的低功率处理器。在一些实施例中,图形处理器2300包括多个图形核2380A至2380N,所述图形核各自包括一组第一子核2350A至2350N和一组第二子核2360A至2360N。所述一组第一子核2350A至2350N中的每个子核至少包括第一组执行单元2352A至2352N和媒体/纹理采样器2354A至2354N。所述一组第二子核2360A至2360N中

的每个子核至少包括第二组执行单元2362A至2362N和采样器2364A至2364N。在一些实施例中,每个子核2350A至2350N、2360A至2360N共享一组共享资源2370A至2370N。在一些实施例中,所述共享资源包括共享高速缓存存储器和像素操作逻辑。其他共享资源也可以包括在图形处理器的各种实施例中。

[0261] 执行单元

[0262] 图24图示了线程执行逻辑2400,所述线程执行逻辑包括在GPE的一些示例性实施例中采用的处理元件阵列。图24的具有与本文中任何其他附图的元件相同的附图标记(或名称)的元件可以采用与在本文中其他地方描述的方式类似的任何方式进行操作或起作用,但不限于这些。

[0263] 在一些实施例中,线程执行逻辑2400包括着色器处理器2402、线程分派器2404、指令高速缓存2406、包括多个执行单元2408A至2408N的可缩放执行单元阵列、采样器2410、数据高速缓存2412、以及数据端口2414。在一个实施例中,可缩放执行单元阵列可以通过基于工作负荷的计算要求启用或禁用一个或多个执行单元(例如,执行单元2408A、2408B、2408C、2408D一直到2408N-1和2408N中的任一个)来动态地进行缩放。在一个实施例中,所包括的部件经由互连结构而互连,所述互连结构链接到部件中的每个部件。在一些实施例中,线程执行逻辑2400包括通过指令高速缓存2406、数据端口2414、采样器2410、以及执行单元2408A至2408N中的一个或多个而到存储器(诸如系统存储器或高速缓存存储器)的一个或多个连接件。在一些实施例中,每个执行单元(例如,2408A)是能够执行多个同时硬件线程而同时针对每个线程并行地处理多个数据元素的独立可编程通用计算单元。在各种实施例中,执行单元阵列2408A至2408N可缩放以包括任意数量的单独执行单元。

[0264] 在一些实施例中,执行单元2408A至2408N主要用于执行着色器程序。着色器处理器2402可以处理各种着色器程序并经由线程分派器2404来分派与着色器程序相关联的执行线程。在一个实施例中,线程分派器包括对来自图形和媒体流水线的线程发起请求进行仲裁且在执行单元2408A至2408N中的一个或多个执行单元上对所请求的线程进行实例化的逻辑。例如,几何流水线(例如,图23的2336)可以将顶点、曲面细分或几何着色器分派至线程执行逻辑2400(图24)以用于处理。在一些实施例中,线程分派器2404还可以处理来自执行着色器程序的运行时线程生成请求。

[0265] 在一些实施例中,执行单元2408A至2408N支持指令集(所述指令集包括对许多标准3D图形着色器指令的本机支持),使得以最小的转换执行来自图形库(例如,Direct 3D和OpenGL)的着色器程序。这些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)以及通用处理(例如,计算和媒体着色器)。执行单元2408A至2408N中的每一个能够进行多发布单指令多数据(SIMD)执行,并且多线程操作在面对较高等待时间存储器访问时实现高效执行环境。每个执行单元内的每个硬件线程具有专用高带宽寄存器堆和相关联的独立线程状态。对能够进行整数、单精度和双精度浮点运算、SIMD分支能力、逻辑运算、超越数运算和其他杂项运算的流水线来说,执行是每时钟多发布的。在等待来自存储器或者共享功能之一的数据的同时,执行单元2408A至2408N内的依赖性逻辑使等待线程休眠直到所请求的数据已被返回为止。在等待线程正在休眠时,硬件资源可以致力于处理其他线程。例如,在与顶点着色器操作相关联的延迟期间,执行单元可以执行针对像素着色器、片段着色器或者包括不同顶点着色器的另一种类

型的着色器程序的操作。

[0266] 执行单元2408A至2408N中的每个执行单元在数据元素阵列上进行操作。数据元素的数量是“执行大小”、或指令的通道数。执行通道是针对数据元素访问、掩蔽、和指令内的流控制的执行的逻辑单元。通道的数量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) 的数量无关。在一些实施例中,执行单元2408A至2408N支持整数和浮点数据类型。

[0267] 执行单元指令集包括SIMD指令。各种数据元素可以作为压缩数据类型存储在寄存器中,并且执行单元将基于元素的数据大小来处理各种元素。例如,当在256位宽的向量上进行操作时,向量的256位存储在寄存器中,并且所述执行单元作为四个分离64位压缩数据元素(四倍字(QW)大小数据元素)、八个分离32位压缩数据元素(双倍字(DW)大小数据元素)、十六个分离16位压缩数据元素(字(W)大小数据元素)、或三十二个分离8位数据元素(字节(B)大小数据元素)在所述向量上进行操作。然而,不同向量宽度和寄存器大小是可能的。

[0268] 一个或多个内部指令高速缓存(例如,2406)被包括在所述线程执行逻辑2400中以便高速缓存所述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,2412)被包括以在线程执行期间高速缓存线程数据。在一些实施例中,采样器2410被包括以为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2410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。

[0269] 在执行期间,图形和媒体流水线经由线程生成和分派逻辑向线程执行逻辑2400发送线程发起请求。一旦一组几何对象已经被处理并被栅格化成像素数据,则着色器处理器2402内的像素处理器逻辑(例如像素着色器逻辑、片段着色器逻辑等)被调用以便进一步计算输出信息并且使得结果被写入到输出表面(例如,色彩缓冲器、深度缓冲器、模板印刷缓冲器等)。在一些实施例中,像素着色器或片段着色器计算各种顶点属性的值,所述各种顶点属性要跨栅格化对象而被内插。在一些实施例中,着色器处理器2402内的像素处理器逻辑然后执行应用编程接口(API)供应的像素或片段着色器程序。为了执行着色器程序,着色器处理器2402经由线程分派器2404将线程分派至执行单元(例如,2408A)。在一些实施例中,像素着色器2402使用采样器2410中的纹理采样逻辑来访问存储器中所存储的纹理映射中的纹理数据。对纹理数据和输入几何数据的算术运算计算每个几何片段的像素颜色数据,或丢弃一个或多个像素而不进行进一步处理。

[0270] 在一些实施例中,数据端口2414提供存储器访问机制,以供线程执行逻辑2400将经处理的数据输出至存储器以用于在图形处理器输出流水线上进行处理。在一些实施例中,数据端口2414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2412)以经由数据端口高速缓存数据以用于存储器访问。

[0271] 图25是图示了根据一些实施例的图形处理器指令格式2500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采用多种格式的指令的指令集。实线框图示了通常包括在执行单元指令中的部件,而虚线包括可选的部件或仅包括在指令子集中的部件。在一些实施例中,所描述和图示的指令格式2500是宏指令,因为它们是供应至执行单元的指令,这与一旦指令被处理而由指令解码产生的微操作形成对照。

[0272] 在一些实施例中,图形处理器执行单元本地支持采用128位指令格式2510的指令。64位压缩指令格式2530可用于基于所选指令、指令选项和操作数数量的一些指令。本机128位指令格式2510提供对所有指令选项的访问,而一些选项和操作被限制在64位指令格式2530中。64位指令格式2530中可用的本机指令根据实施例而不同。在一些实施例中,部分地使用索引字段2513中的一组索引值来压缩指令。执行单元硬件基于索引值来参考一组压缩表,并使用压缩表输出来重构采用128位指令格式2510的本机指令。

[0273] 针对每种格式,指令操作码2512定义执行单元要执行的操作。执行单元跨每个操作数的多个数据元素来并行地执行每条指令。例如,响应于加指令,执行单元跨每个颜色通道执行同时加操作,所述颜色通道表示纹理元素或图片元素。默认地,执行单元跨操作数的所有数据通道执行每条指令。在一些实施例中,指令控制字段2514启用对某些执行选项(诸如通道选择(例如,预测)以及数据通道次序(例如,搅和))的控制。针对128位指令格式2510中的指令,执行大小字段2516限制了将并行执行的数据通道的数量。在一些实施例中,执行大小字段2516不可用于在64位压缩指令格式2530中使用。

[0274] 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(源0 2520、源1 2522)和一个目的地2518。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含的。数据操控指令可以具有第三源操作数(例如,源2 2524),其中,指令操作码2512确定源操作数的数量。指令的最后的源操作数可以是利用指令传递的立即(例如,硬编码)值。

[0275] 在一些实施例中,128位指令格式2510包括访问/寻址模式字段2526,所述访问/寻址模式字段2526例如指定了是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,直接由指令中的位来提供一个或多个操作数的寄存器地址。

[0276] 在一些实施例中,128位指令格式2510包括访问/寻址模式字段2526,所述访问/寻址模式字段2526指定针对指令的寻址模式和/或访问模式。在一个实施例中,访问模式用于定义针对指令的数据访问对齐。一些实施例支持访问模式,包括16字节对齐访问模式和1字节对齐访问模式,其中,访问模式的字节对齐确定了指令操作数的访问对齐。例如,当在第一模式中时,指令可以使用字节对齐寻址以用于源操作数和目的地操作数,并且当在第二模式中时,指令可以使用16字节对齐寻址以用于所有的源操作数和目的地操作数。

[0277] 在一个实施例中,访问/寻址模式字段2526的寻址模式部分确定指令是要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,可以基于指令中的地址寄存器值和地址立即字段来计算一个或多个操作数的寄存器地址。

[0278] 在一些实施例中,基于操作码2512位字段对指令进行分组以简化操作码解码2540。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。在一些实施例中,移动和逻辑操作码组2542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2542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2544(例如,调用(call)、跳(jmp))包括采用0010xxxxb(例如,0x20)形式的指令。杂项指令组2546包括指令的混合,包括采用0011xxxxb(例如,0x30)形式的同步指令(例如,等待、发送)。并行数学指令组2548包括采用0100xxxxb(例如,0x40)形式的逐分量的算术指令(例如,加、乘(mul))。并行数学组2548跨数据通道并行地执行算术运算。向量数学组2550包括

采用0101xxxxb(例如,0x50)形式的算术指令(例如,dp4)。向量数学组对向量操作数执行算术计算,诸如点积计算。

[0279] 图形流水线

[0280] 图26是图形处理器2600的另一实施例的框图。图26的具有与本文中任何其他附图的元件相同的附图标记(或名称)的元件可以采用与在本文中其他地方描述的方式类似的任何方式进行操作或起作用,但不限于这些。

[0281] 在一些实施例中,图形处理器2600包括图形流水线2620、媒体流水线2630、显示引擎2640、线程执行逻辑2650、以及渲染输出流水线2670。在一些实施例中,图形处理器2600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通过环形互连2602而发布至图形处理器2600的命令而被控制。在一些实施例中,环形互连2602将图形处理器2600耦合至其他处理部件,诸如其他图形处理器或通用处理器。来自环形互连2602的命令由命令流送器2603解译,所述命令流送器将指令供应至图形流水线2620或媒体流水线2630的单独部件。

[0282] 在一些实施例中,命令流送器2603引导顶点获取器2605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送器2603所提供的顶点处理命令。在一些实施例中,顶点获取器2605将顶点数据提供给顶点着色器2607,所述顶点着色器对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2605和顶点着色器2607通过经由线程分派器2631向执行单元2652A至2652B分派执行线程来执行顶点处理指令。

[0283] 在一些实施例中,执行单元2652A至2652B是具有用于执行图形和媒体操作的指令集的向量处理器阵列。在一些实施例中,执行单元2652A至2652B具有附接的L1高速缓存2651,所述L1高速缓存专用于每个阵列或在阵列之间共享。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区为包含不同分区中的数据和指令。

[0284] 在一些实施例中,图形流水线2620包括用于执行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2611配置曲面细分操作。可编程域着色器2617提供对曲面细分输出的后端评估。镶嵌器2613在外壳着色器2611的方向处进行操作并且包含专用逻辑,所述专用逻辑用于基于粗几何模型来生成详细几何对象的集合,所述粗几何模型作为输入而被提供至图形流水线2620。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(例如,外壳着色器2611、镶嵌器2613和域着色器2617)进行旁路。

[0285] 在一些实施例中,完整的几何对象可以由几何着色器2619经由被分派至执行单元2652A至2652B的一个或多个线程来处理、或者直接行进至裁剪器2629。在一些实施例中,几何着色器在整个几何对象(而非如图形流水线的先前级中的顶点或顶点补丁)上进行操作。如果禁用曲面细分,则几何着色器2619从顶点着色器2607接收输入。在一些实施例中,几何着色器2619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。

[0286] 在栅格化之前,裁剪器2629处理顶点数据。裁剪器2629可以是固定功能的裁剪器或者具有裁剪和几何着色器功能的可编程裁剪器。在一些实施例中,渲染输出流水线2670中的栅格器和深度测试部件2673分派像素着色器以将几何对象转换成其每像素表示。在一些实施例中,像素着色器逻辑包括在线程执行逻辑2650中。在一些实施例中,应用可以对栅

格器和深度测试部件2673进行旁路并且经由流出单元2623访问未栅格化的顶点数据。

[0287] 图形处理器2600具有互连总线、互连结构、或某个其他的互连机构,所述互连机构允许数据和消息在处理器的主要部件之间传递。在一些实施例中,执行单元2652A至2652B和(多个)相关联的高速缓存2651、纹理和媒体采样器2654、以及纹理/采样器高速缓存2658经由数据端口2656进行互连,以便执行存储器访问并且与处理器的渲染输出流水线部件进行通信。在一些实施例中,采样器2654、高速缓存2651、2658以及执行单元2652A至2652B各自具有单独的存储器访问路径。

[0288] 在一些实施例中,渲染输出流水线2670包含栅格器和深度测试部件2673,所述栅格器和深度测试部件将基于顶点的对象转换为相关联的基于像素的表示。在一些实施例中,栅格器逻辑包括用于执行固定功能三角形和线栅格化的窗口器/掩蔽器单元。相关联的渲染高速缓存2678和深度高速缓存2679在一些实施例中也是可用的。像素操作部件2677对数据执行基于像素的操作,然而在一些实例中,与2D操作(例如,利用混合的位块图像传送)相关联的像素操作由2D引擎2641执行、或者在显示时间处由显示控制器2643使用重叠显示平面来代替。在一些实施例中,共享的L3高速缓存2675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。

[0289] 在一些实施例中,图形处理器媒体流水线2630包括媒体引擎2637和视频前端2634。在一些实施例中,视频前端2634从命令流送器2603接收流水线命令。在一些实施例中,媒体流水线2630包括单独的命令流送器。在一些实施例中,视频前端2634在将所述命令发送至媒体引擎2637之前处理媒体命令。在一些实施例中,媒体引擎2637包括用于生成线程以用于经由线程分派器2631分派至线程执行逻辑2650的线程生成功能。

[0290] 在一些实施例中,图形处理器2600包括显示引擎2640。在一些实施例中,显示引擎2640在处理器2600外部并且经由环形互连2602、或某个其他互连总线或结构与图形处理器耦合。在一些实施例中,显示引擎2640包括2D引擎2641和显示控制器2643。在一些实施例中,显示引擎2640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2643与显示设备(未示出)耦合,所述显示设备可以是系统集成显示设备(如在膝上型计算机中)、或者经由显示设备连接器外接的外部显示设备。

[0291] 在一些实施例中,图形流水线2620和媒体流水线2630可配置成基于多个图形和媒体编程接口执行操作并且并非专用于任何一种应用编程接口(API)。在一些实施例中,图形处理器的驱动器软件将专用于特定图形或媒体库的API调用转换成可由图形处理器处理的命令。在一些实施例中,为全部来自Khronos Group的开放图形库(OpenGL)、开放计算语言(OpenCL)和/或Vulkan图形和计算API提供支持。在一些实施例中,还可以为来自微软公司的Direct3D库提供支持。在一些实施例中,可以支持这些库的组合。还可以为开源计算机视觉库(OpenCV)提供支持。如果可以做出从未来API的流水线到图形处理器的流水线的映射,则具有兼容3D流水线的未来API也将受到支持。

[0292] 图形流水线编程

[0293] 图27A是图示了根据一些实施例的图形处理器命令格式2700的框图。图27B是图示了根据实施例的图形处理器命令序列2710的框图。图27A中的实线框图示了通常包括在图形命令中的部件,而虚线包括可选的或者仅包括在所述图形命令的子集中的部件。图27A的示例性图形处理器命令格式2700包括用于标识命令的目标客户端2702、命令操作代码(操

作码) 2704、以及用于命令的相关数据2706的数据字段。一些命令中还包括子操作码2705和命令大小2708。

[0294] 在一些实施例中,客户端2702指定了处理命令数据的图形设备的客户端单元。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节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对命令进行处理的对应处理流水线。一旦命令被客户端单元接收到,客户端单元就读取操作码2704以及子操作码2705 (如果存在的话) 以确定要执行的操作。客户端单元使用数据字段2706中的信息来执行命令。针对一些命令,期望显式的命令大小2708来指定命令的大小。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,经由双倍字的倍数对命令进行对齐。

[0295] 图27B中的流程图示出了示例性图形处理器命令序列2710。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设置、执行并终止图形操作集合。仅出于示例目的示出并描述了样本命令序列,由于实施例不限于这些特定命令或者该命令序列。而且,所述命令可以作为命令序列中的一批命令而发布,使得图形处理器将以至少部分同时的方式处理命令序列。

[0296] 在一些实施例中,图形处理器命令序列2710可以以流水线转储清除命令2712开始以便使得任何活动图形流水线完成针对所述流水线的当前未决命令。在一些实施例中,3D流水线2722和媒体流水线2724不同时进行操作。执行流水线转储清除以使得活动图形流水线完成任何未决命令。响应于流水线转储清除,用于图形处理器的命令解析器将暂停命令处理直到活动绘画引擎完成未决操作并且使相关的读高速缓存失效。可选地,渲染高速缓存中被标记为“脏”的任何数据可以被转储清除到存储器。在一些实施例中,流水线转储清除命令2712可以用于流水线同步或者用在将图形处理器置于低功率状态中之前。

[0297] 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显式地切换时,使用流水线选择命令2713。在一些实施例中,在发布流水线命令之前在执行上下文内仅要求流水线选择命令2713一次,除非所述上下文要发布针对全部两条流水线的命令。在一些实施例中,在经由流水线选择命令2713进行的流水线切换之前立即需要流水线转储清除命令2712。

[0298] 在一些实施例中,流水线控制命令2714配置用于操作的图形流水线并且用于对3D流水线2722和媒体流水线2724进行编程。在一些实施例中,流水线控制命令2714配置活动流水线的流水线状态。在一个实施例中,流水线控制命令2714用于流水线同步并且用于在处理一批命令之前从活动流水线内的一个或多个高速缓存存储器中清除数据。

[0299] 在一些实施例中,用于返回缓冲器状态的命令2716用于配置返回缓冲器的集合以供相应的流水线写入数据。一些流水线操作要求分配、选择或配置一个或多个返回缓冲器,所述操作在处理期间将中间数据写入到所述一个或多个返回缓冲器中。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存储输出数据并且执行跨线程通信。在一些实施例中,配置返回缓冲器状态2716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。

[0300] 命令序列中的剩余命令基于用于操作的活动流水线而不同。基于流水线确定

2720,所述命令序列被定制到以3D流水线状态2730开始的3D流水线2722或者在媒体流水线状态2740处开始的媒体流水线2724。

[0301] 用于配置3D流水线状态2730的命令包括用于顶点缓冲器状态、顶点元素状态、常量颜色状态、深度缓冲器状态、以及要在处理3D图元命令之前配置的其他状态变量的3D状态设置命令。这些命令的值至少部分地基于使用中的特定3D API而确定。在一些实施例中,3D流水线状态2730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)。

[0302] 在一些实施例中,3D图元2732命令用于提交要由3D流水线处理的3D图元。经由3D图元2732命令传递给图形处理器的命令和相关联参数被转发到所述图形流水线中的顶点获取功能。顶点获取功能使用3D图元2732命令数据来生成顶点数据结构。所述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2732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2722将着色器执行线程分派至图形处理器执行单元。

[0303] 在一些实施例中,经由执行2734命令或事件触发3D流水线2722。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“去”(“go”)或“踢”(“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流水线将针对3D图元来执行几何处理。一旦完成操作,则对所产生的几何对象进行栅格化,并且像素引擎对所产生的像素进行着色。对于那些操作,还可以包括用于控制像素着色和像素后端操作的附加命令。

[0304] 在一些实施例中,当执行媒体操作时,图形处理器命令序列2710跟随在媒体流水线2724路径之后。一般地,针对媒体流水线2724进行编程的具体用途和方式取决于要执行的媒体或计算操作。在媒体解码期间,特定的媒体解码操作可以被卸载到所述媒体流水线。在一些实施例中,还可对媒体流水线进行旁路,并且可以整体地或部分地使用由一个或多个通用处理核提供的资源来执行媒体解码。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,所述图形处理器用于使用计算着色器程序来执行SIMD向量运算,所述计算着色器程序与渲染图形图元不是显式相关的。

[0305] 在一些实施例中,以与3D流水线2722类似的方式对媒体流水线2724进行配置。将用于配置媒体流水线状态的一组命令2740分派或放置到命令队列中媒体对象命令2742之前。在一些实施例中,媒体流水线状态命令2740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。这包括用于在媒体流水线内配置视频解码和视频编码逻辑的数据,诸如编码或解码格式。在一些实施例中,媒体流水线状态命令2740还支持使用指向包含一批状态设置的“间接”状态元件的一个或多个指针。

[0306] 在一些实施例中,媒体对象命令2742将指针供应至媒体对象以供媒体流水线处理。媒体对象包括存储器缓冲器,所述存储器缓冲器包含要处理的视频数据。在一些实施例中,在发布媒体对象命令2742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2742被排队,则经由执行命令2744或等效的执行事件(例如,寄存器写入)来触发媒体流水线2724。然后可以通过由3D流水线2722或媒体流水线2724提供的操作对来自媒体流水线2724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。

[0307] 图形软件架构

[0308] 图28图示了根据一些实施例的数据处理系统2800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2810、操作系统2820、以及至少一个处理器2830。在一些实施例中,处理器2830包括图形处理器2832以及一个或多个通用处理器核2834。图形应用2810和操作系统2820各自在数据处理系统的系统存储器2850中执行。

[0309] 在一些实施例中,3D图形应用2810包含一个或多个着色器程序,所述一个或多个着色器程序包括着色器指令2812。着色器语言指令可以采用高级着色器语言,诸如高级着色器语言(HLSL)或OpenGL着色器语言(GLSL)。所述应用还包括可执行指令2814,所述可执行指令采用适合于由通用处理器核2834执行的机器语言。所述应用还包括由顶点数据定义的图形对象2816。

[0310] 在一些实施例中,操作系统2820是来自微软公司的Microsoft® Windows®操作系统、专有UNIX式操作系统、或使用Linux内核变体的开源UNIX式操作系统。操作系统2820可以支持图形API 2822,诸如Direct3D API、OpenGL API或Vulkan API。当Direct3D API正在使用时,操作系统2820使用前端着色器编译器2824以将采用HLSL的任何着色器指令2812编译成较低级的着色器语言。所述编译可以是即时(JIT)编译,或者所述应用可以执行着色器预编译。在一些实施例中,在对3D图形应用2810进行编译期间,将高级着色器编译成低级着色器。在一些实施例中,着色器指令2812是以中间形式(诸如,Vulkan API所使用的标准便携式中间表示(SPIR)的版本)提供的。

[0311] 在一些实施例中,用户模式图形驱动器2826包含后端着色器编译器2827,所述后端着色器编译器用于将着色器指令2812转换成硬件专用的表示。当OpenGL API正在使用时,将采用GLSL高级语言的着色器指令2812传递至用户模式图形驱动器2826以用于编译。在一些实施例中,用户模式图形驱动器2826使用操作系统内核模式功能2828来与内核模式图形驱动器2829进行通信。在一些实施例中,内核模式图形驱动器2829与图形处理器2832进行通信以便分派命令和指令。

[0312] IP核实现

[0313] 至少一个实施例的一个或多个方面可以由存储在机器可读介质上的代表性代码实现,所述机器可读介质表示和/或定义诸如处理器之类的集成电路内的逻辑。例如,机器可读介质可以包括表示处理器内的各种逻辑的指令。当由机器读取时,所述指令可以使机器制造用于执行本文所述的技术的逻辑。这种表示(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种客户或制造设施。可以制造集成电路,使得电路执行与本文所述的实施例中的任一实施例相关联地描述的操作。

[0314] 图29是图示了根据实施例的可以用于制造用于执行操作的集成电路的IP核开发系统2900的框图。IP核开发系统2900可以用于生成可并入到更大的设计中或用于构造整个集成电路(例如,SOC集成电路)的模块化、可重复使用设计。设计设施2930可以采用高级编程语言(例如,C/C++)生成对IP核设计的软件仿真2910。软件仿真2910可以用于使用仿真模型2912来设计、测试并验证IP核的行为。仿真模型2912可以包括功能、行为和/或时序仿真。然后可以根据仿真模型2912创建或合成寄存器传送级(RTL)设计2915。RTL设计2915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RTL设计2915外,还可以创建、设计或合成逻辑级或晶体管级处的较低级设计。因此,初始设计和仿真的特定细节可以发生变化。

[0315] 可以由设计设施将RTL设计2915或等效物进一步合成为硬件模型2920,所述硬件模型可以采用硬件描述语言(HDL)或物理设计数据的某种其他表示。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2940(例如,硬盘、闪存、或任何非易失性存储介质)来存储IP核设计以用于递送至第3方制造设施2965。可替代地,可以通过有线连接2950或无线连接2960来传输(例如,经由互联网)IP核设计。制造设施2965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以被配置成执行根据本文所述的至少一个实施例的操作。

[0316] 示例性片上系统集成电路

[0317] 图30至图32图示了根据本文所述的各种实施例的可以使用一个或多个IP核来制造的示例性集成电路和相关联图形处理器。除了所图示的内容之外,还可以包括其他逻辑和电路,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。

[0318] 图30是图示了根据实施例的可以使用一个或多个IP核来制造的示例性片上系统集成电路3000的框图。示例性集成电路3000包括一个或多个应用处理器3005(例如,CPU)、至少一个图形处理器3010,并且另外还可以包括图像处理器3015和/或视频处理器3020,其中的任一项都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3000包括外围或总线逻辑,包括USB控制器3025、UART控制器3030、SPI/SDIO控制器3035和I²S/I²C控制器3040。另外,集成电路还可以包括显示设备3045,所述显示设备耦合至高清晰度多媒体接口(HDMI)控制器3050和移动产业处理器接口(MIPI)显示界面3055中的一个或多个。可以由闪存子系统3060(包括闪存和闪存控制器)来提供存储。可以经由存储器控制器3065来提供存储器接口以访问SDRAM或SRAM存储器设备。另外,一些集成电路还包括嵌入式安全引擎3070。

[0319] 图31是图示了根据实施例的可以使用一个或多个IP核来制造的片上系统集成电路的示例性图形处理器3110的框图。图形处理器3110可以是图30的图形处理器3010的变体。图形处理器3110包括顶点处理器3105和一个或多个片段处理器3115A至3115N(例如,3115A、3115B、3115C、3115D、一直到3115N-1和3115N)。图形处理器3110可以经由单独的逻辑执行不同的着色器程序,使得顶点处理器3105被优化以执行顶点着色器程序的操作,而一个或多个片段处理器3115A至3115N执行片段(例如,像素)着色操作以用于片段或像素着色器程序。顶点处理器3105执行3D图形流水线的顶点处理阶段并生成图元和顶点数据。(多个)片段处理器3115A至3115N使用由顶点处理器3105生成的图元和顶点数据来产生显示在显示设备上的帧缓冲器。在一个实施例中,(多个)片段处理器3115A至3115N被优化以执行OpenGL API中提供的片段着色器程序,这些片段着色器程序可以用于执行与Direct 3D API中提供的像素着色器程序类似的操作。

[0320] 另外,图形处理器3110还包括一个或多个存储器管理单元(MMU)3120A至3120B、(多个)高速缓存3125A至3125B和(多个)电路互连3130A至3130B。一个或多个MMU 3120A至3120B为图形处理器3110、包括为顶点处理器3105和/或(多个)片段处理器3115A至3115N提供虚拟到物理地址映射,除了存储在一个或多个高速缓存3125A至3125B中的顶点或图像/

纹理数据之外,所述虚拟到物理地址映射还可以引用存储在存储器中的顶点或图像/纹理数据。在一个实施例中,一个或多个MMU 3120A至3120B可以与系统内的其他MMU、包括与图30的一个或多个应用处理器3005、图像处理器3015和/或视频处理器3020相关联的一个或多个MMU同步,使得每个处理器3005至3020可以参与共享或统一的虚拟存储器系统。根据实施例,一个或多个电路互连3130A至3130B使得图形处理器3110能够经由SoC的内部总线或经由直接连接来与SoC内的其他IP核交互。

[0321] 图32是图示了根据实施例的可以使用一个或多个IP核来制造的片上系统集成电路的附加示例性图形处理器3210的框图。图形处理器3210可以是图30的图形处理器3010的变体。图形处理器3210包括图31的集成电路3100的一个或多个MMU 3120A至3120B、高速缓存3125A至3125B和电路互连3130A至3130B。

[0322] 图形处理器3210包括一个或多个着色器核3215A至3215N(例如,3215A、3215B、3215C、3215D、3215E、3215F、一直到3215N-1和3215N),所述一个或多个着色器核提供统一的着色器核架构,其中单个核或类型或核可以执行所有类型的可编程着色器代码,包括着色器程序代码以实现顶点着色器、片段着色器和/或计算着色器。存在的着色器核的确切数量可以在实施例和实现之间变化。另外,图形处理器3210还包括核间任务管理器3205,所述核间任务管理器充当用于将执行线程分派给一个或多个着色器核3215A至3215N的线程分派器以及用于加快分块操作以进行基于图块的渲染的分块单元3218,其中场景的渲染操作在图像空间中被细分,例如以便利用场景内的局部空间一致性或优化内部高速缓存的使用。

[0323] 以下示例涉及另外的实施例。示例1是一种用于在多核平台上运行具有多个网络节点的深度学习网络的流水线框架,所述流水线框架包括:网络工作负荷分析器,其要接收工作负荷、要分析所述工作负荷的计算分发、并且要将所述网络节点分组成多组;以及网络执行器,其要将每个组指派给所述多核平台的处理核,使得相应的处理核处理针对相应组的接收到的工作负荷的计算任务。

[0324] 在示例2中,示例1的主题可以可选地包括所述网络工作负荷分析器将所述网络分组成多个组,所述组的数量等于所述多核平台的核的数量。

[0325] 在示例3中,示例1或2的主题可以可选地包括所述网络工作负荷分析器基于每个组的计算量来对所述网络节点进行分组。

[0326] 在示例4中,示例3的主题可以可选地包括所述网络工作负荷分析器向每个组分发类似量的工作负荷。

[0327] 在示例5中,所述示例中的任何一个或多个的主题可以可选地包括所述深度学习网络被配置为非循环图,并且通过每个网络节点的数据流在单个方向上流过所述图,并且其中,通过每个组的数据流针对每个组在相同方向上。

[0328] 在示例6中,所述示例中的任何一个或多个的主题可以可选地包括所述工作负荷包括连续图像的序列,并且其中,每个图像被指派给多个组。

[0329] 示例7是一种多处理器计算方法,包括:接收工作负荷以供流水线框架在具有多个网络节点的深度学习网络上运行;分析所述工作负荷跨所述网络节点的计算分发;基于所述计算分发将所述网络节点分组成多组;将每个组指派给所述流水线框架的相应处理资源;以及基于所述指派在所述处理资源上执行所述工作负荷。

[0330] 在示例8中,示例7的主题可以可选地包括将所述执行的输出提供给另外的过程。

[0331] 在示例9中,示例7或8的主题可以可选地包括所述处理资源包括多核处理器的核。

[0332] 在示例10中,示例9的主题可以可选地包括每个相应的处理核处理相应的网络节点组的接收到的工作负荷的计算任务。

[0333] 在示例11中,所述示例中的任何一个或多个的主题可以可选地包括对所述网络节点进行分组包括对所述网络节点进行分组,以便跨所述组均匀地分发所述工作负荷。

[0334] 示例12是一种深度学习系统,包括:传感器阵列,其要捕获图像序列;输入缓冲器,其要接收所述图像序列;具有多个核的网络执行器,其要在所述多个核上执行深度学习网络;以及网络工作负荷分析器,其要接收所述图像序列作为工作负荷,要分析所述工作负荷的计算分发,并且要将所述网络节点分组成多组,其中所述网络执行器将每个组指派给相应的核并使用所述相应的核来执行所述组。

[0335] 在示例13中,示例12的主题可以可选地包括所述网络执行器还要将共享存储器分配给所述多个核中的一个核作为输出并分配给所述相应核中的另一个核作为输入,以便顺序执行每个相应的网络节点组。

[0336] 在示例14中,示例12或13的主题可以可选地包括所述网络工作负荷分析器基于每个组的计算量来对所述网络节点进行分组。

[0337] 在示例15中,所述示例中的任何一个或多个的主题可以可选地包括所述深度学习网络被配置为非循环图,并且通过每个网络节点的数据流在单个方向上流过所述图,并且其中,通过每个组的数据流针对每个组在相同方向上。

[0338] 前面的描述和附图应被视为例示性而非限制性意义。本领域技术人员将理解,可以对本文描述的实施例进行各种修改和改变而不脱离所附权利要求中阐述的本发明的更广泛的精神和范围。

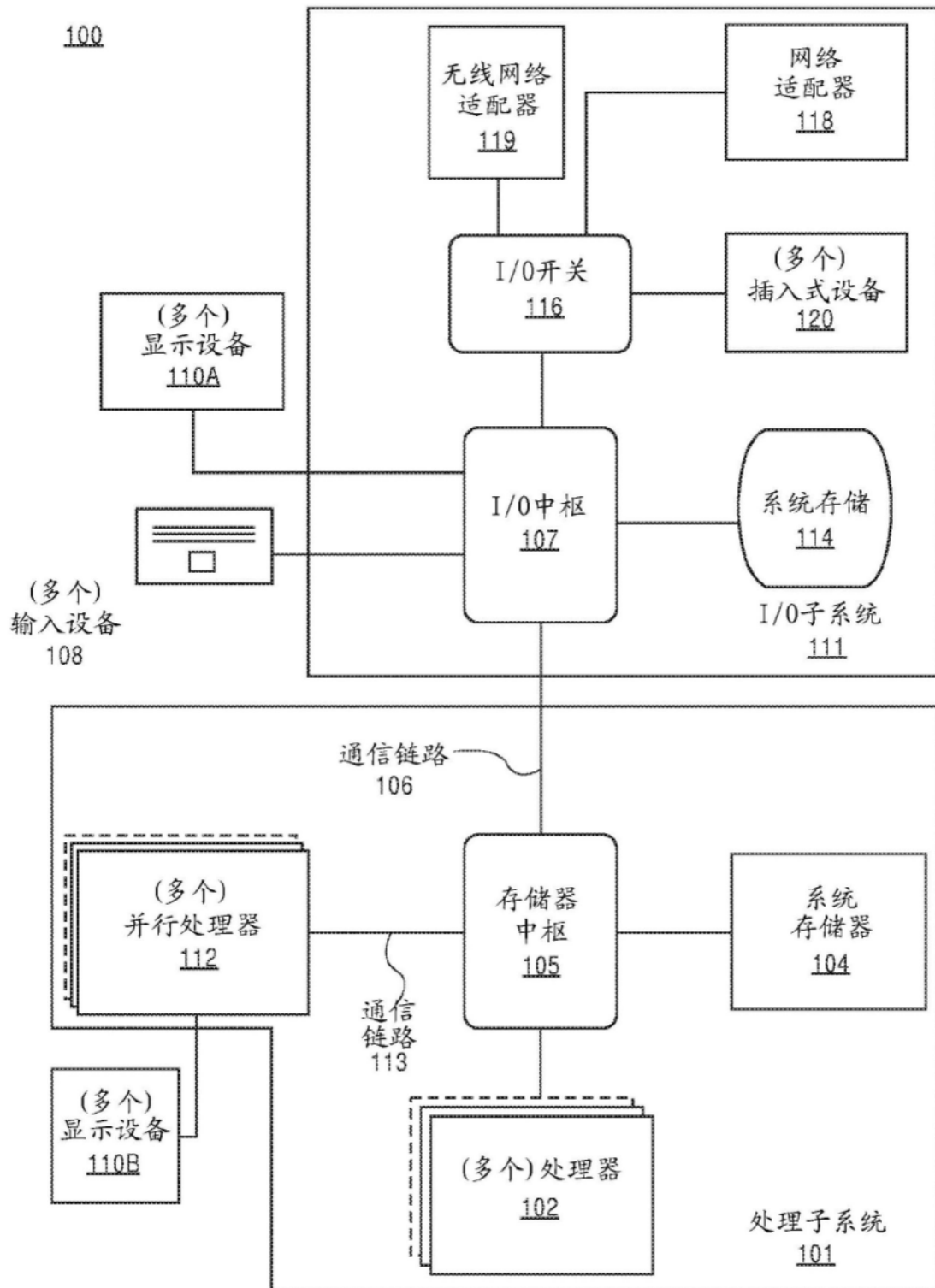


图1

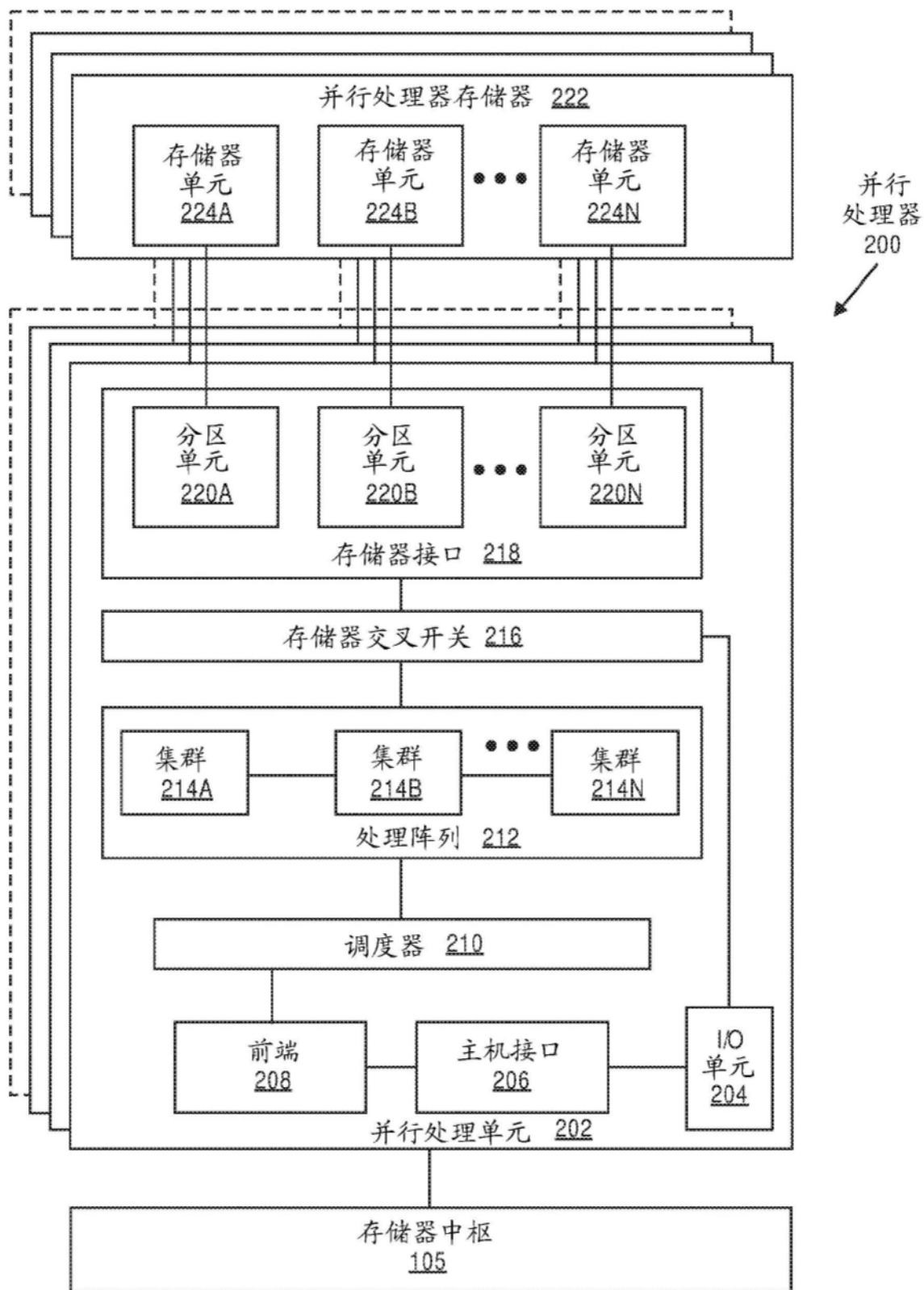


图2A

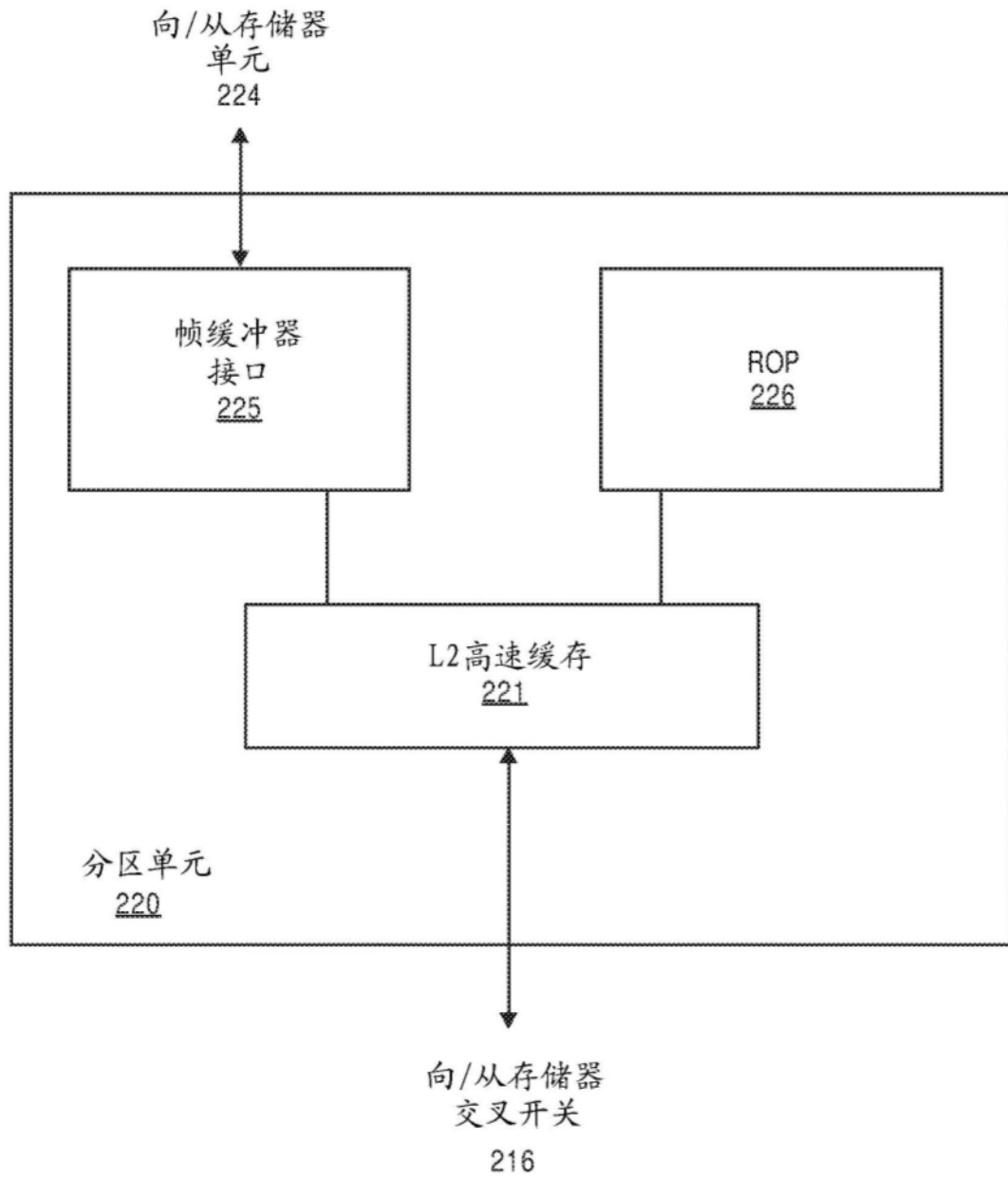


图2B

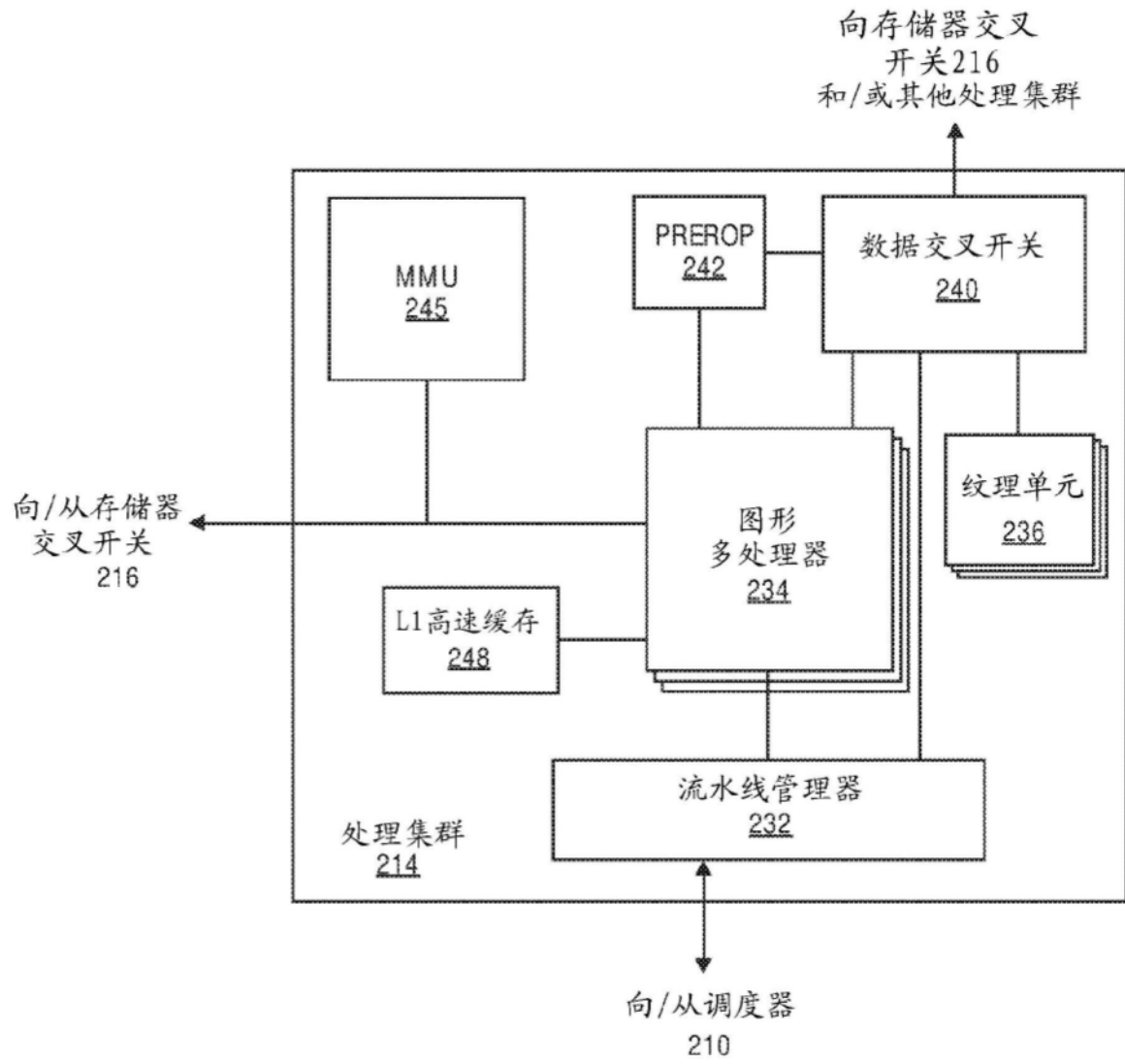


图2C

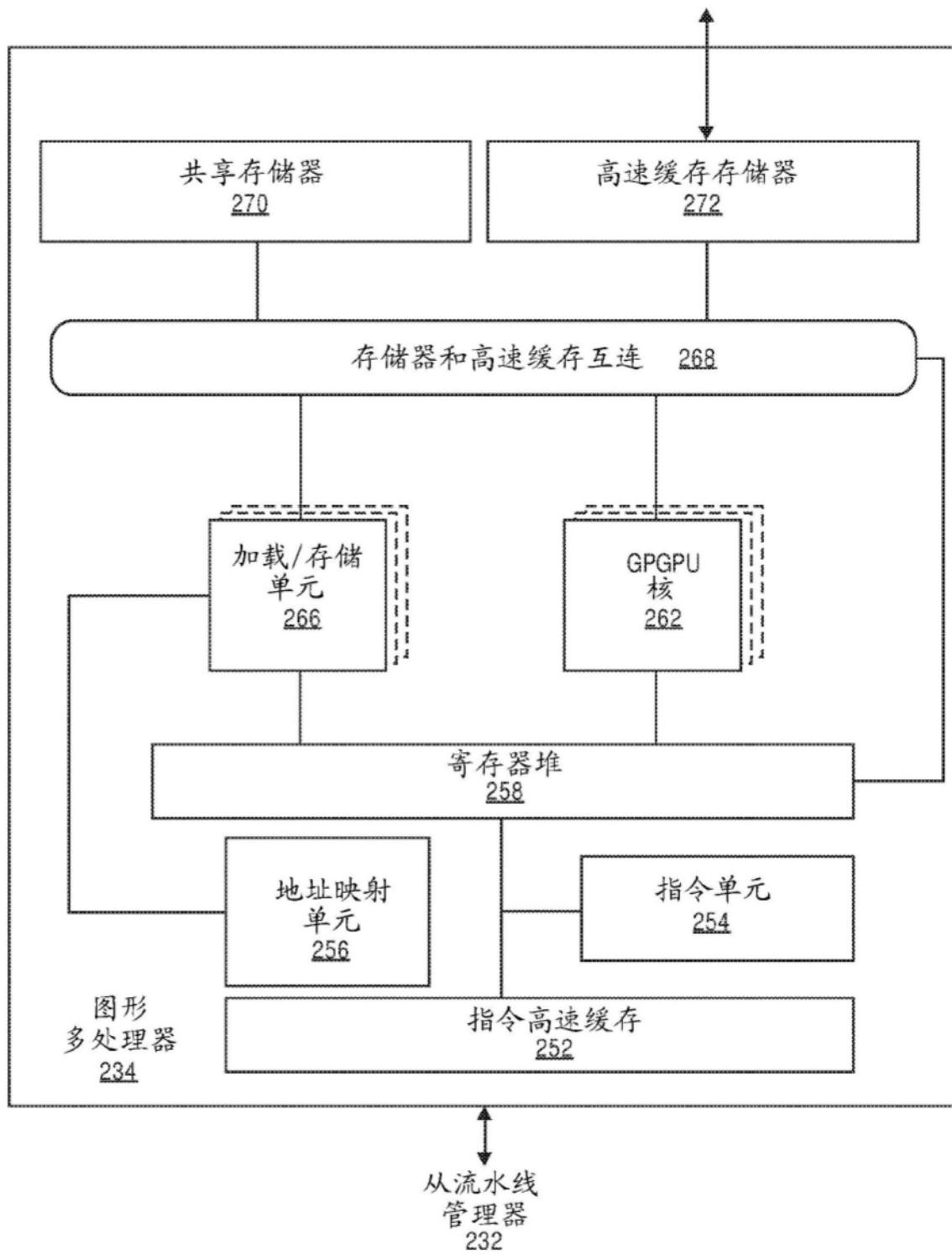


图2D

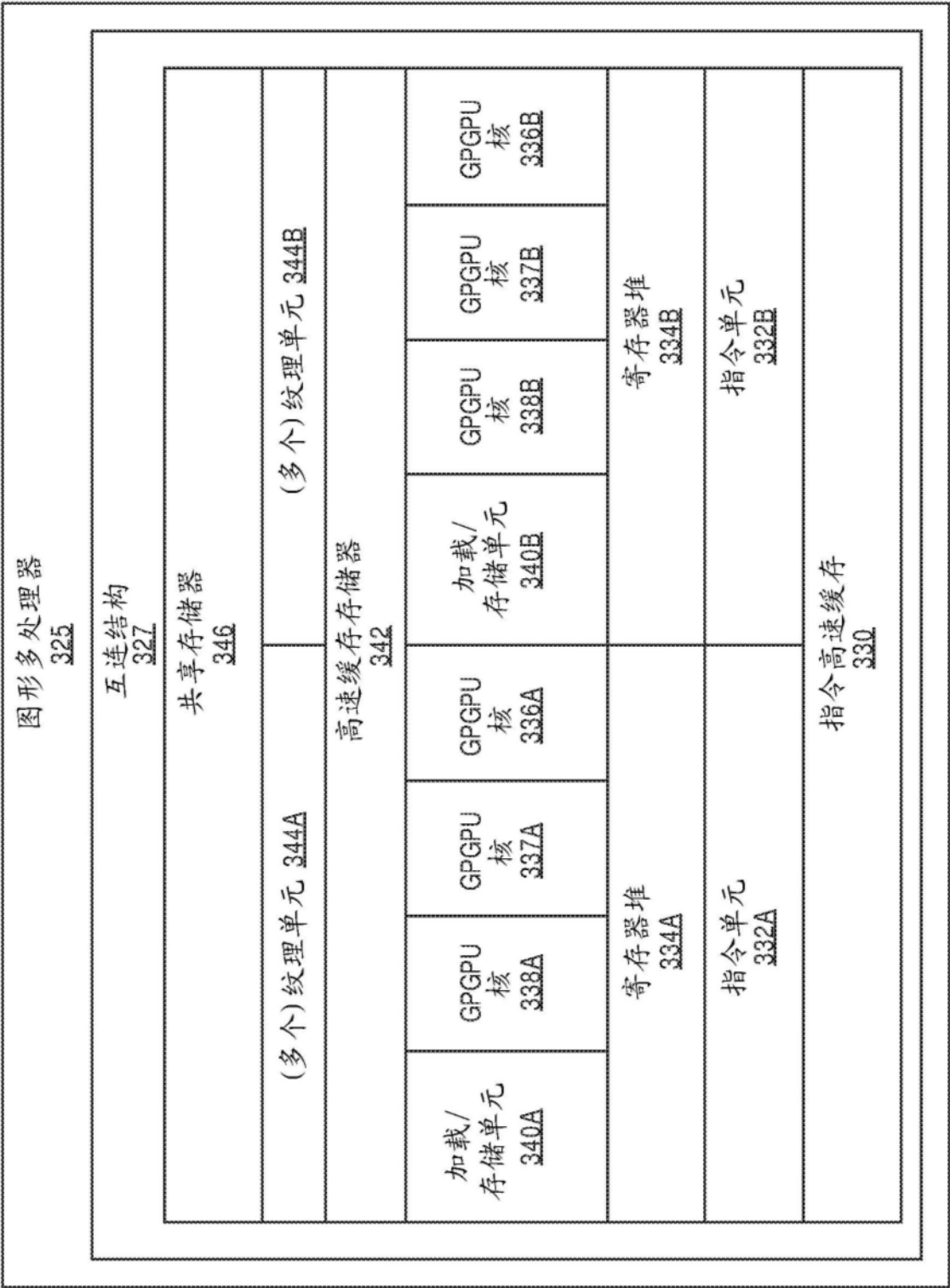


图3A

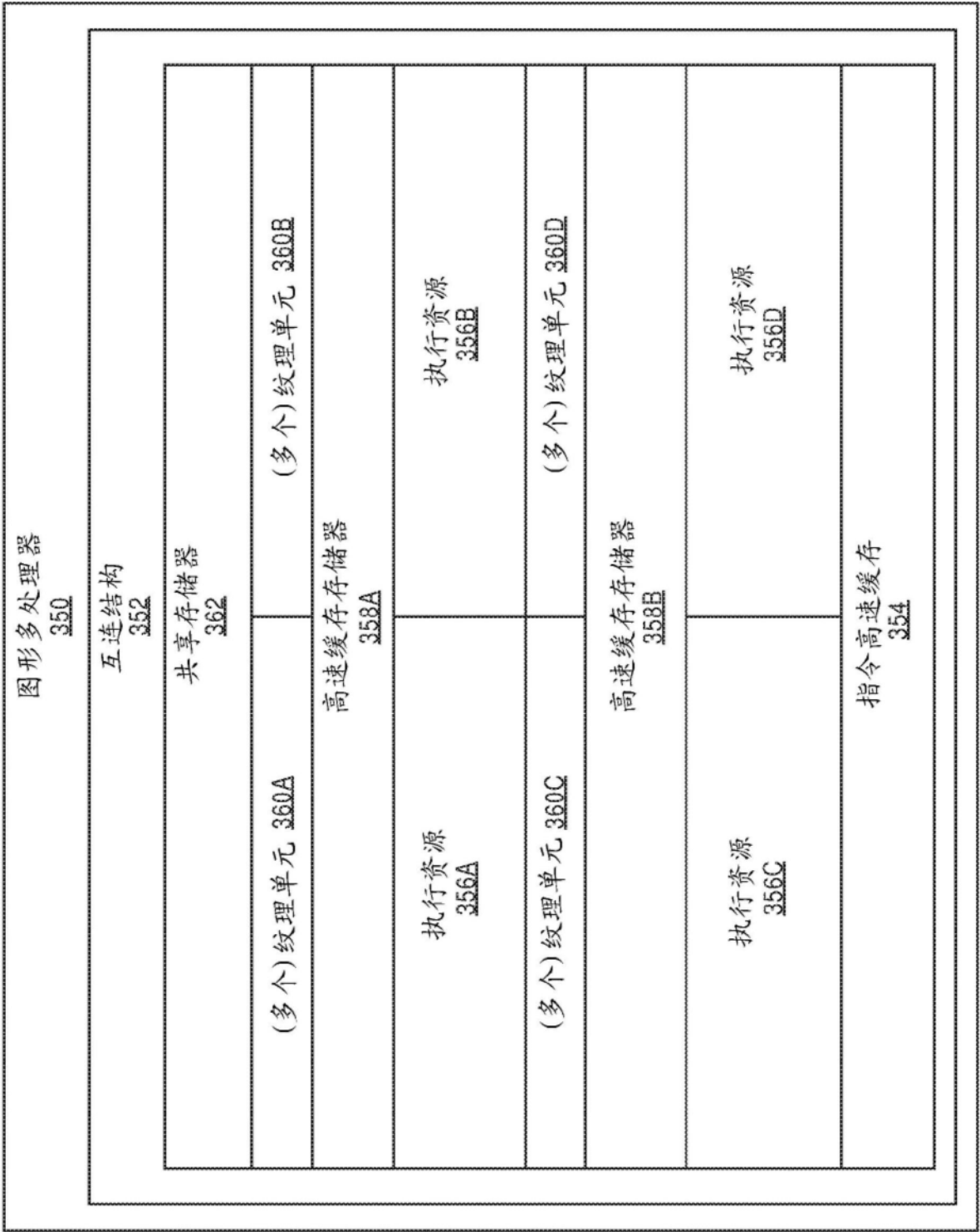


图3B

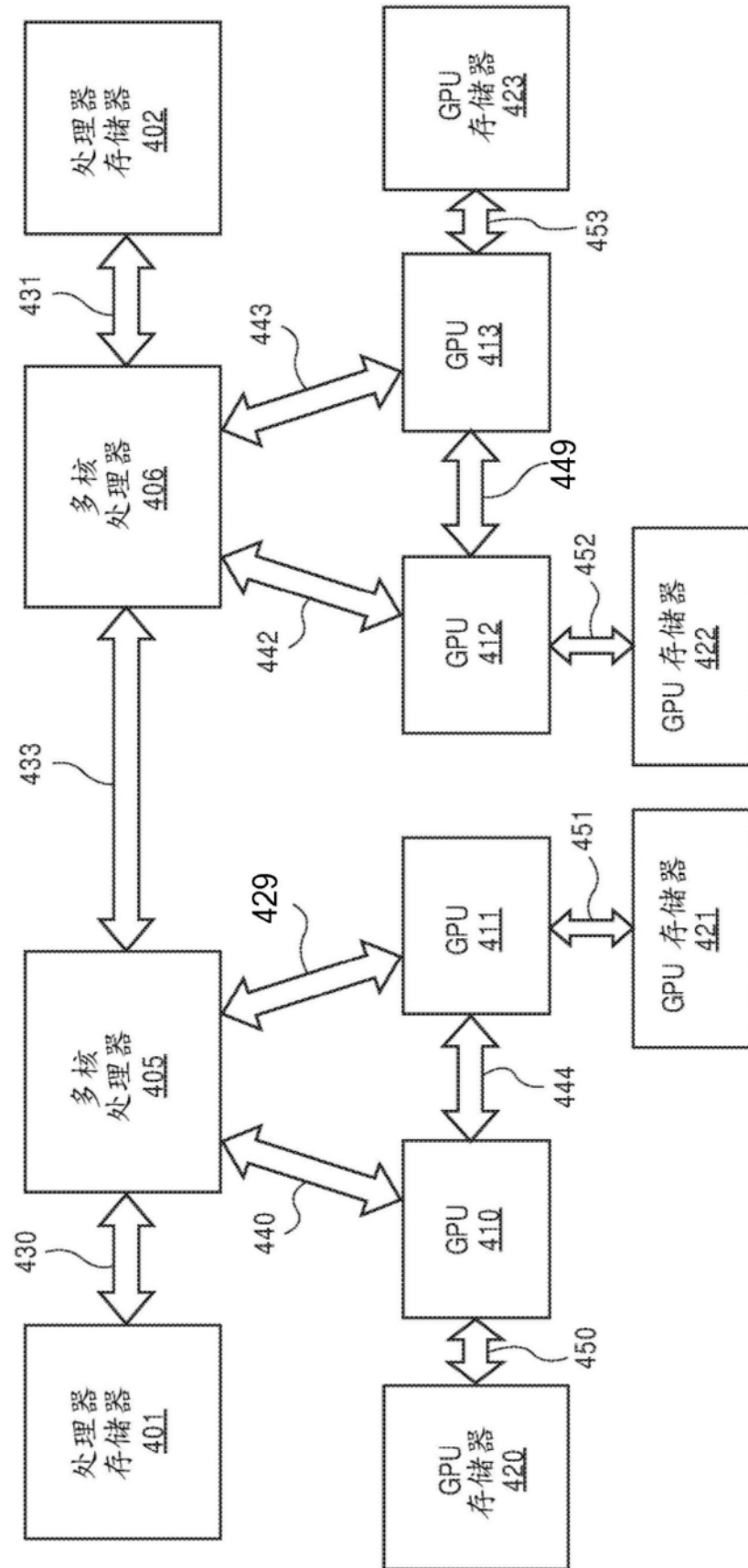


图4A

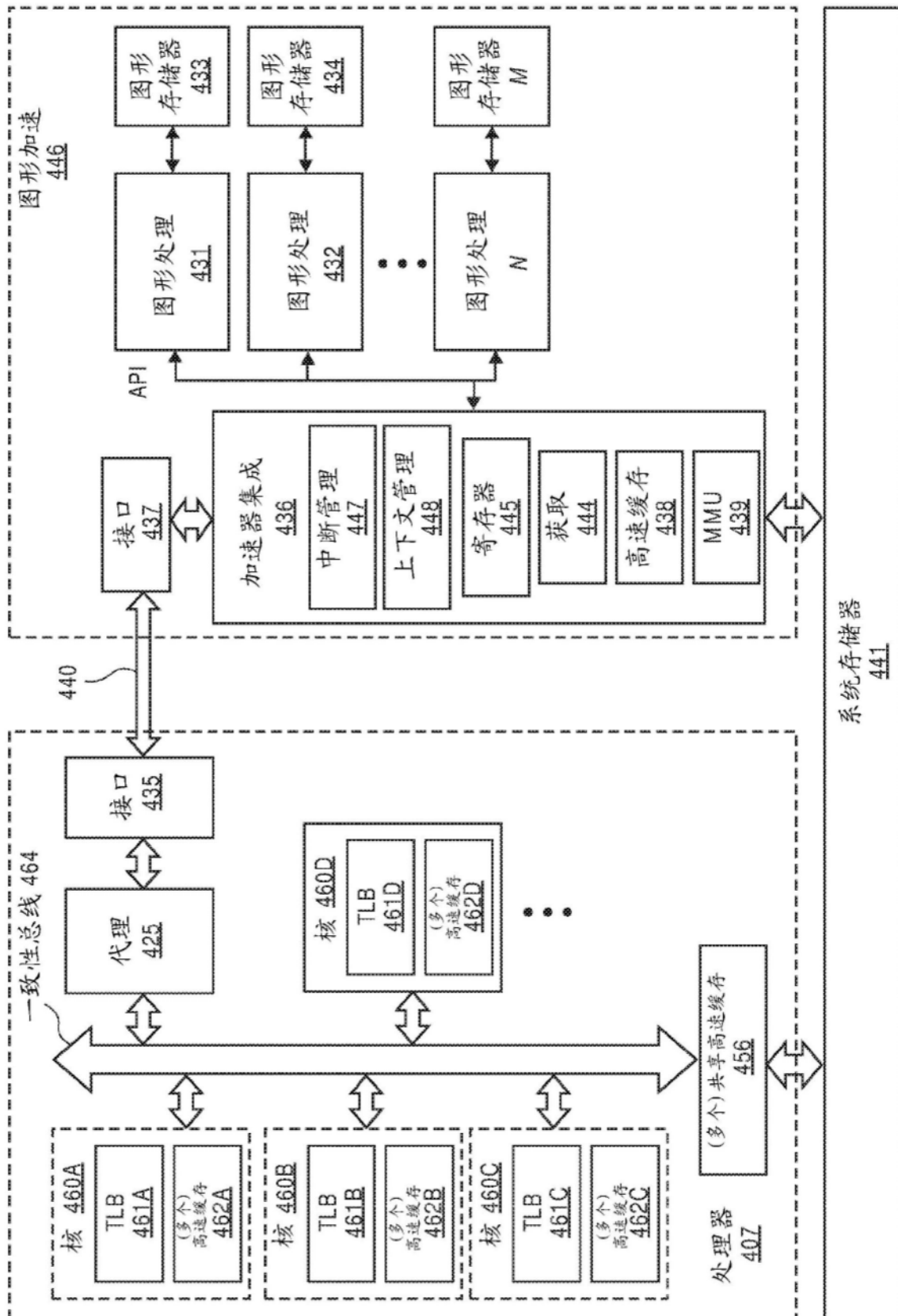


图4B

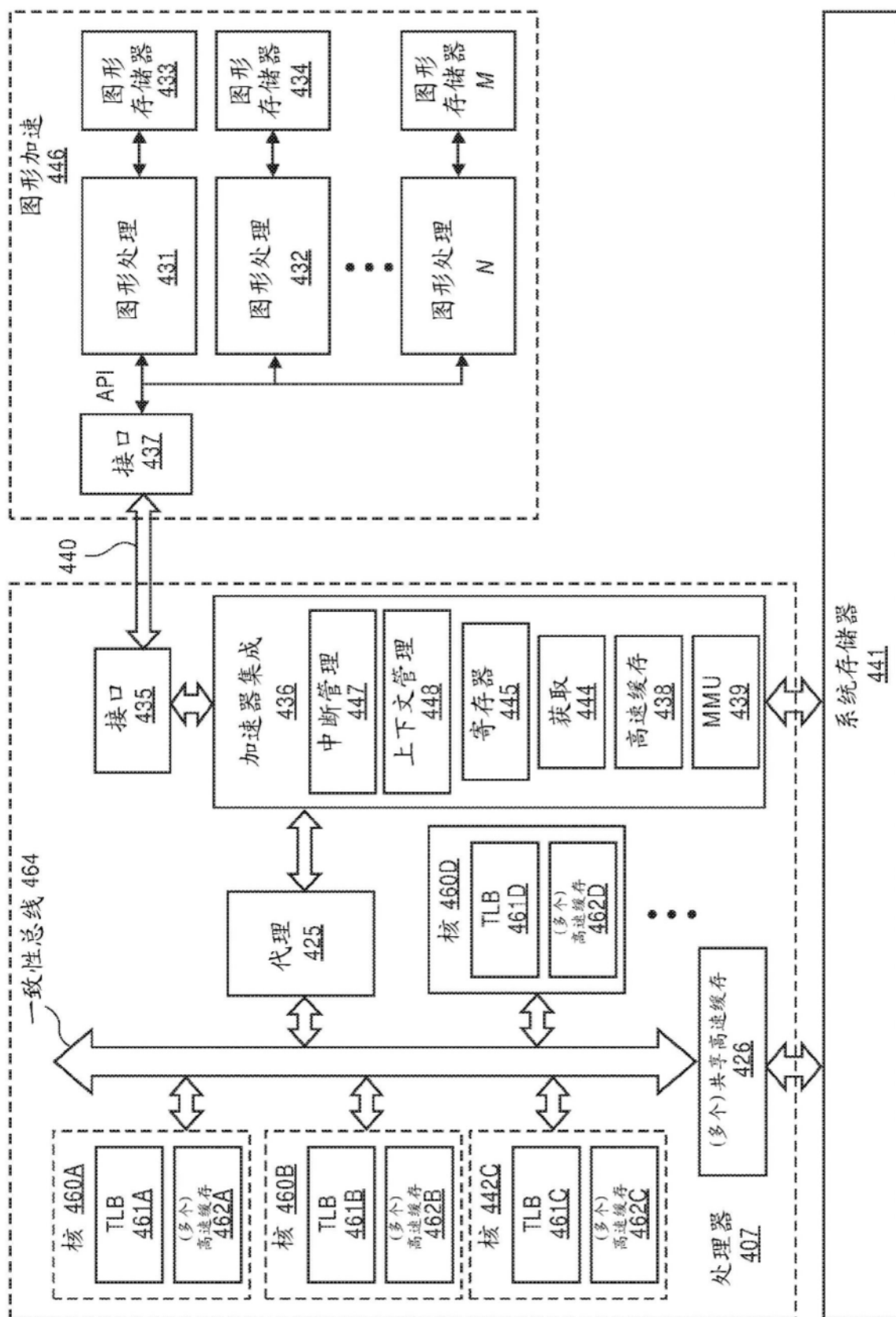


图4C

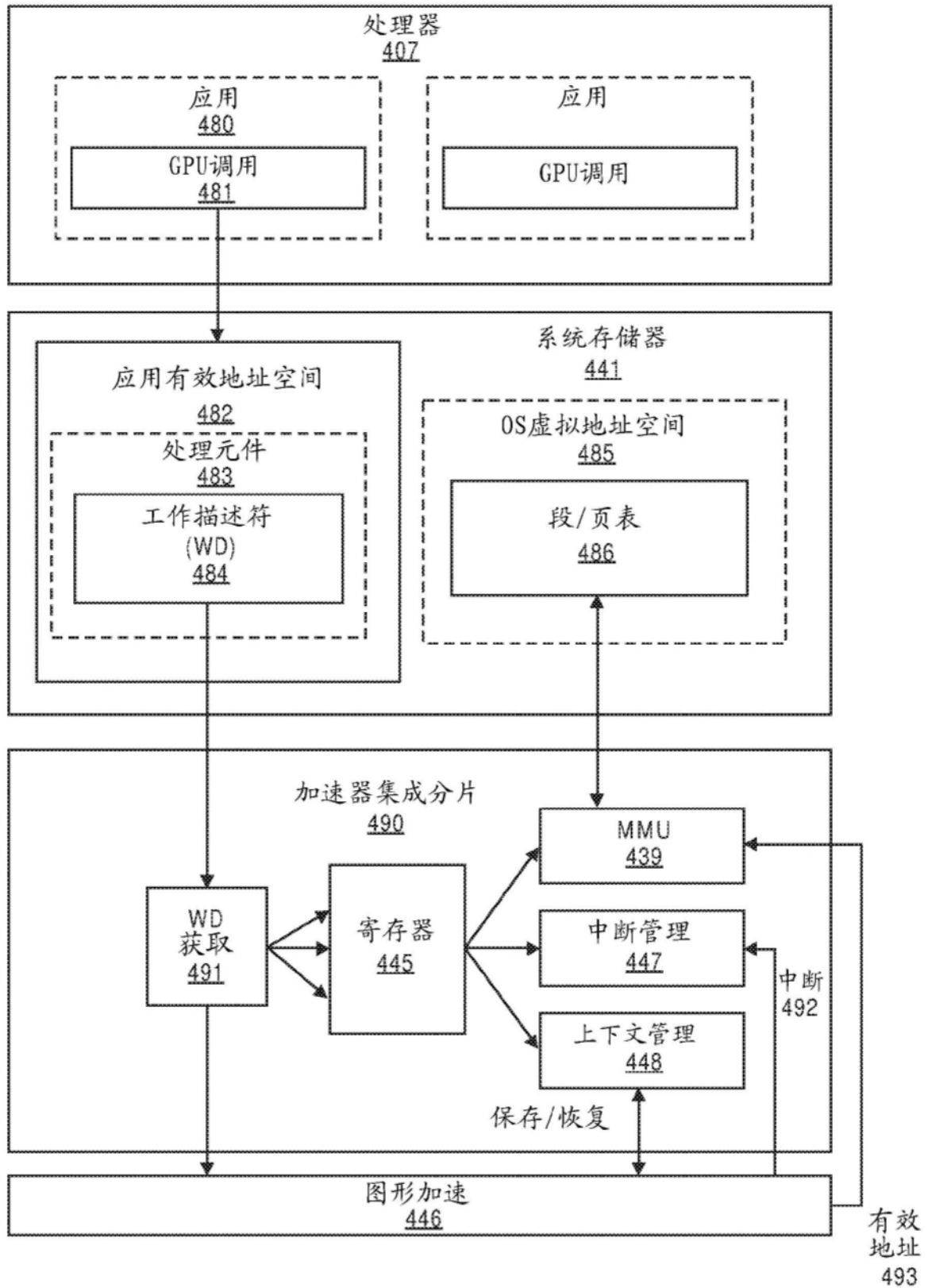


图4D

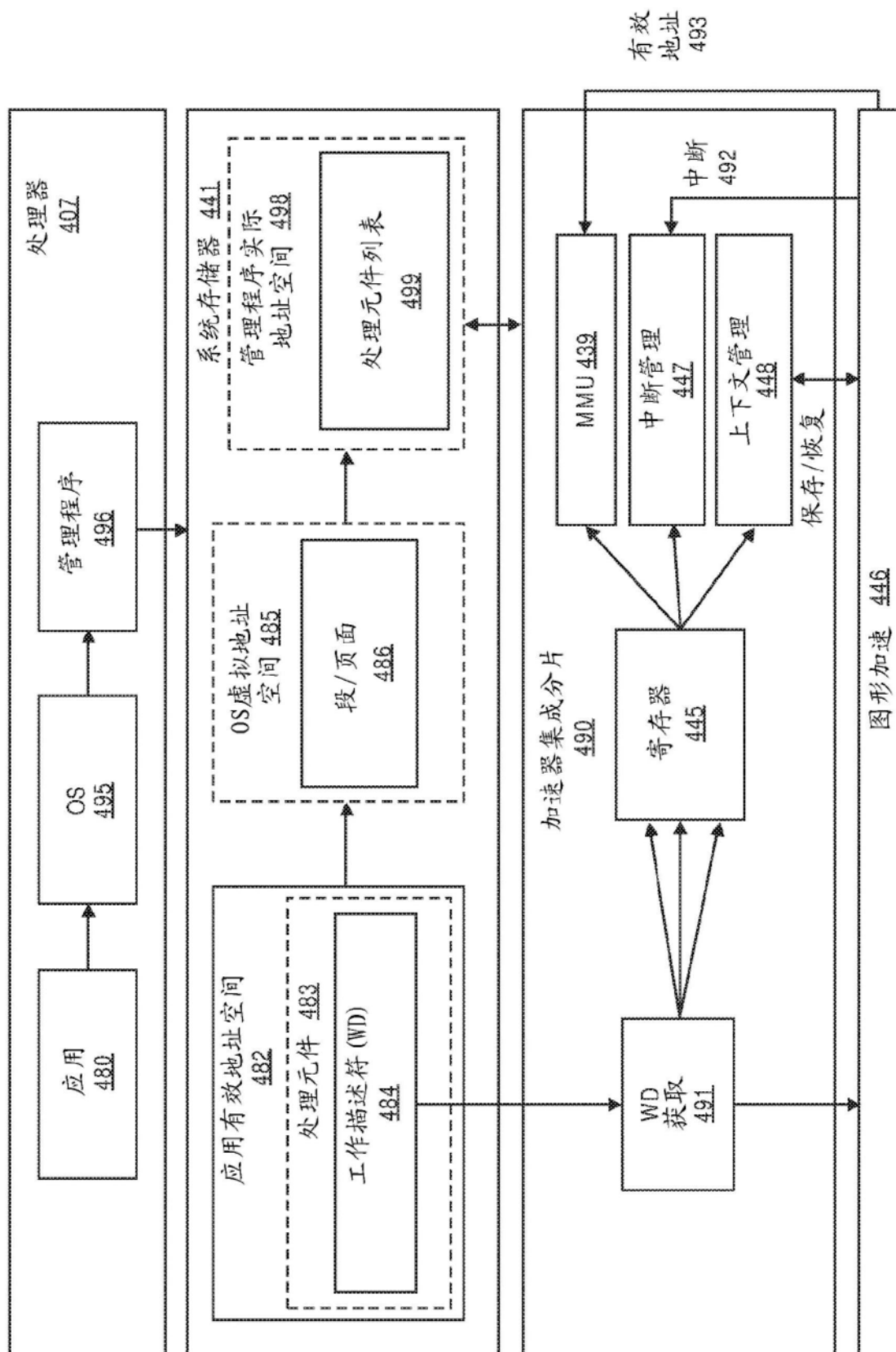


图4E

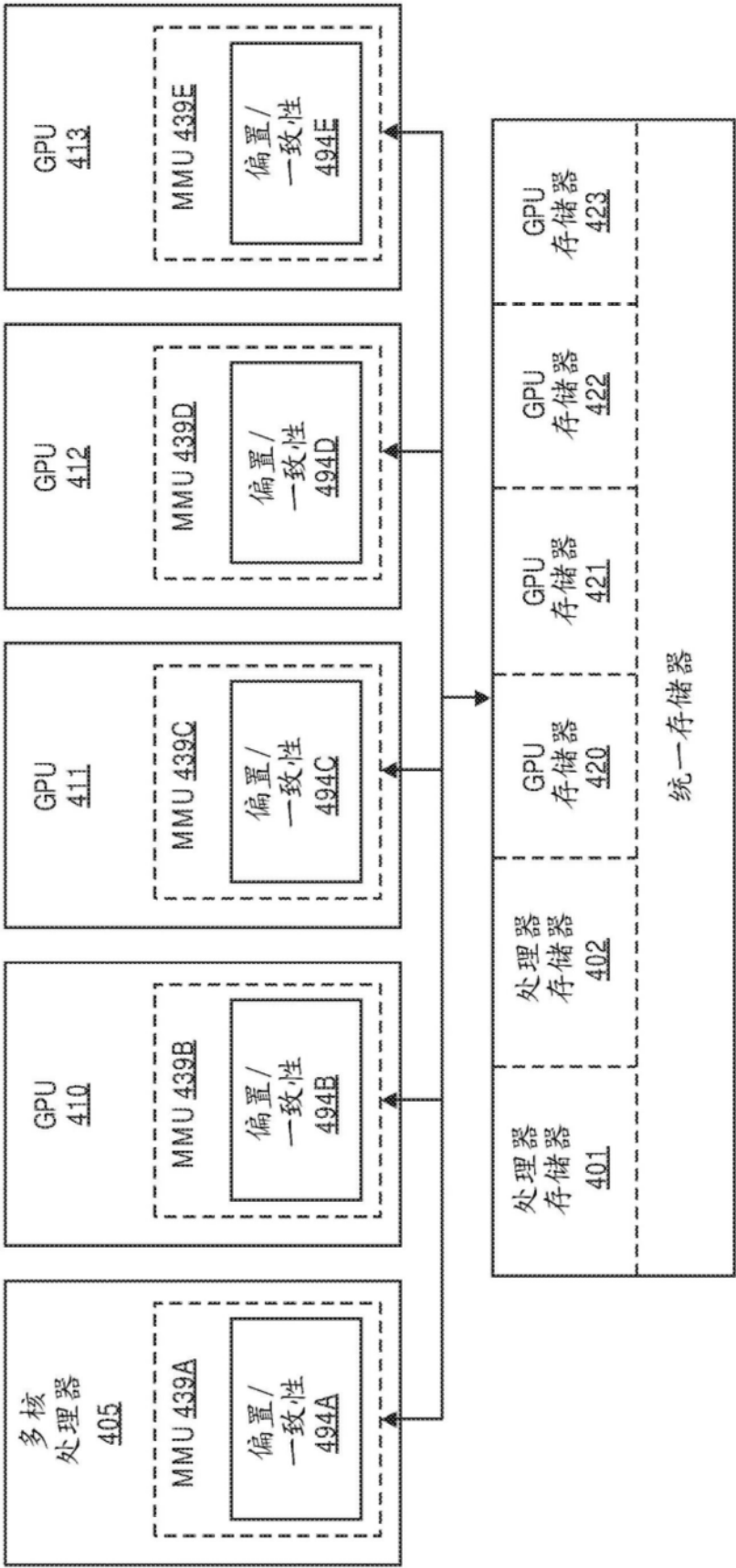


图4F

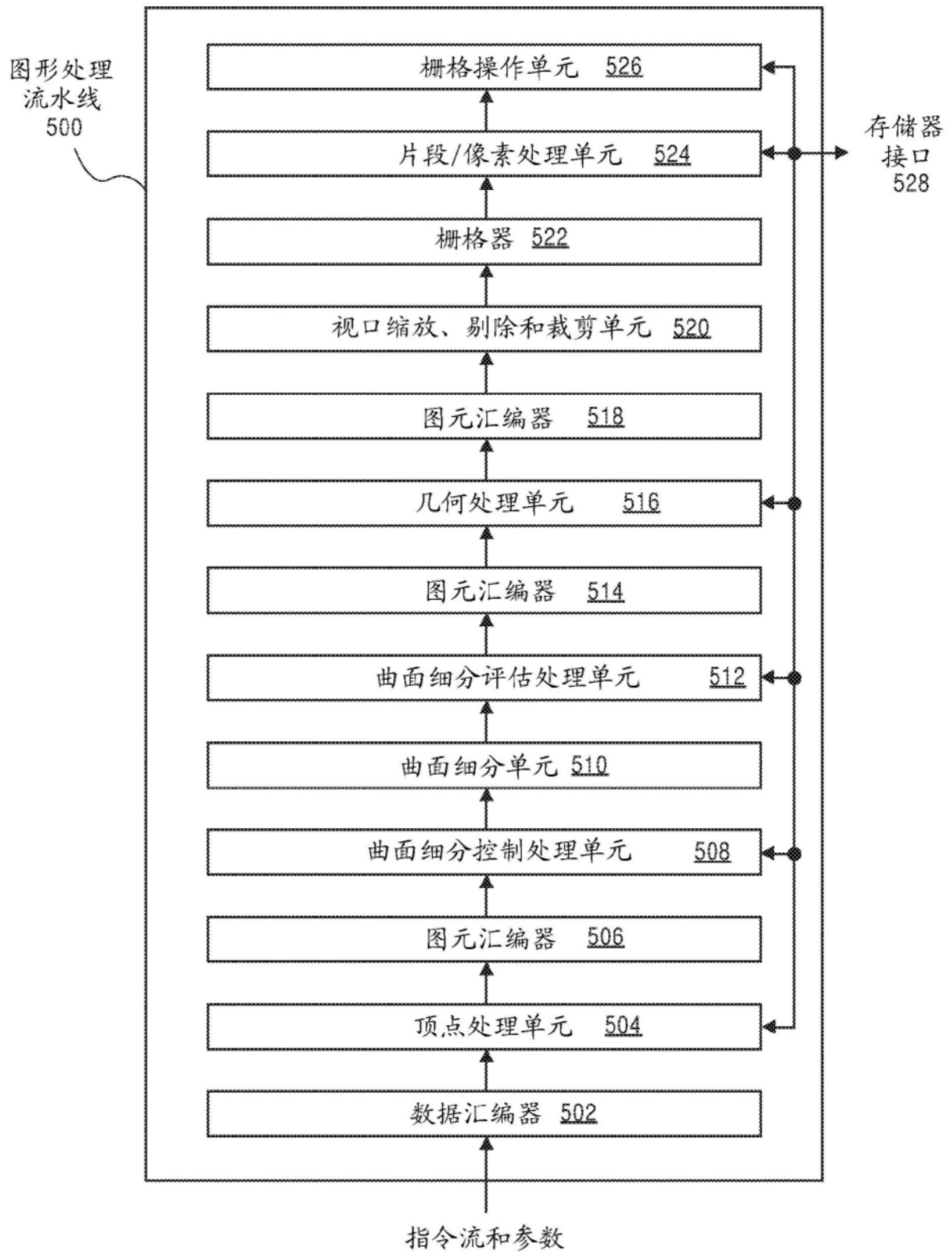


图5

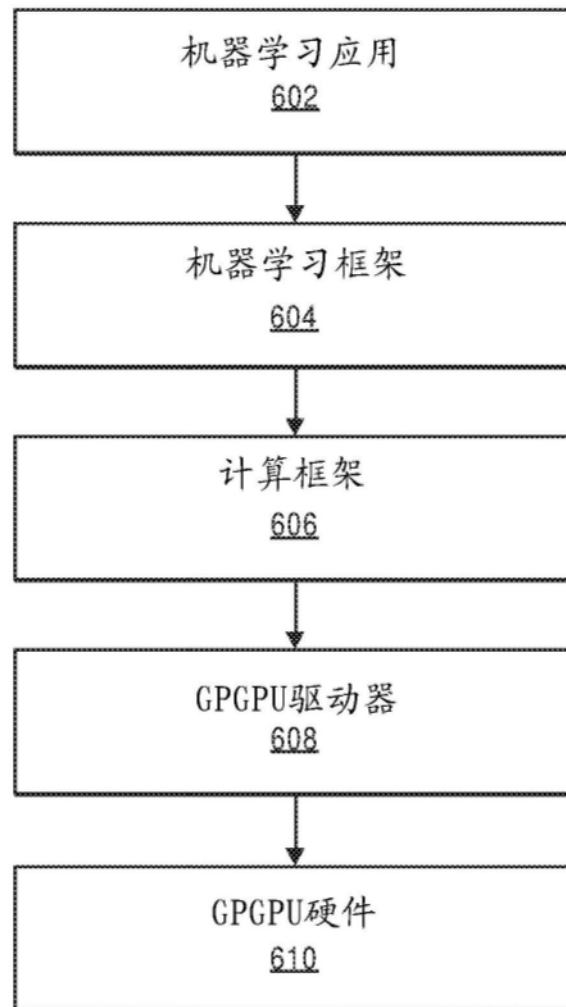
600

图6

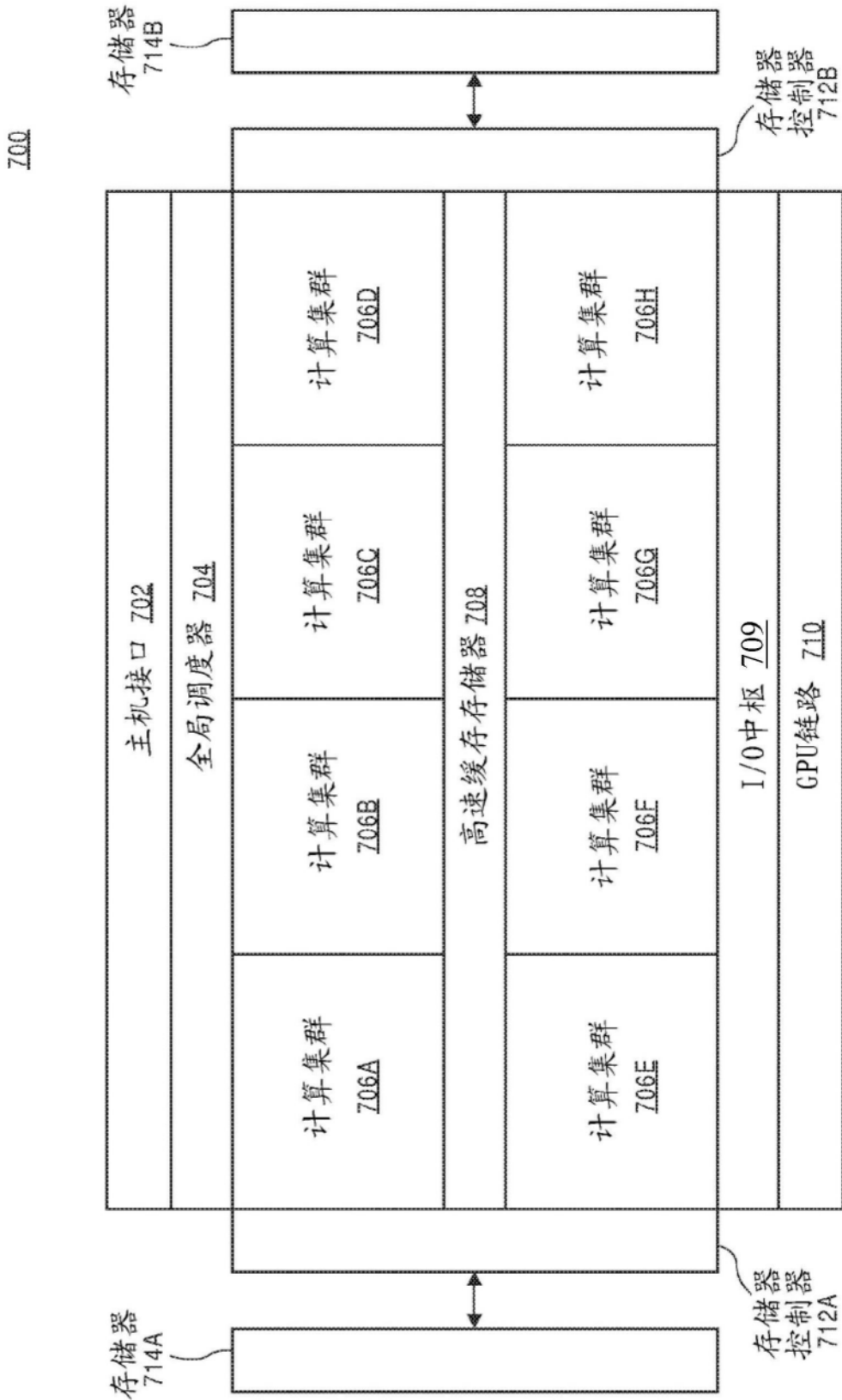


图7

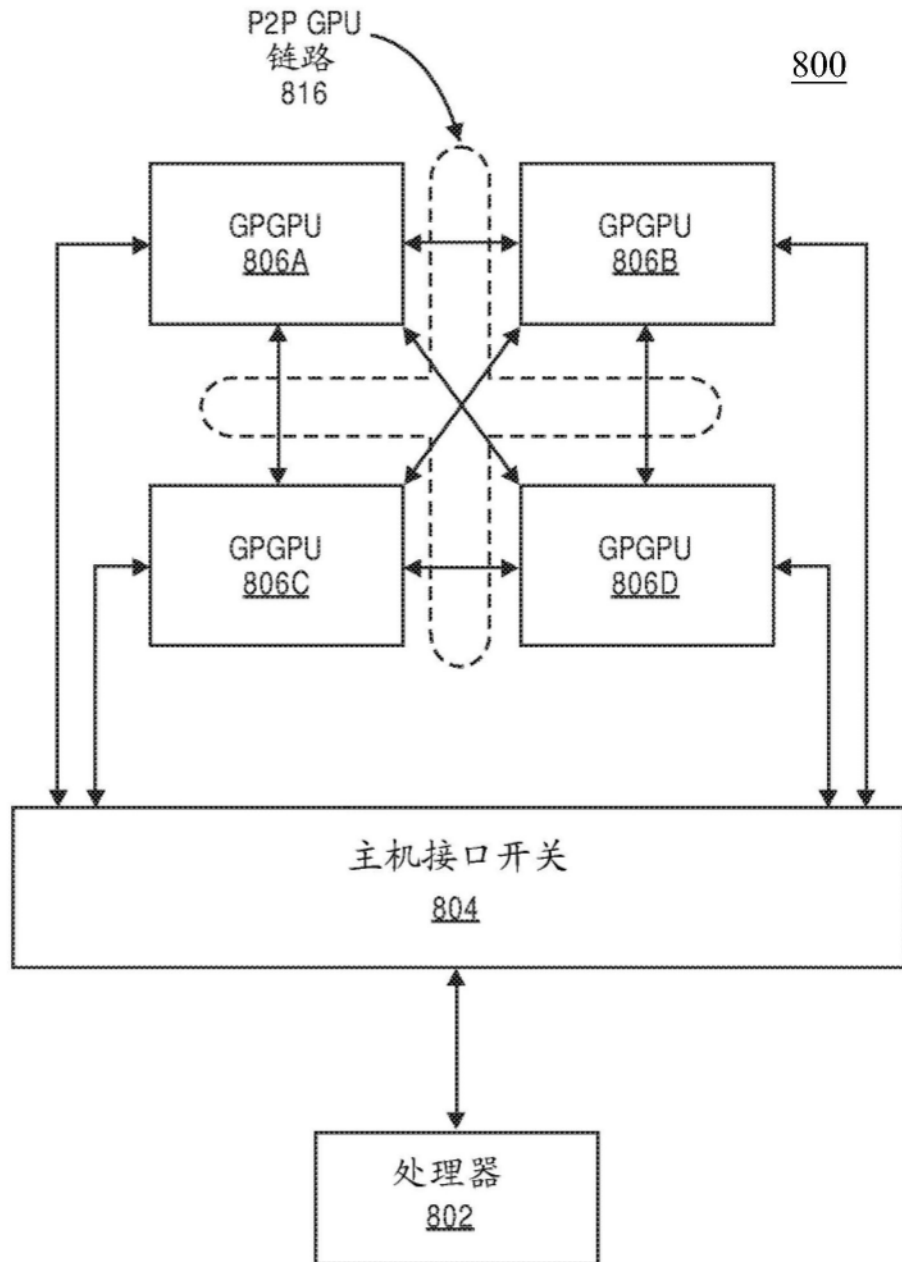


图8

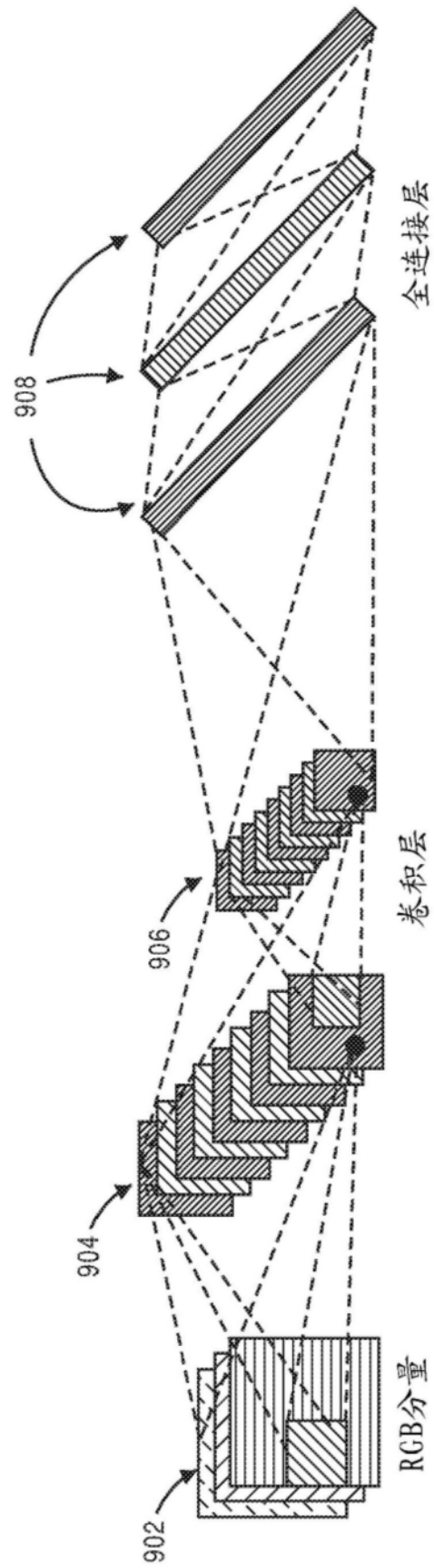


图9A

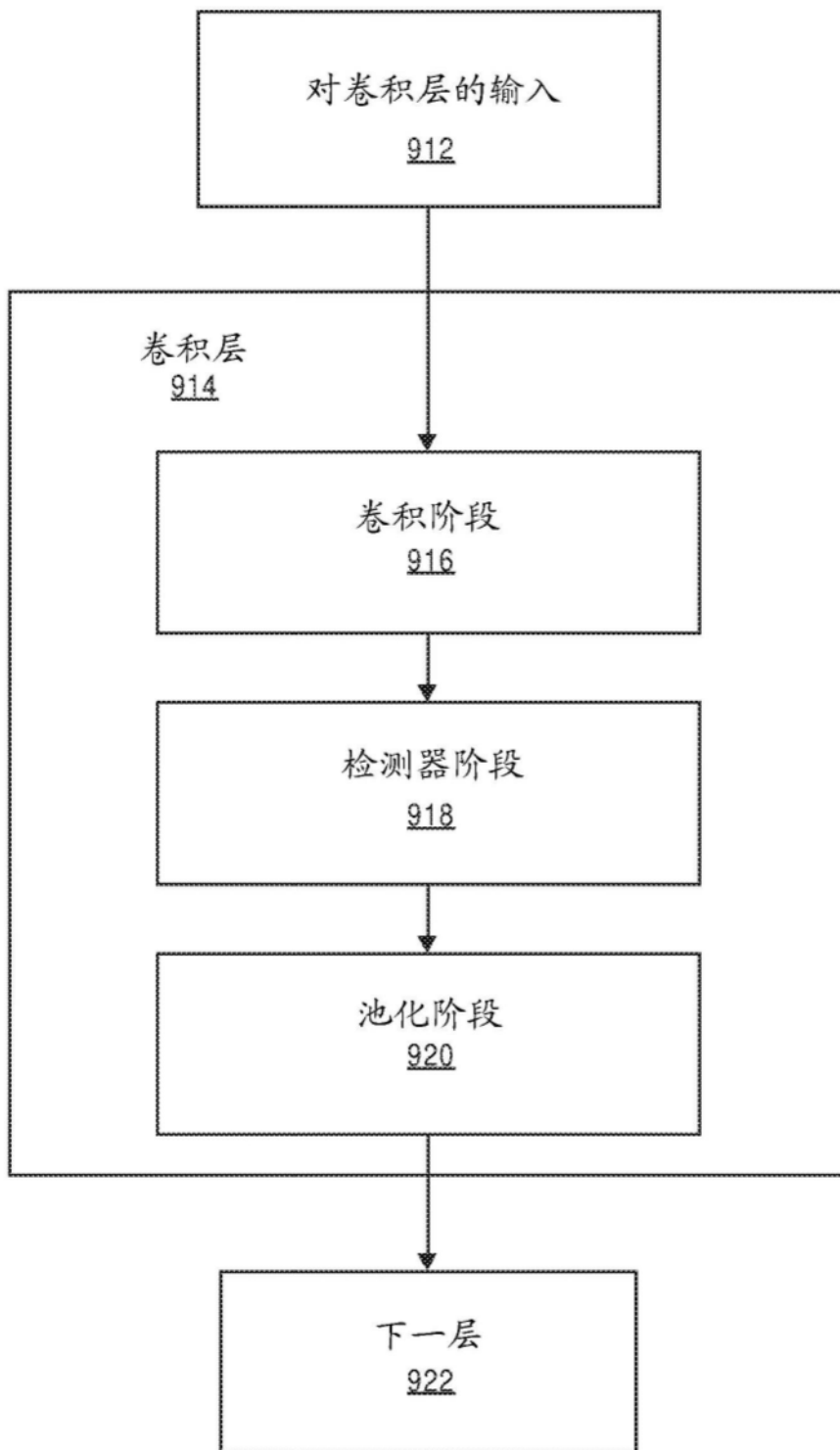


图9B

1000

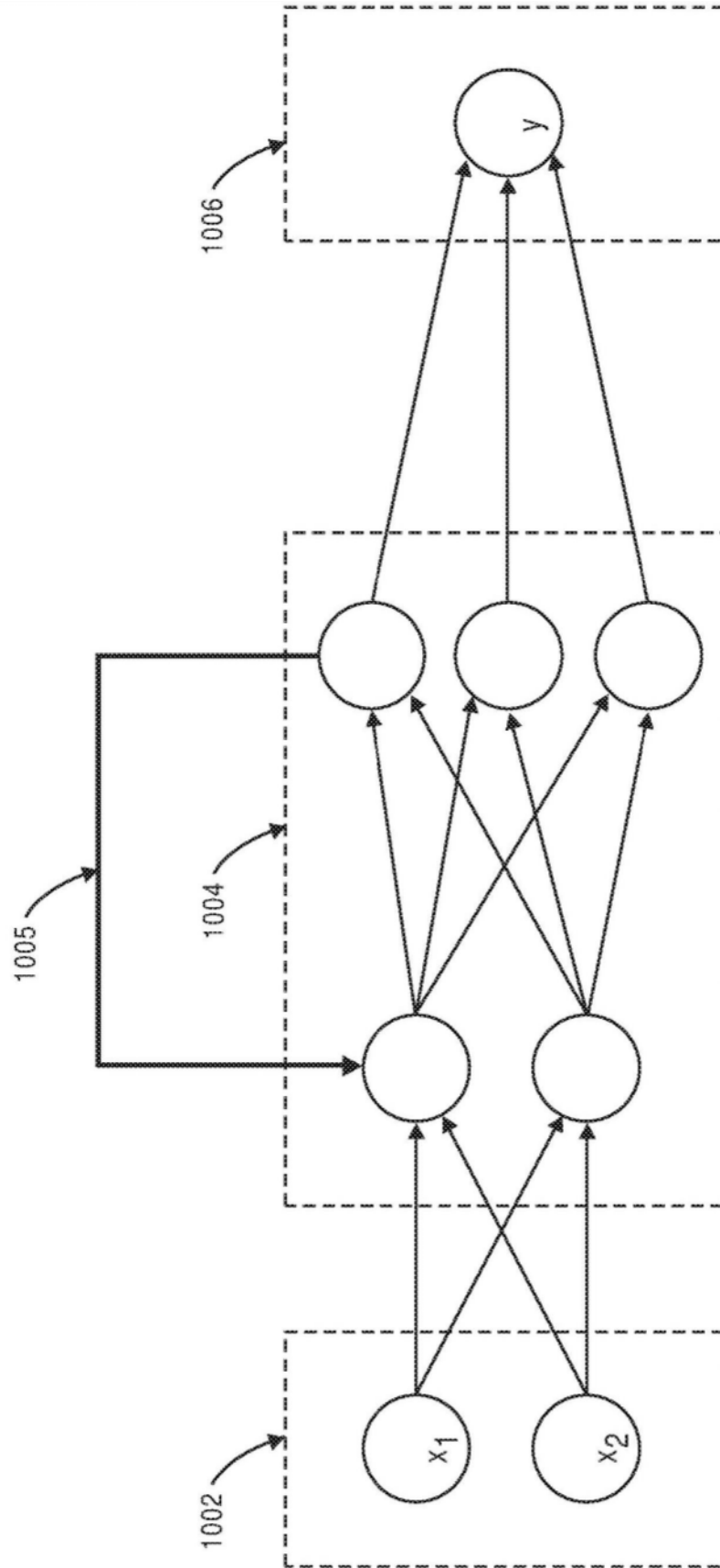


图10

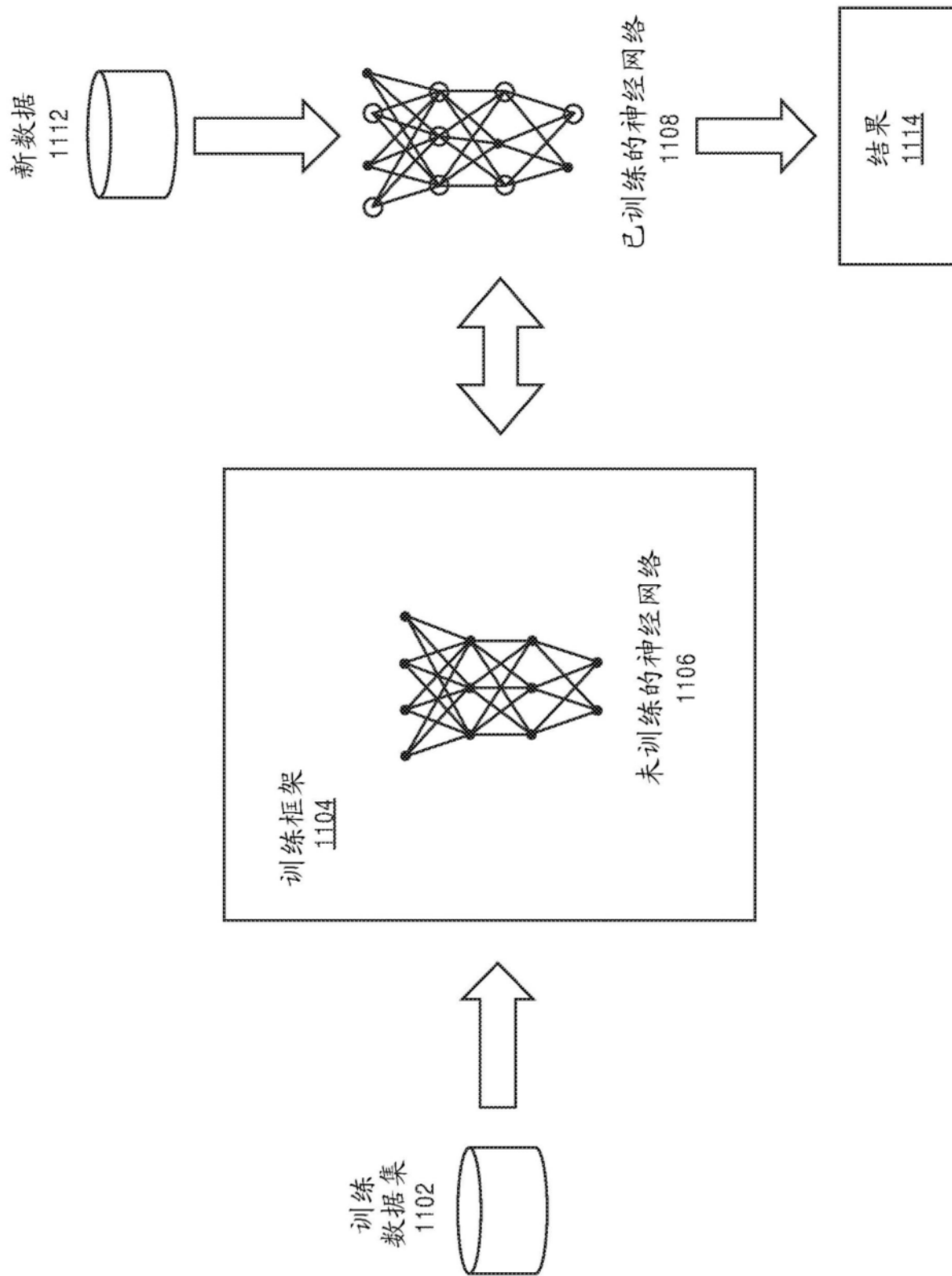


图11

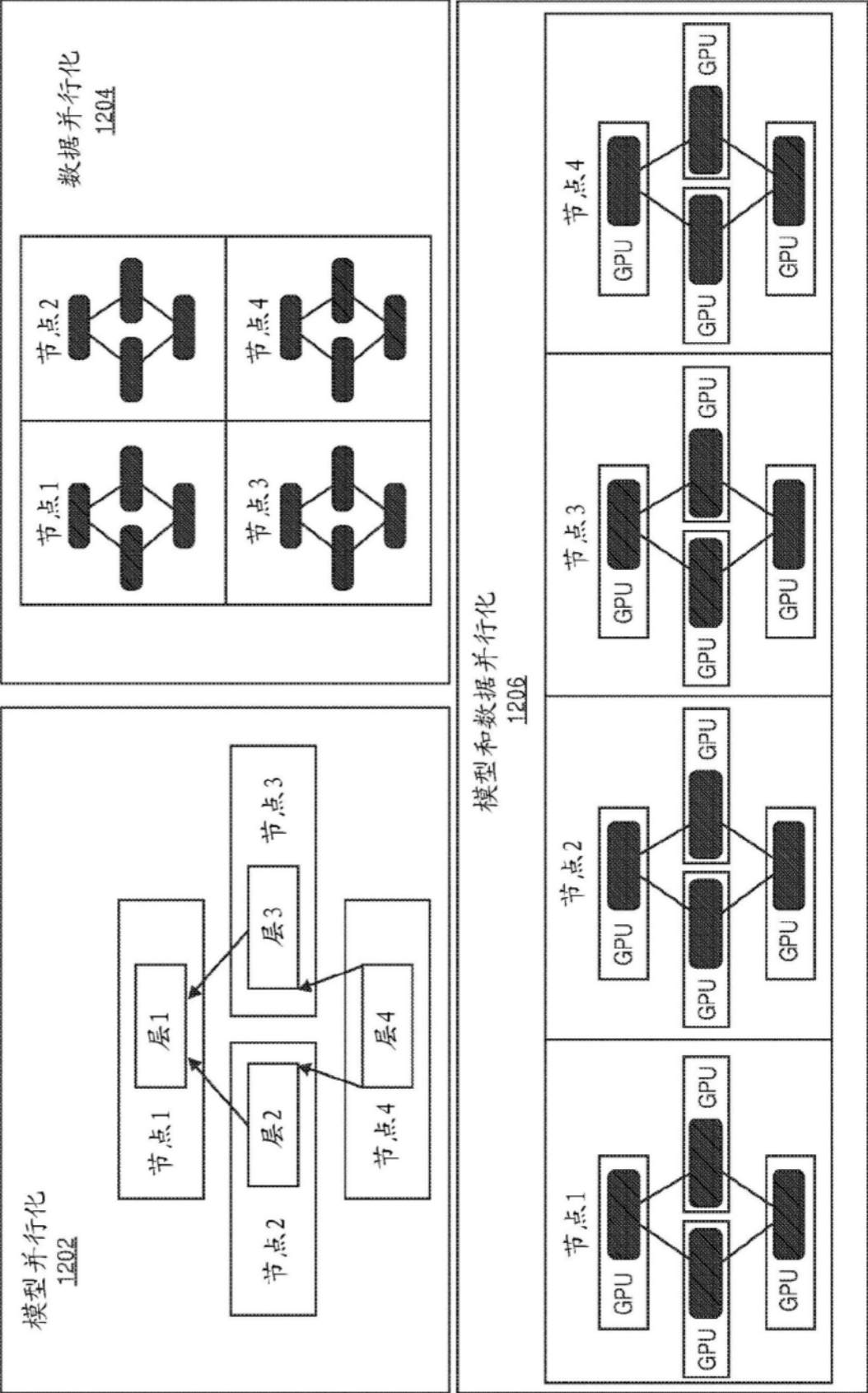


图12

1300

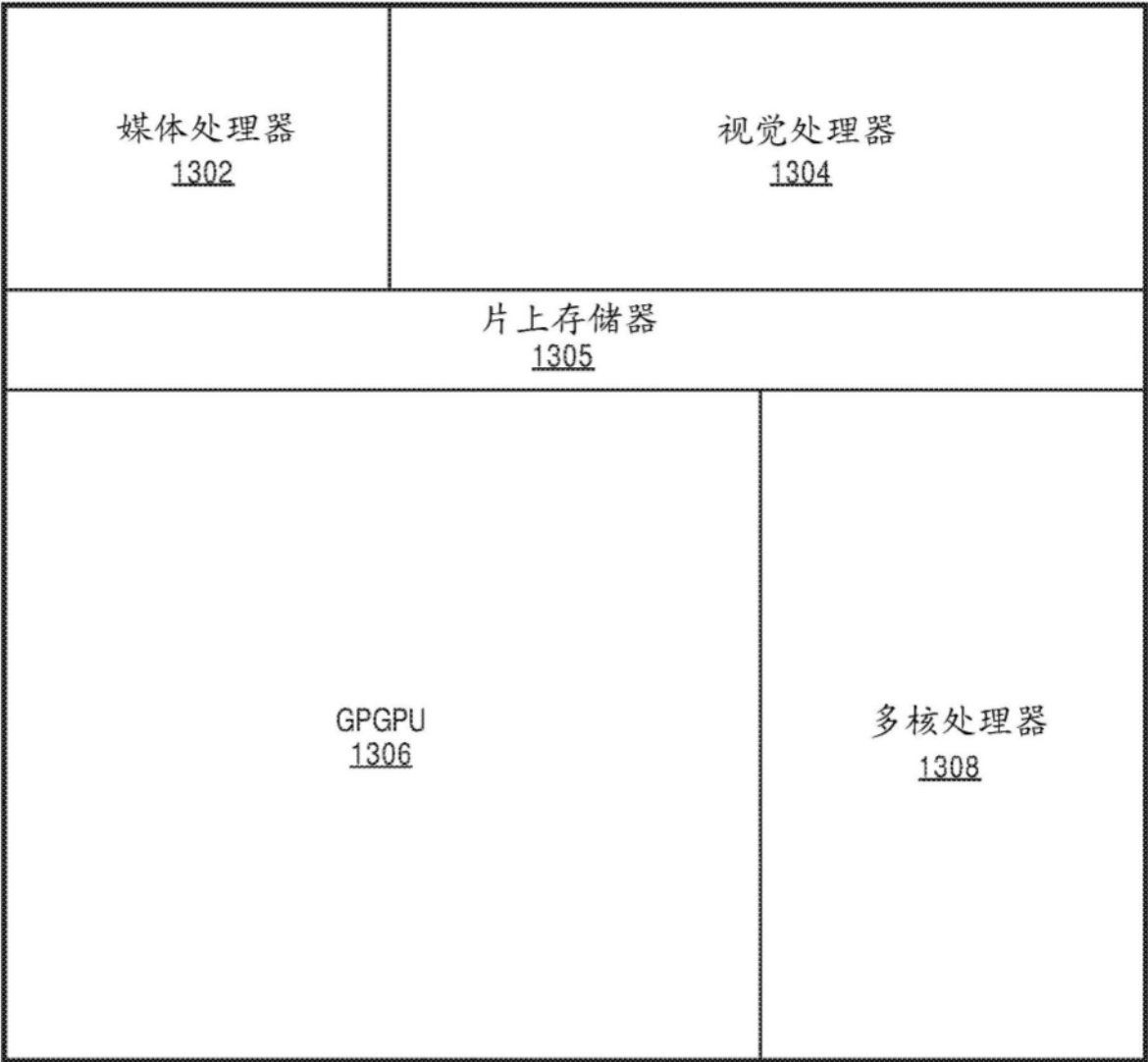


图13

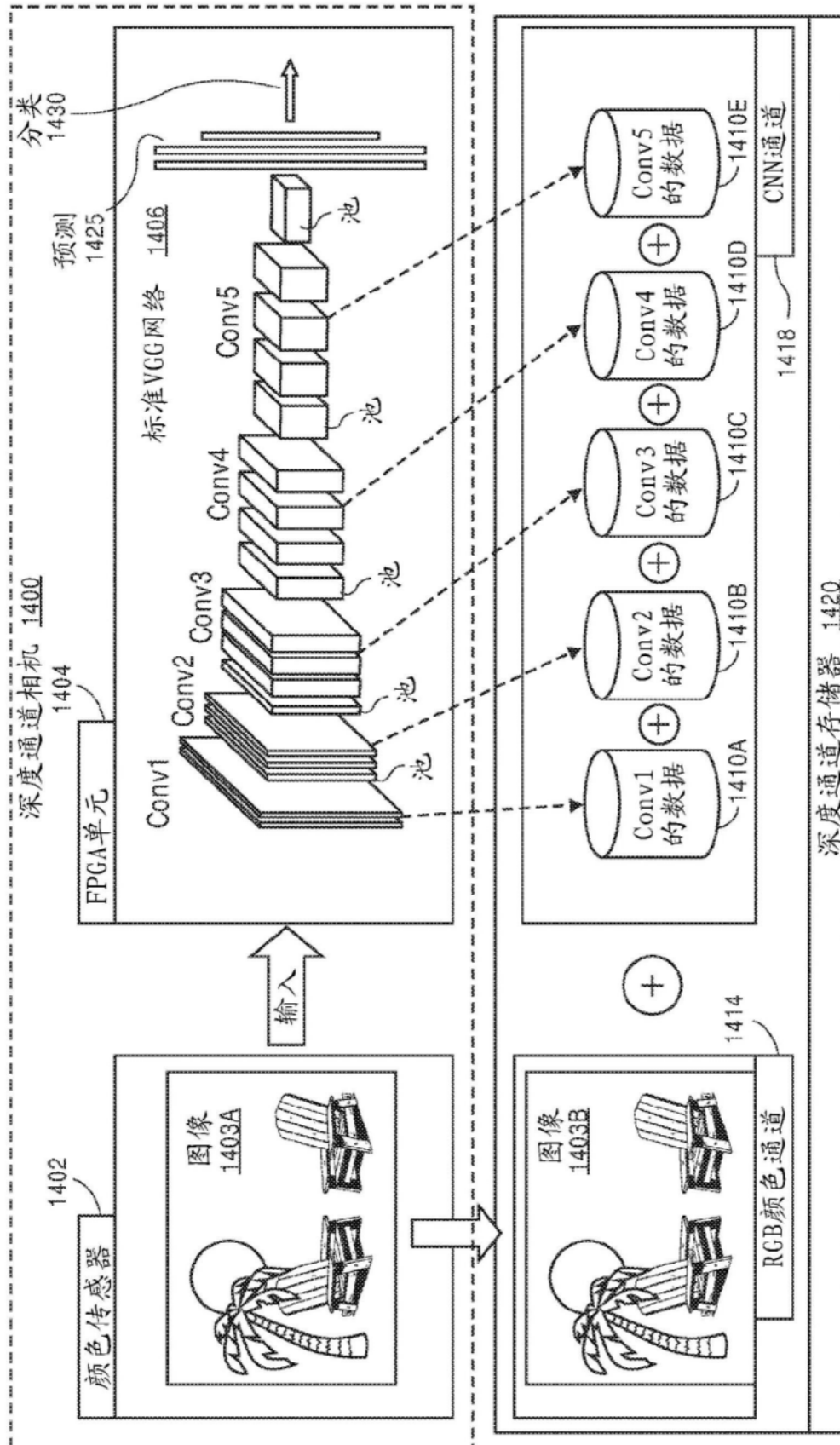


图14

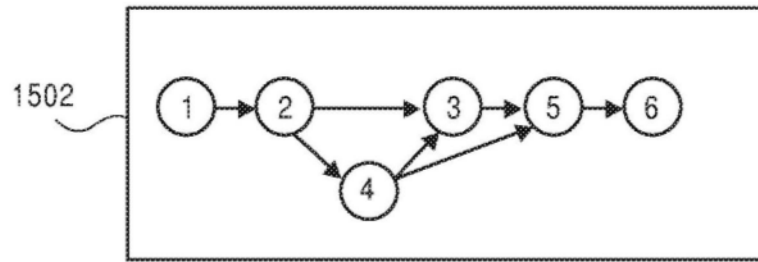


图15

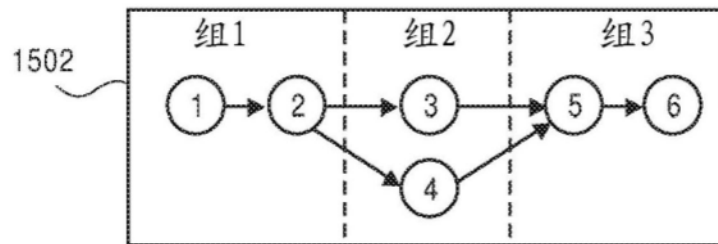


图16

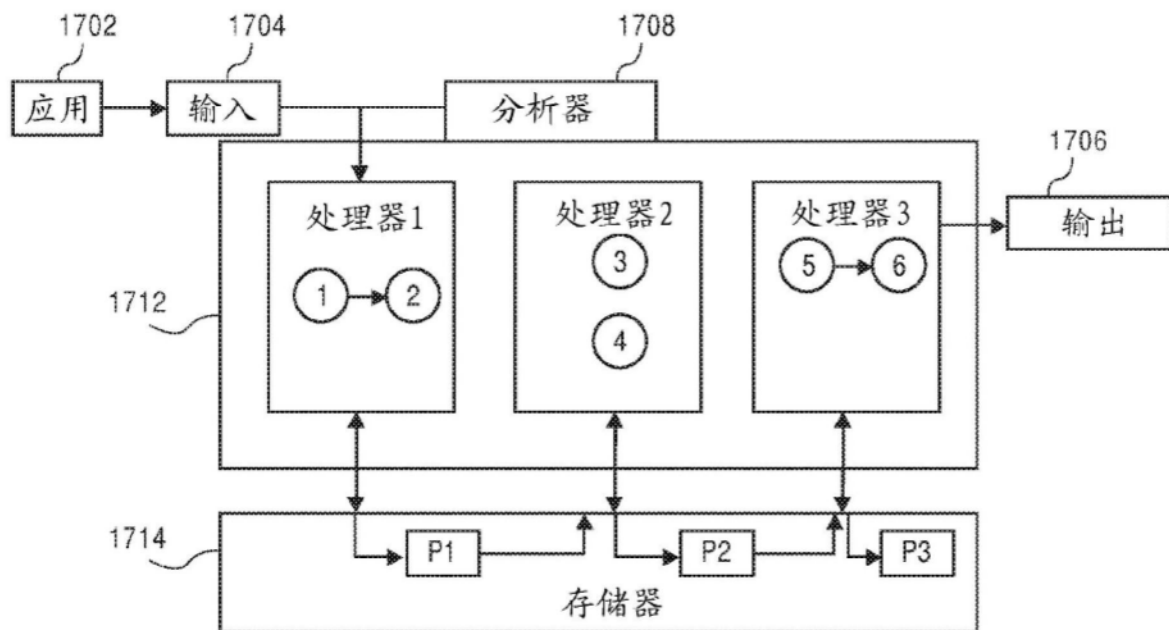


图17

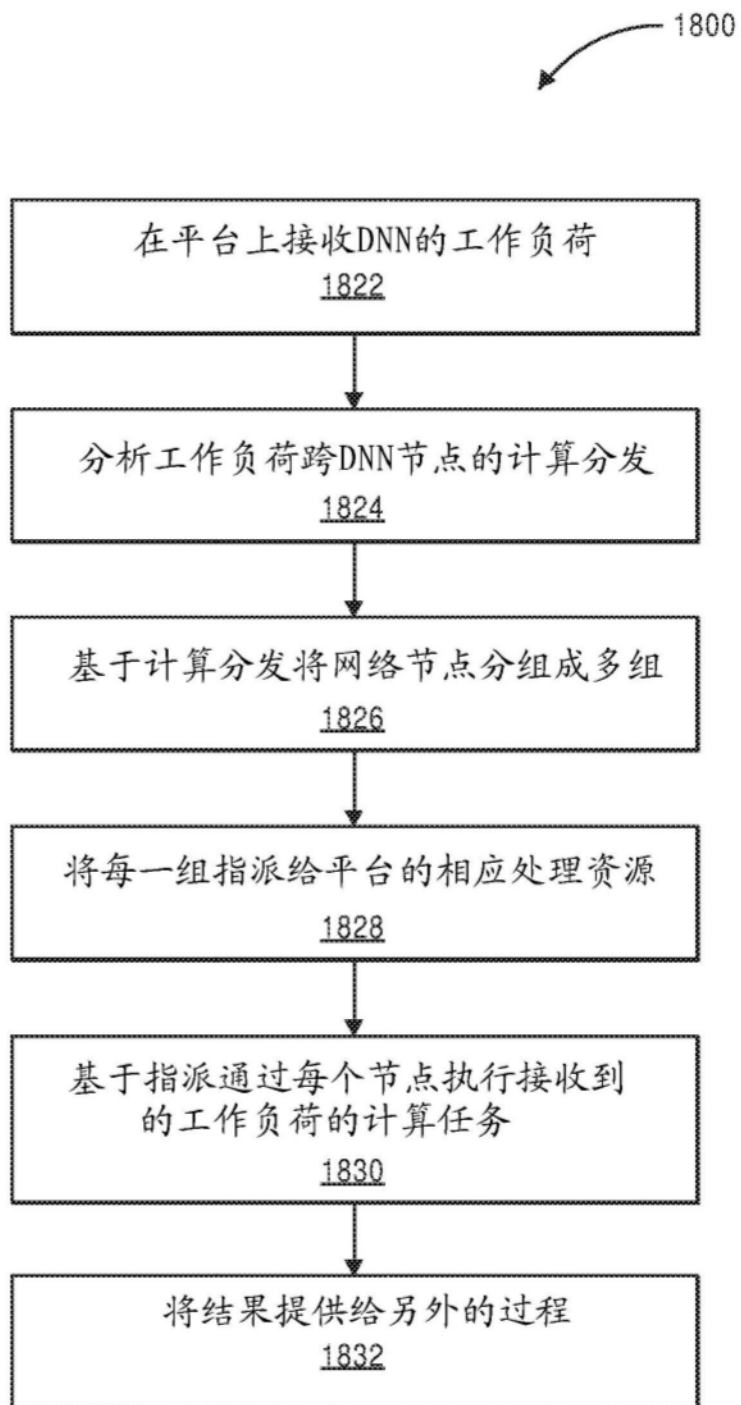


图18

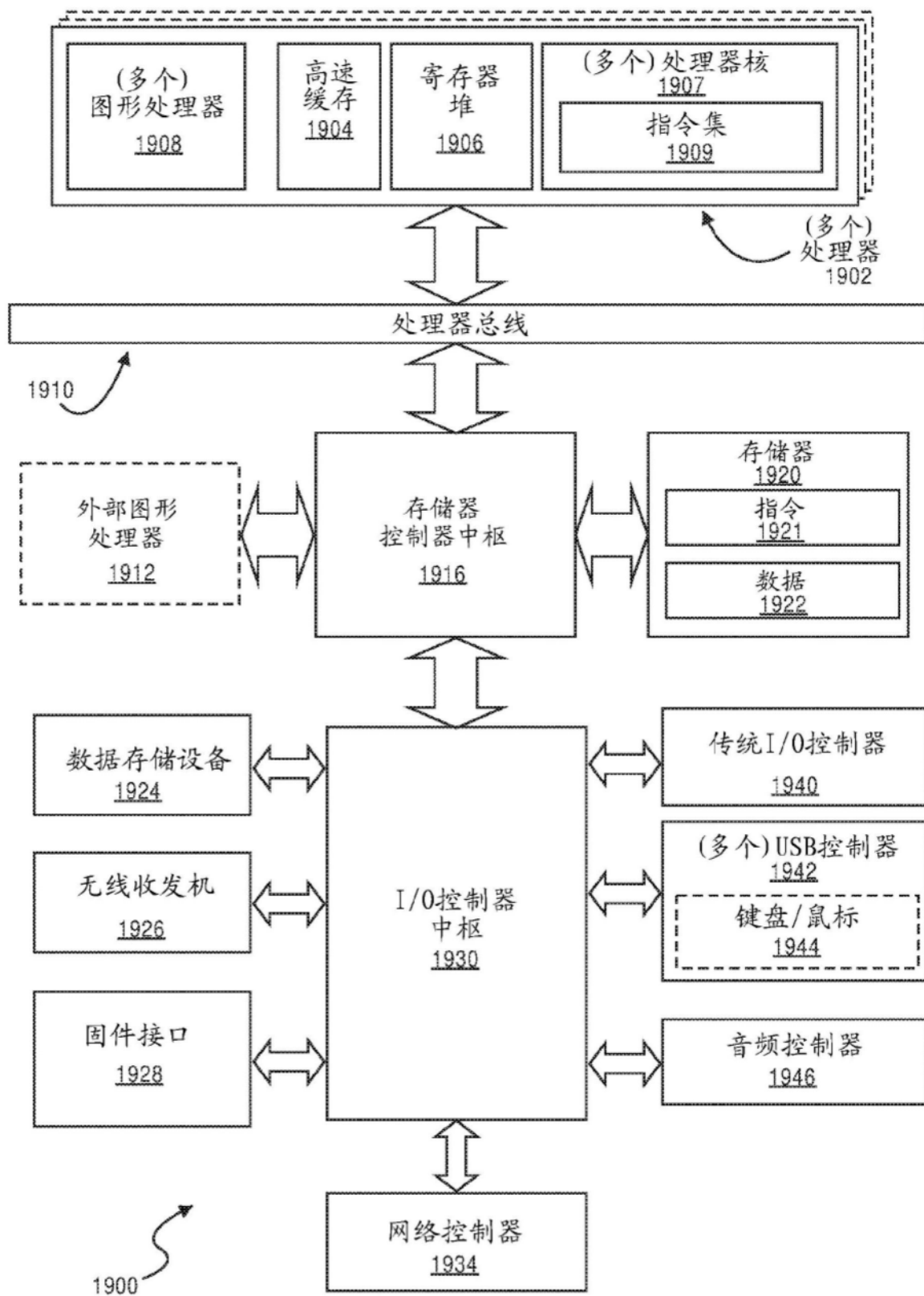


图19

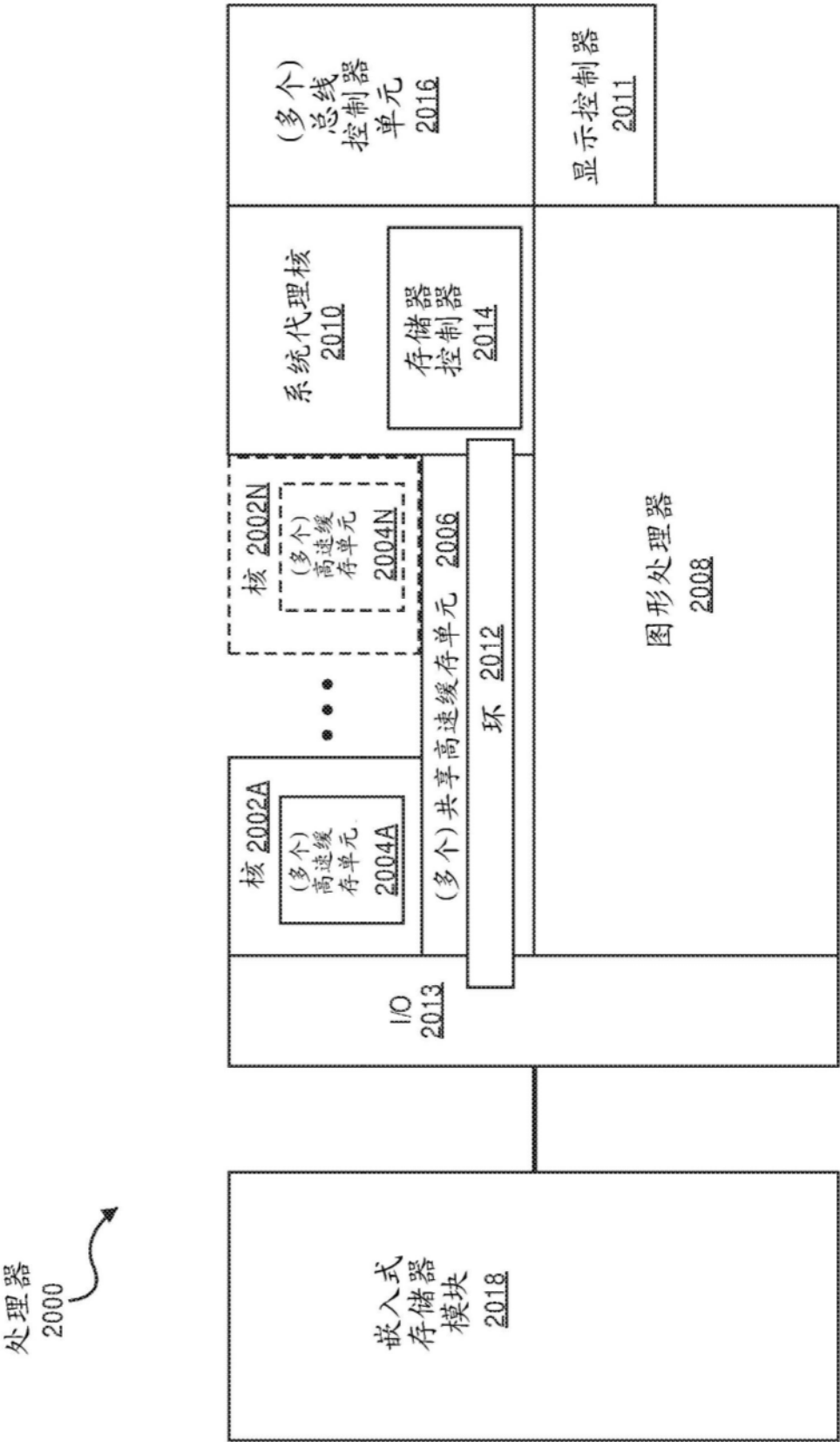


图20

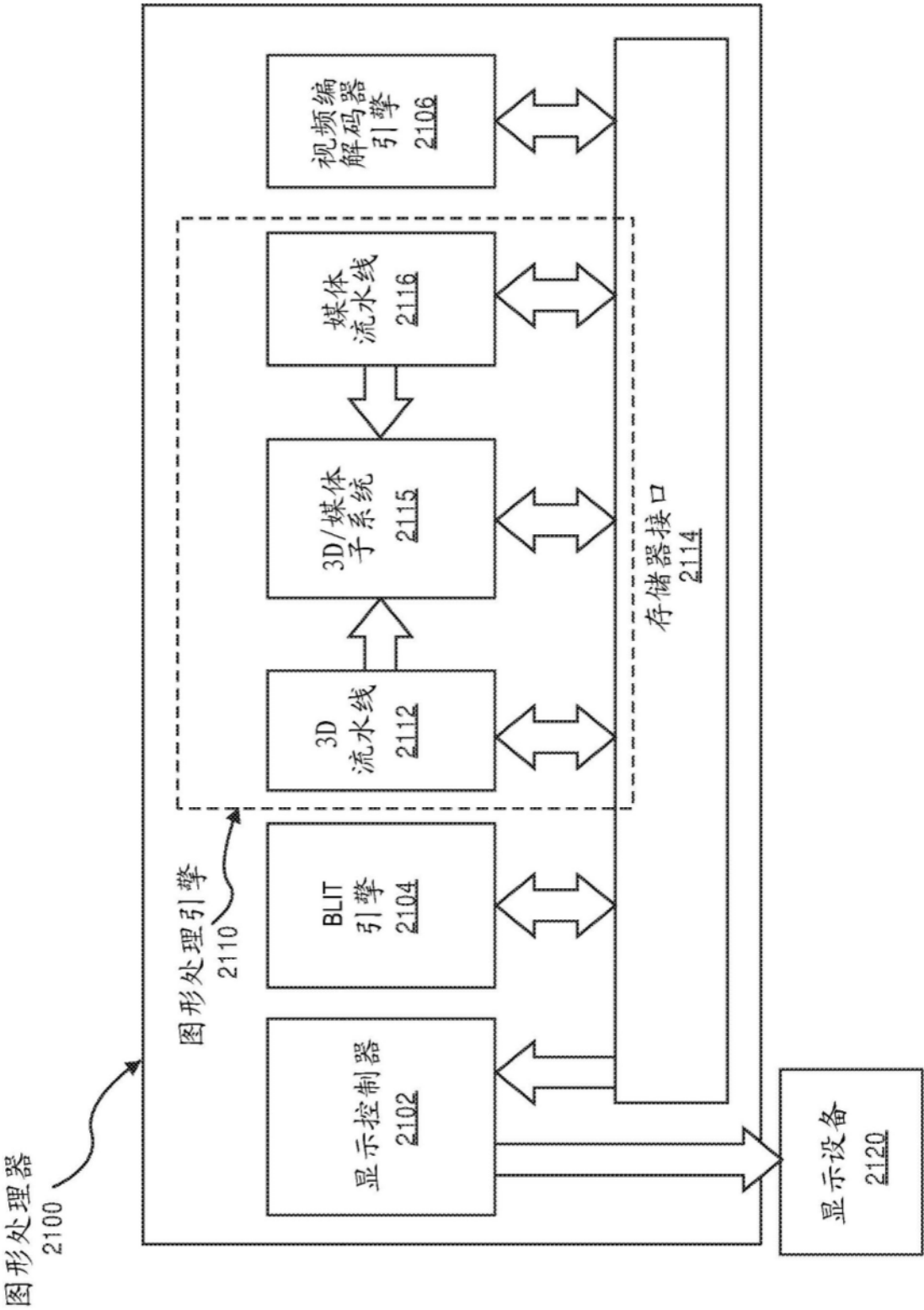


图21

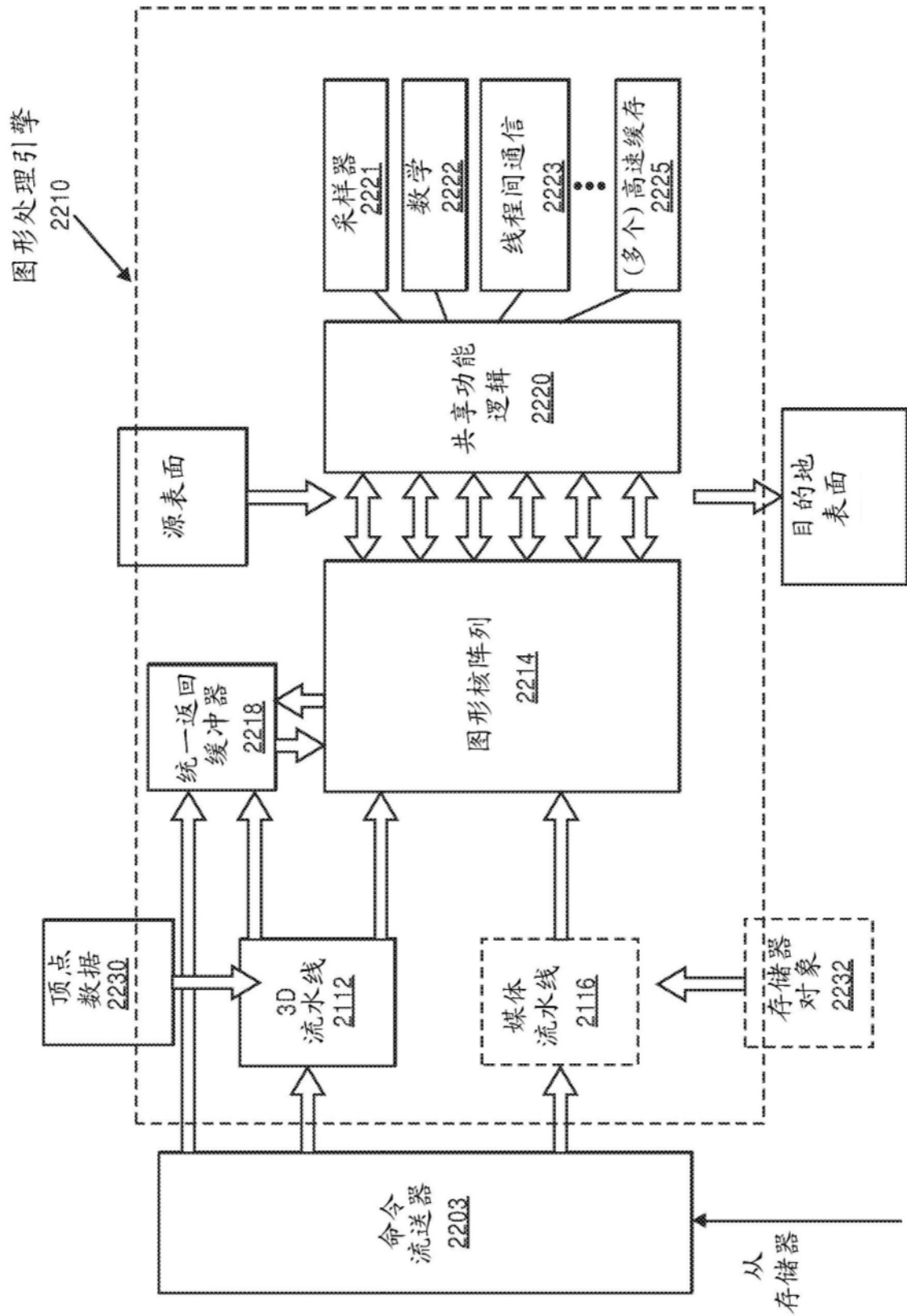


图22

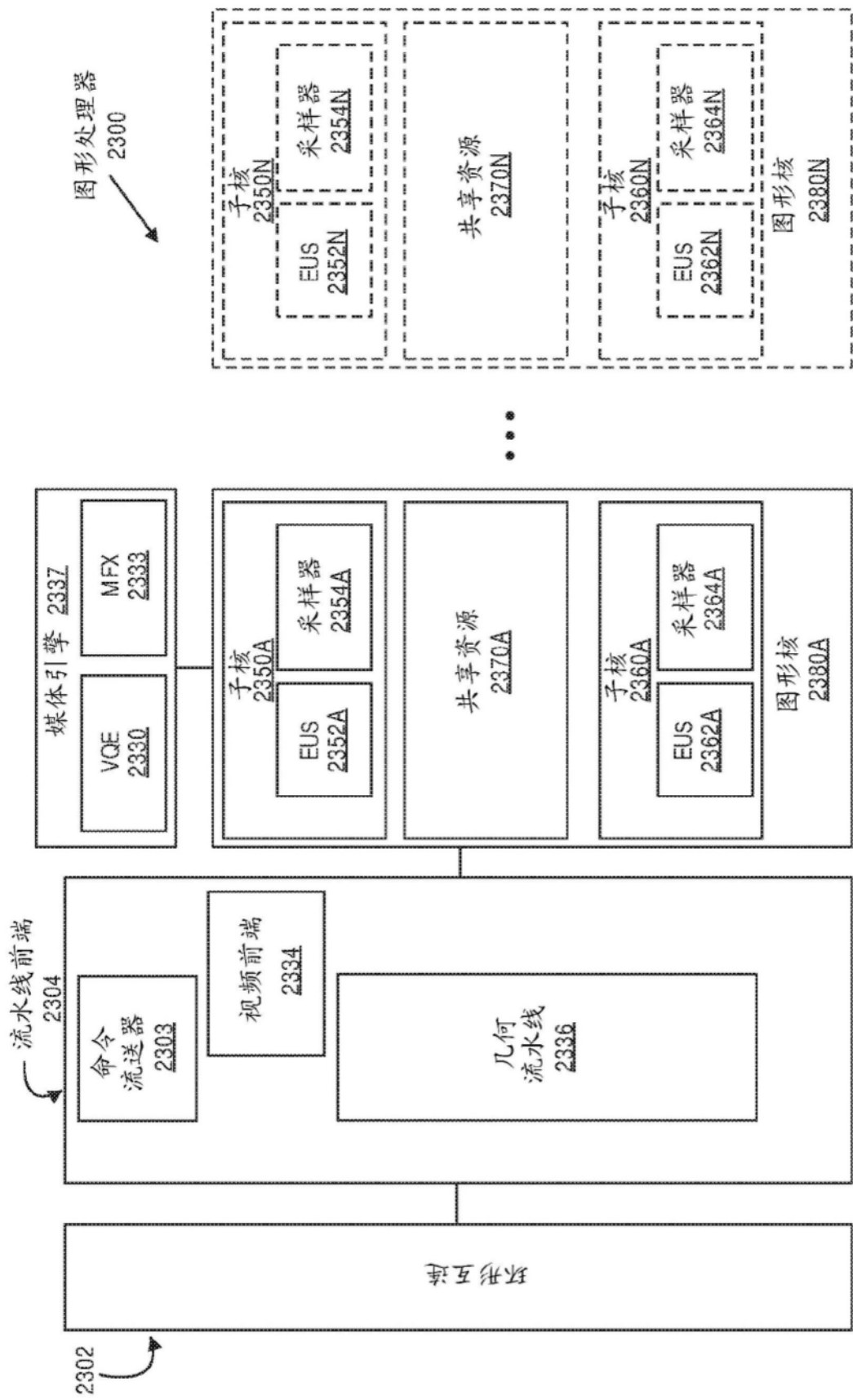


图23

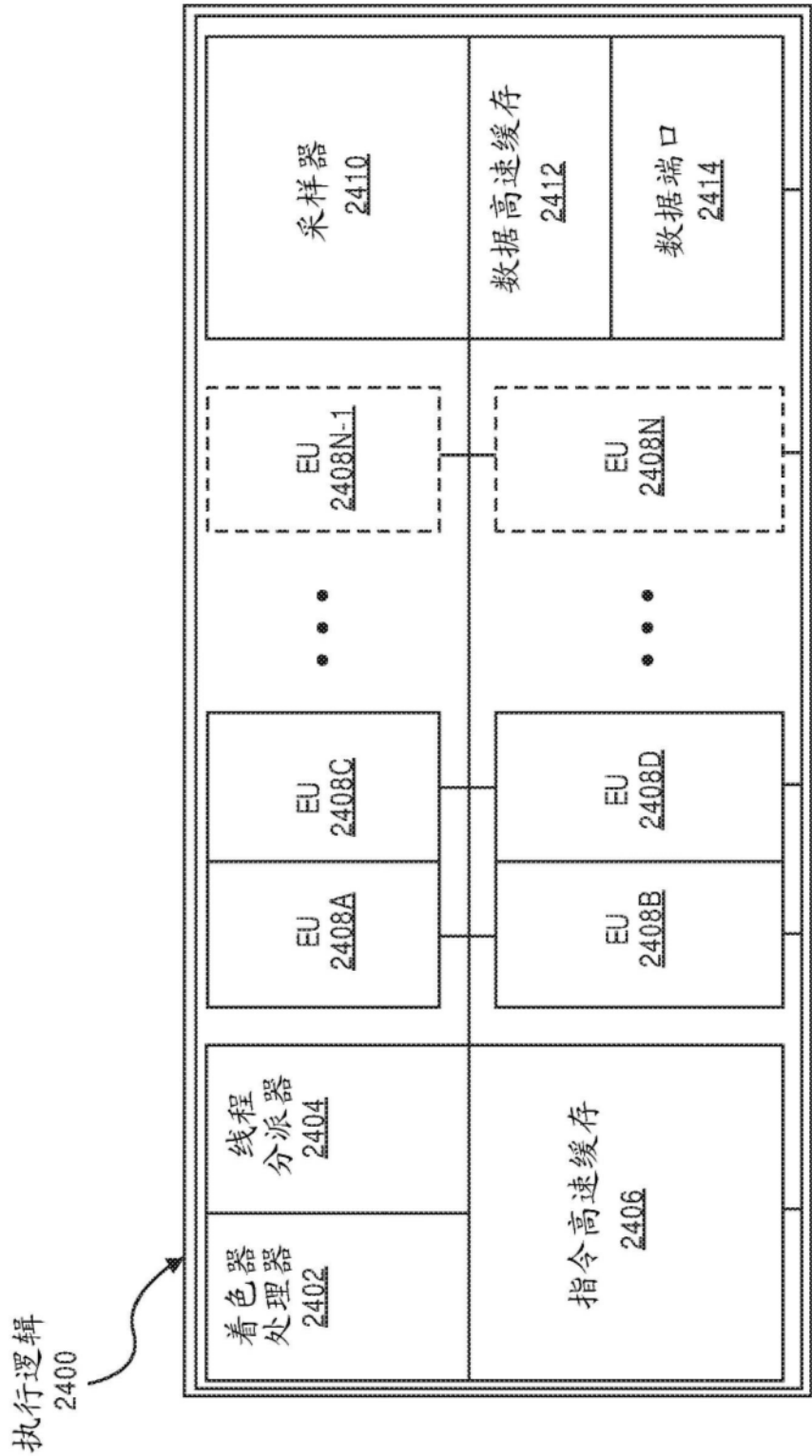


图24

图形核指令格式

2500

128位指令

2510



64位压缩指令

2530



操作码解码

2540

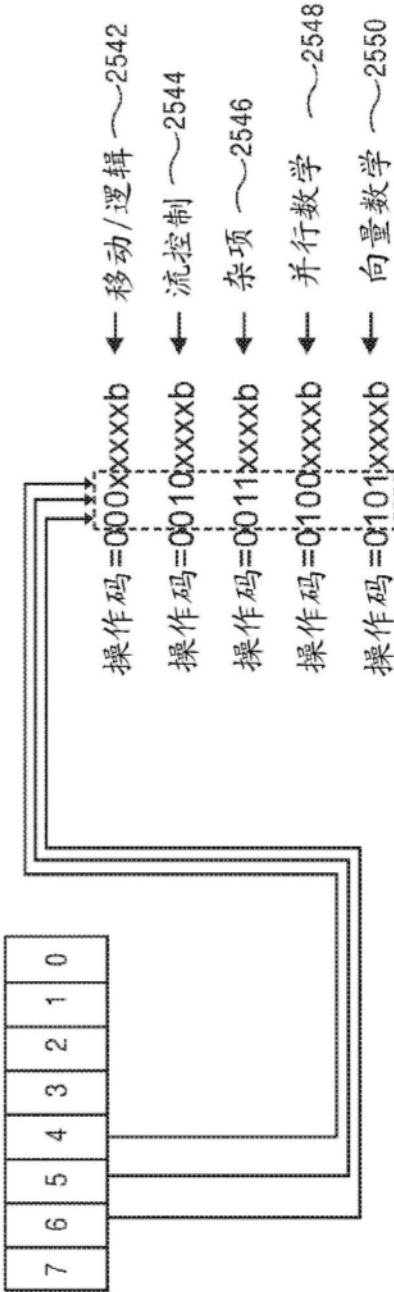


图25

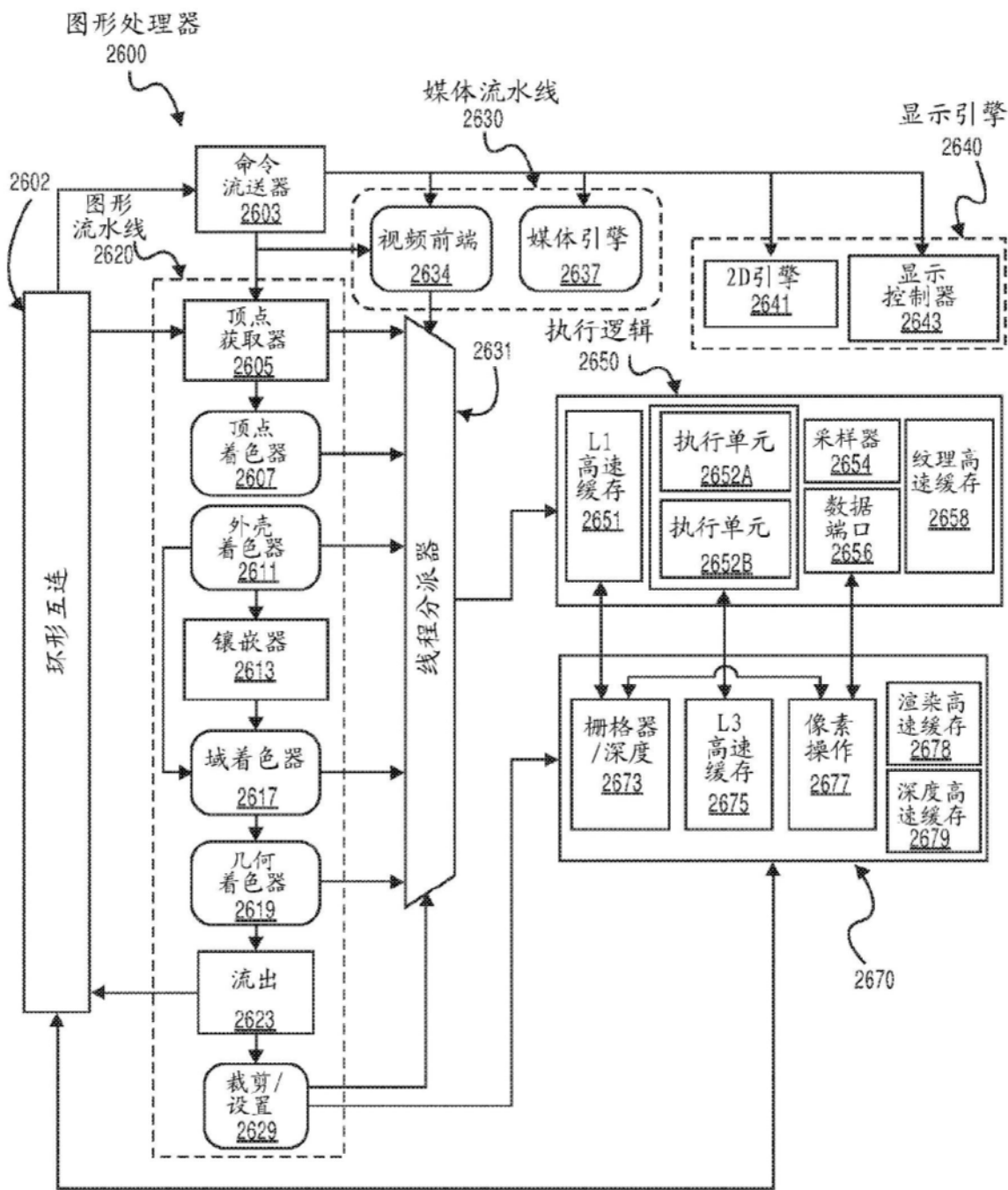


图26

样本命令格式
2700

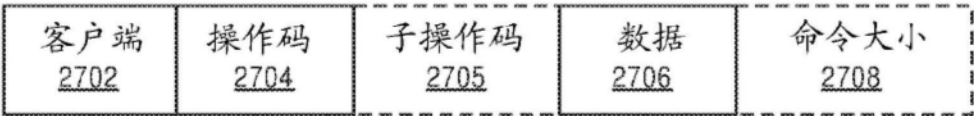


图27A

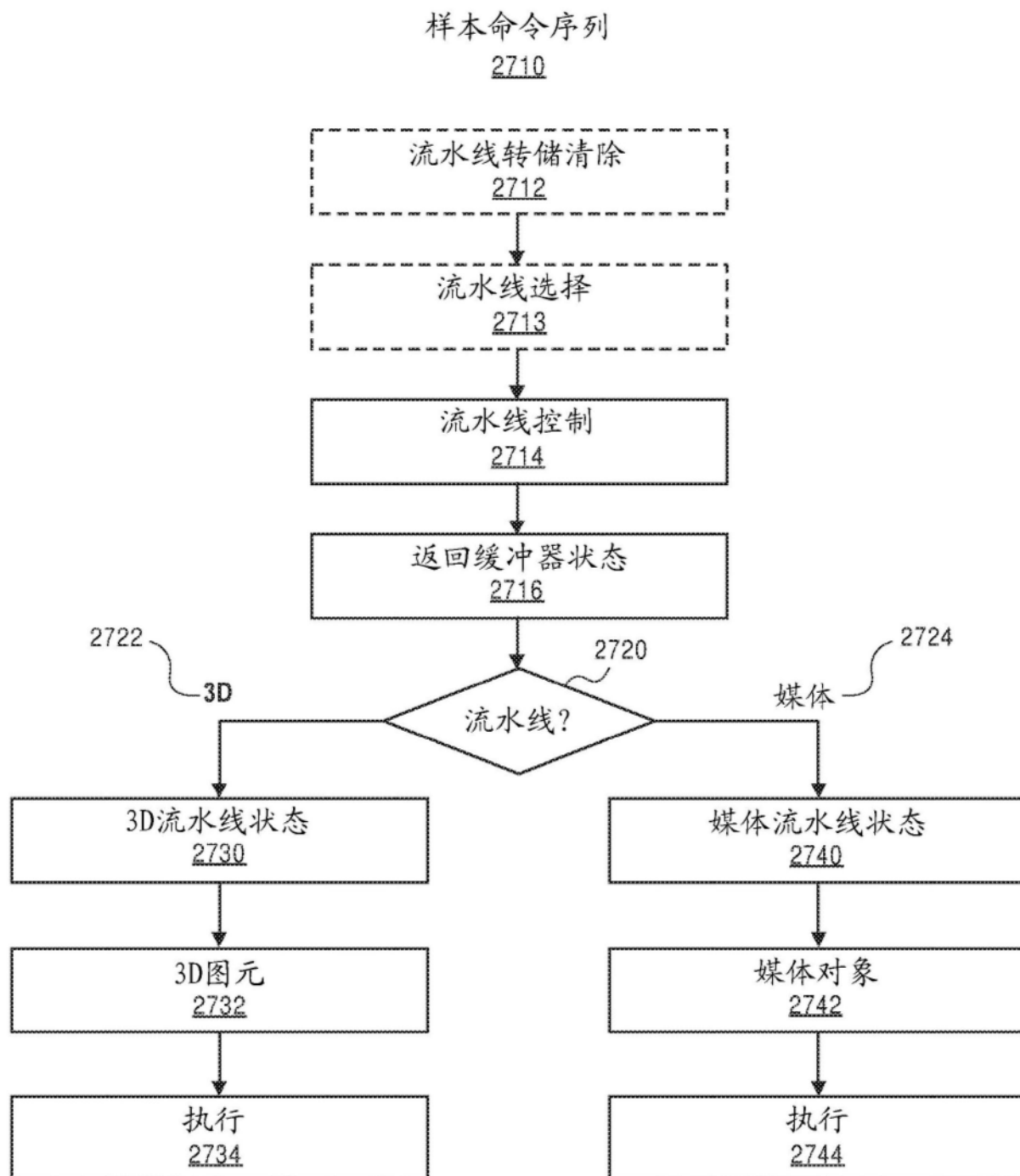


图27B

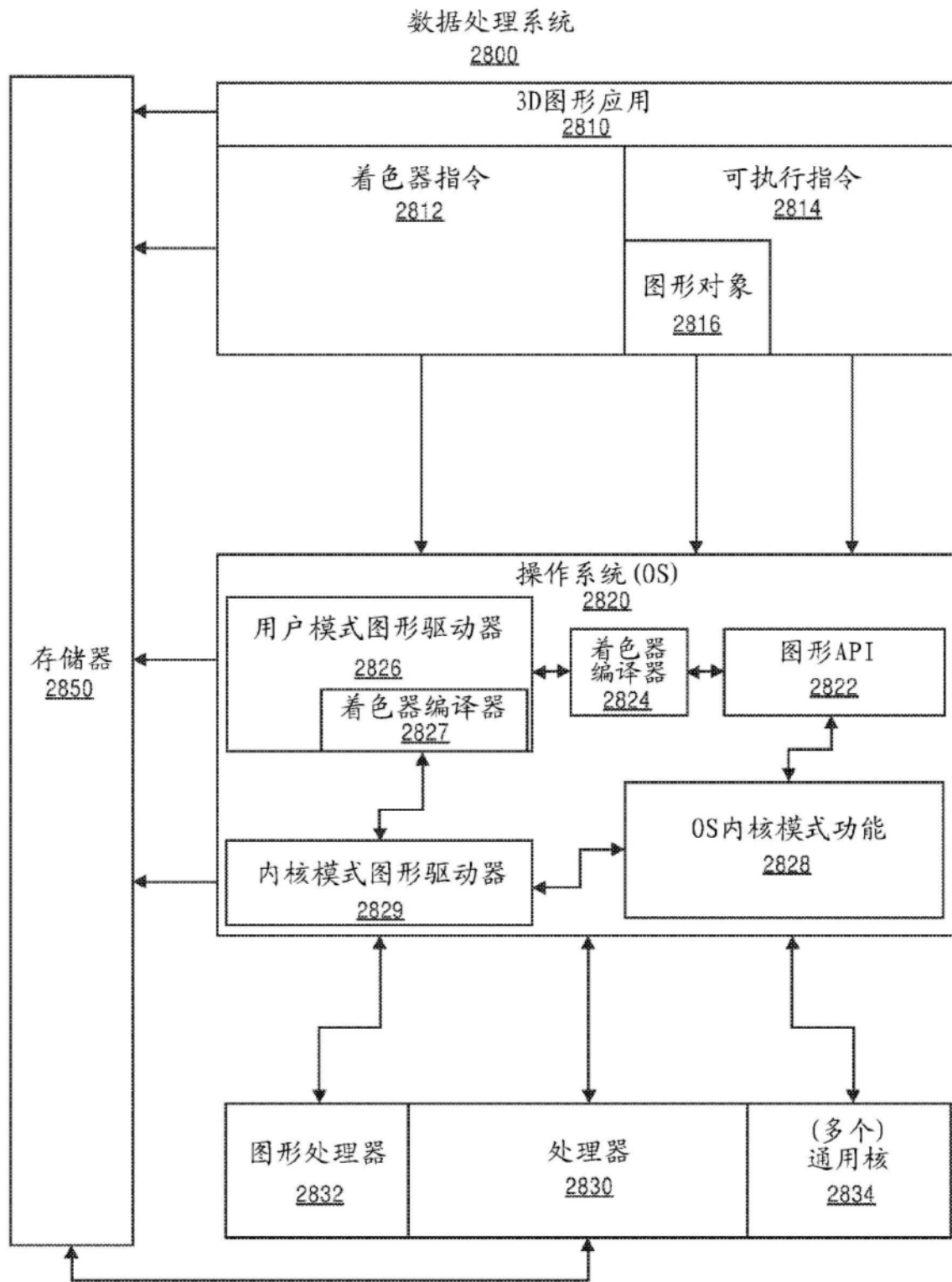


图28

IP核开发
2900

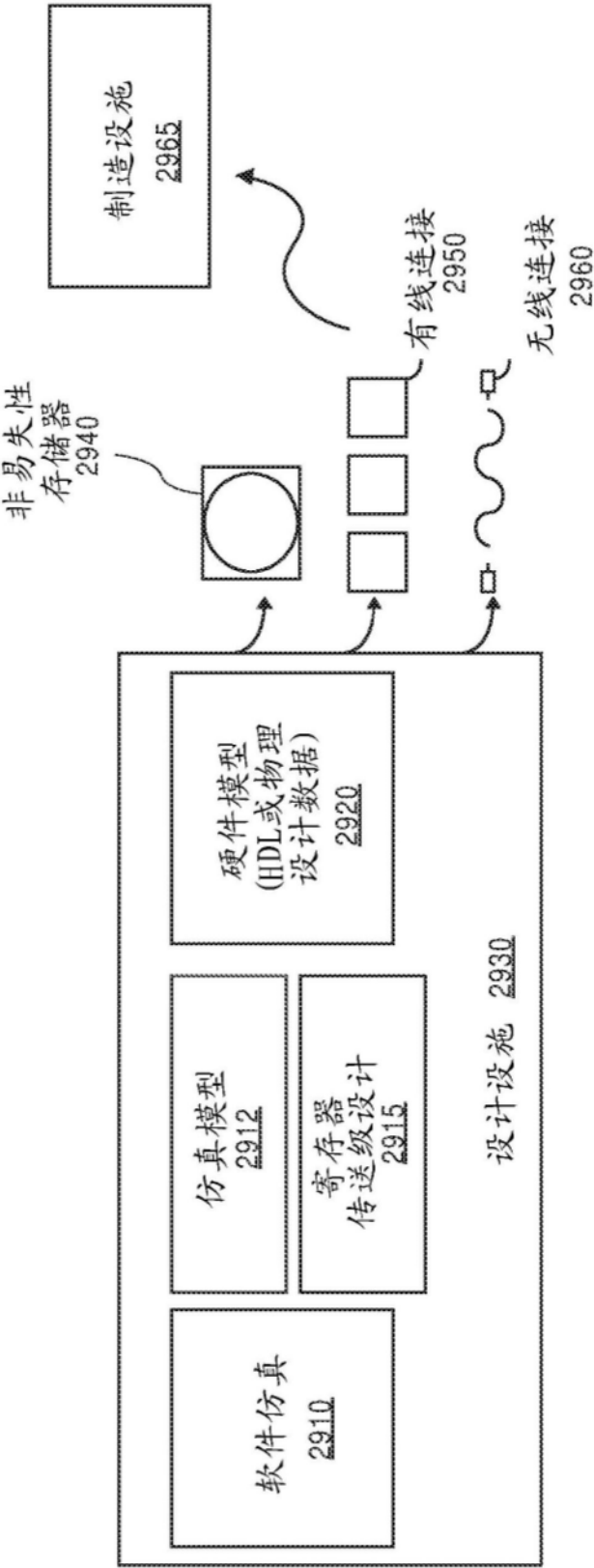


图29

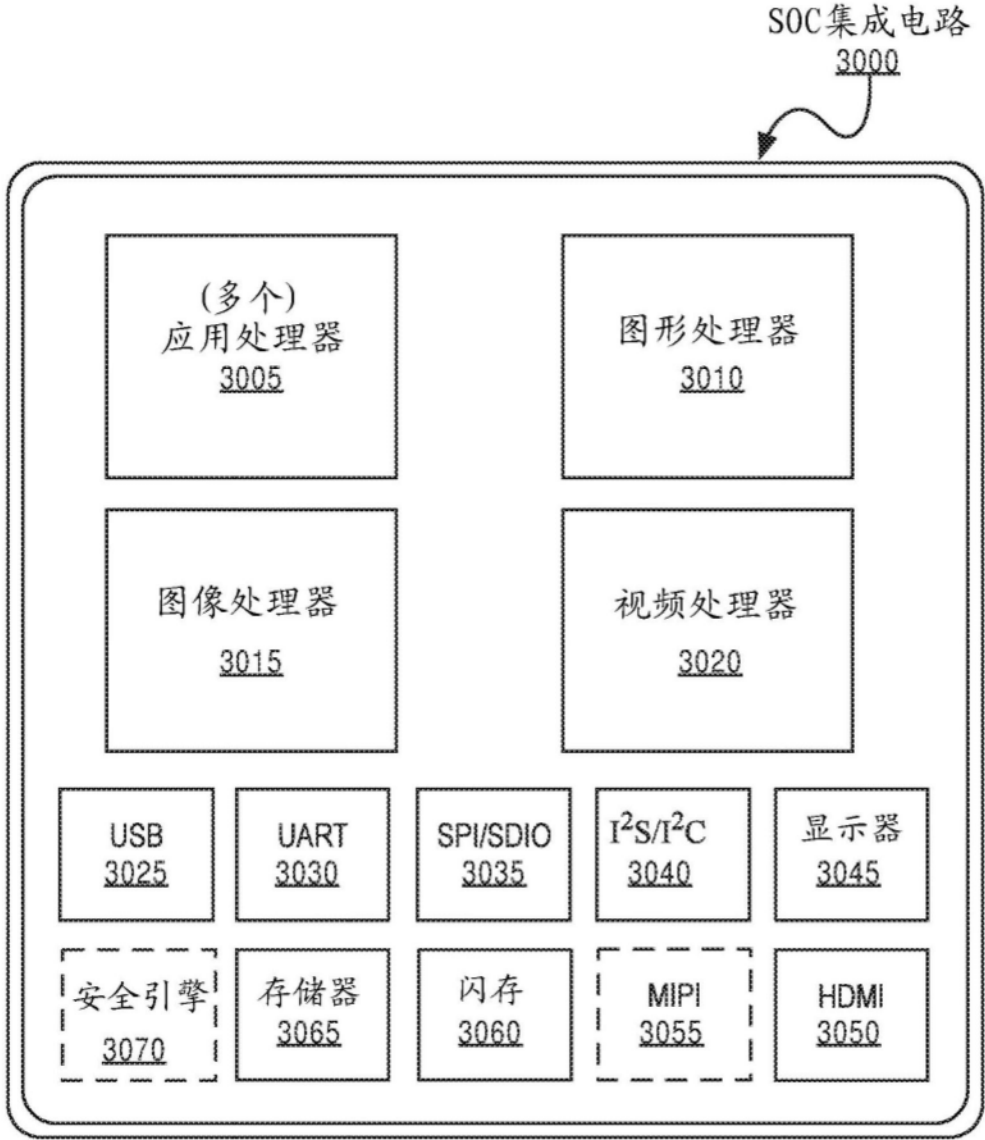


图30

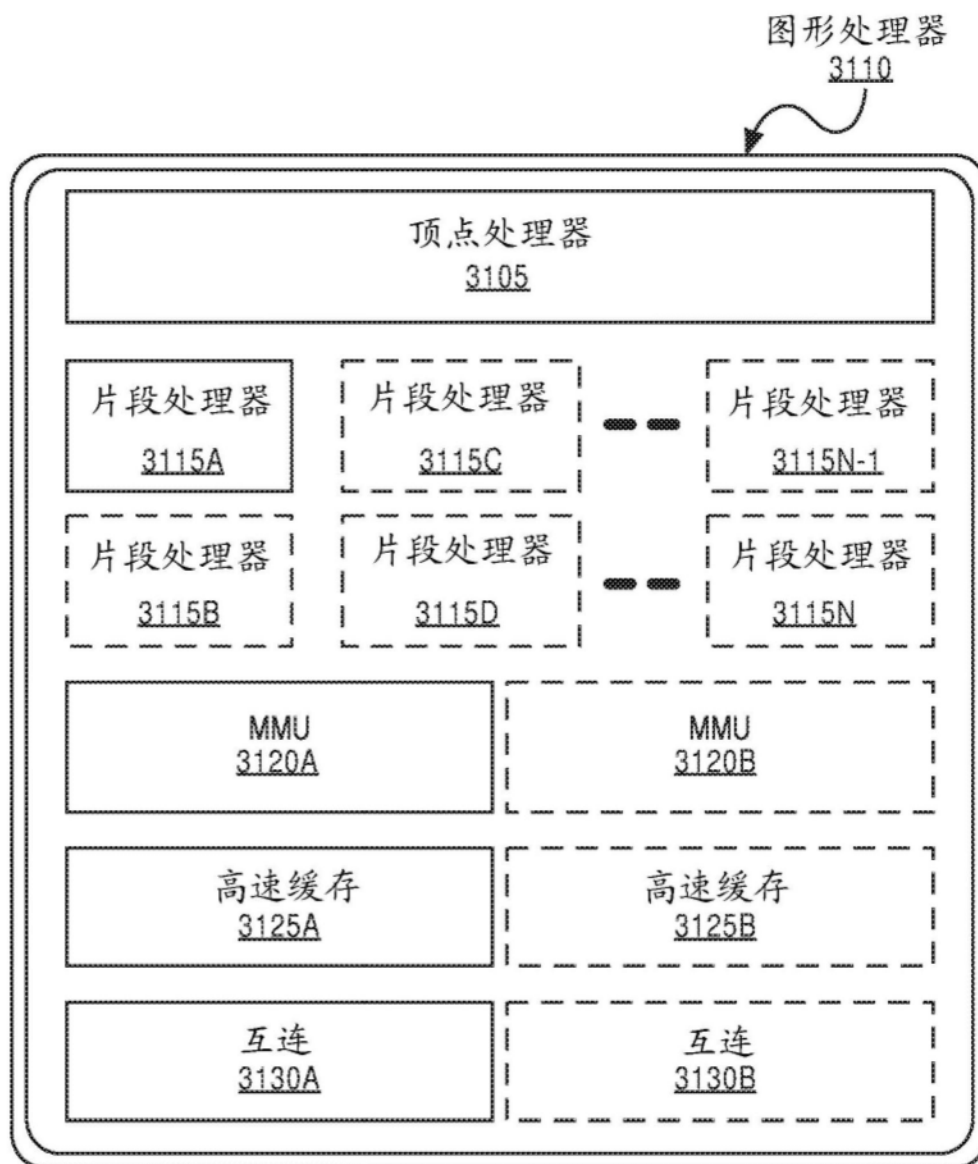


图31

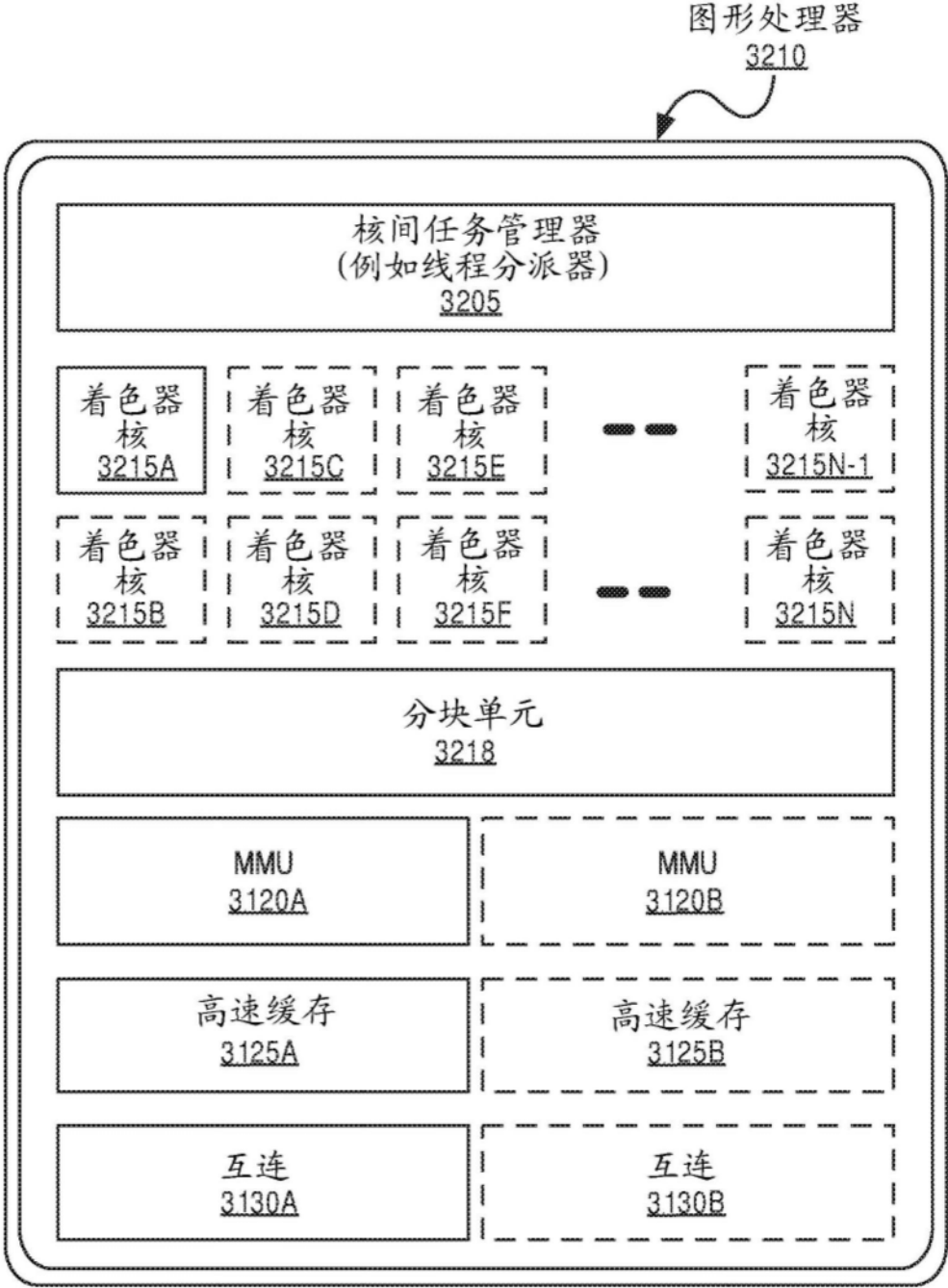


图32