(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102640115 B
(45)授权公告日 2015.03.25

(21)申请号 201080049174.7
(22)申请日 2010.09.03
(30)优先权数据
61/239,712 2009.09.03 US
12/874,134 2010.09.01 US
(85)PCT国际申请进入国家阶段日
2012.04.28
(86)PCT国际申请的申请数据
PCT/US2010/047786 2010.09.03
(87)PCT国际申请的公布数据
WO2011/028986 EN 2011.03.10
(73)专利权人 先进微装置公司
地址 美国加利福尼亚州
(72)发明人 M·曼托尔 R·迈凯利
(74)专利代理机构 北京戈程知识产权代理有限公司 11314
代理人 程伟 王锦阳
(51)Int.Cl.
G06F 9/50(2006.01)
G06F 15/163(2006.01)
G06T 15/00(2011.01)
G06T 1/20(2006.01)
(56)对比文件
US 6,252,600 B1,2001.06.26, 全文.
US 2009/0160867 A1,2009.06.25, 全文.
US 2008/0109810 A1,2008.05.08, 说明书第2页第18段至第3页第26段,图1-4.
审查员 赵传海

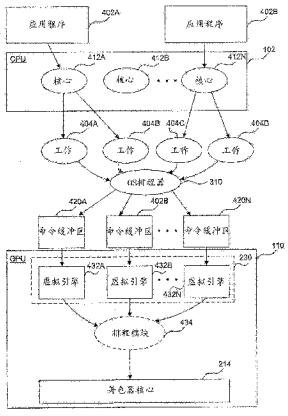
权利要求书3页 说明书10页 附图6页

(54)发明名称

包括具有多缓冲区以使在着色器核心上不同类型工作能够异步并行分派的指令处理器的图形处理单元

(57)摘要

一种包括多个虚拟引擎以及着色器核心的处理单元。该多个虚拟引擎组构成用以(i)接收来自操作系统(operation system,OS)的实质上彼此平行的多个工作,以及(ii)加载关联于该多个工作的每一个的状态数据集。该着色器核心组构成基于关联于该多个工作的每一个的状态数据集以执行该实质上平行的多个工作。该处理单元也可包括用以排程将发出给该着色器核心的该多个工作的排程模块。



CN 102640115 B

1. 一种处理单元，包括：

多个引擎，其关联于第一处理单元且组构成：

接收来自关联于第二处理单元的排程模块的多个工作，以及

加载关联于该多个工作的每一个的状态数据；以及

着色器核心，其关联于该第一处理单元且组构成接收来自该多个引擎的至少一个的该多个工作，并且基于关联于第一和第二工作的每一个的各自的状态数据以执行来自该多个工作的该第一工作，同时执行来自该多个工作的该第二工作。

2. 如权利要求 1 所述的处理单元，其中，该着色器核心包括：

第一多个处理组件，组构成基于关联于该第一工作的第一状态数据集以执行该第一工作；以及

第二多个处理组件，组构成基于关联于该第二工作的第二状态数据集以执行该第二工作。

3. 如权利要求 1 所述的处理单元，其中，该多个引擎包括：

组构成用以接收该第一工作的第一队列，以及

组构成用以接收该第二工作的第二队列。

4. 如权利要求 1 所述的处理单元，其中，该第一工作包括低延迟工作且该第二工作包括经常延迟工作。

5. 如权利要求 1 所述的处理单元，其中，该第一工作包括图形处理工作且该第二工作包括一般计算工作。

6. 如权利要求 1 所述的处理单元，其中，该着色器核心组构成基于关联于该多个工作的每一个的优先级以接收该多个工作。

7. 如权利要求 1 所述的处理单元，其中，该着色器核心包括多个处理组件，其中，该处理组件的每一个对应于该多个引擎的各自引擎，并且组构成执行来自其各自引擎的一或多个工作。

8. 如权利要求 1 所述的处理单元，其中，该着色器核心包括多个处理组件，其中，该处理组件的至少一个组构成贡献第一处理时间给该第一工作以及第二处理时间给该第二工作。

9. 一种计算装置，当该计算装置执行时，则界定一处理单元，该处理单元包括：

多个引擎，其关联于第一处理单元且组构成：

接收来自关联于第二处理单元的排程模块的多个工作，以及

加载关联于该多个工作的每一个的状态数据；以及

着色器核心，其关联于该第一处理单元且组构成接收来自该多个引擎的至少一个的该多个工作，并且基于关联于第一和第二工作的每一个的各自的该态数据以执行来自该多个工作的该第一工作，同时执行来自该多个工作的该第二工作。

10. 如权利要求 9 所述的计算装置，其中，该着色器核心包括：

第一多个处理组件，组构成基于关联于该第一工作的第一状态数据集以执行该第一工作；以及

第二多个处理组件，组构成基于关联于该第二工作的第二状态数据集以执行该第二工作。

11. 如权利要求 9 所述的计算装置，其中，该多个引擎包括：

组构成用以接收该第一工作的第一队列，以及

组构成用以接收该第二工作的第二队列。

12. 如权利要求 9 所述的计算装置，其中，该第一工作包括低延迟工作且该第二工作包括经常延迟工作。

13. 如权利要求 9 所述的计算装置，其中，该第一工作包括图形处理工作且该第二工作包括一般计算工作。

14. 如权利要求 9 所述的计算装置，其中，该着色器核心组构成基于关联于该多个工作的每一个的优先级以接收该多个工作。

15. 一种计算系统，包括：

内存；

第一处理单元；以及

总线，耦合至该内存及该第一处理单元，其中，该第一处理单元包括：

多个引擎，组构成：

接收来自关联于第二处理单元的排程模块的多个工作，以及

加载关联于该多个工作的每一个的状态数据；以及

着色器核心，组构成接收来自该多个引擎的至少一个的该多个工作，并且基于关联于第一和第二工作的每一个的各自的状态数据以执行来自该多个工作的该第一工作，同时执行来自该多个工作的该第二工作。

16. 如权利要求 15 所述的计算系统，其中，该着色器核心包括：

第一多个处理组件，组构成基于关联于该第一工作的第一状态数据集以执行该第一工作；以及

第二多个处理组件，组构成基于关联于该第二工作的第二状态数据集以执行该第二工作。

17. 如权利要求 15 所述的计算系统，其中，该多个引擎包括：

组构成用以接收该第一工作的第一队列，以及

组构成用以接收该第二工作的第二队列。

18. 如权利要求 15 所述的计算系统，其中，该第一工作包括低延迟工作且该第二工作包括经常延迟工作。

19. 如权利要求 15 所述的计算系统，其中，该第一工作包括图形处理工作且该第二工作包括一般计算工作。

20. 如权利要求 15 所述的计算系统，其中，该着色器核心组构成基于关联于该多个工作的每一个的优先级以接收该多个工作。

21. 一种用以在处理单元中处理工作的计算机实行方法，包括：

以关联于第一处理单元的多个引擎，接收多个工作，其中，该多个引擎接收来自关联于第二处理单元的排程模块的该多个工作；

加载关联于该多个工作的每一个的状态数据；

以关联于该第一处理单元的着色器核心，接收来自该多个引擎的至少一个的该多个工作；以及

以该着色器核心，基于关联于第一和第二工作的每一个的状态数据，以执行来自该多个工作的该第一工作，同时执行来自该多个工作的该第二工作。

22. 如权利要求 21 所述的计算机实行方法，其中，该执行包括：

基于关联于该第一工作的第一状态数据集以执行在该着色器核心的第一多个处理组件中的该第一工作；以及

基于关联于该第二工作的第二状态数据集以执行在该着色器核心的第二多个处理组件中的该第二工作。

23. 如权利要求 21 所述的计算机实行方法，其中，以该多个引擎接收该多个工作包括：

使该第一工作队列于第一队列；以及

使该第二工作队列于第二队列。

24. 如权利要求 21 所述的计算机实行方法，其中，该第一工作包括低延迟工作且该第二工作包括经常延迟工作。

25. 如权利要求 21 所述的计算机实行方法，其中，该第一工作包括图形处理工作且该第二工作包括一般计算工作。

26. 如权利要求 21 所述的计算机实行方法，其中，以该着色器核心接收来自该多个引擎的至少一个的该多个工作包括以该着色器核心，基于关联于该多个工作的每一个的优先级以接收该多个工作。

27. 一种用以提供工作给处理单元的计算机实行方法，包括：

从第一处理单元，接收来自一个或更多应用程序的多个工作，其中，各个工作包括优先类型的指示；以及

提供该多个工作及关联于各个工作的该优先类型的指示至第二处理单元，其中，该第二处理单元包括：

多个引擎，其组构成接收该多个工作以及加载关联于该多个工作的每一个的状态数据；以及

着色器核心，组构成接收来自该多个引擎的至少一个的该多个工作，并且基于关联于第一和第二工作的每一个的各自的状态数据以执行来自该多个工作的该第一工作，同时执行来自该多个工作的该第二工作。

包括具有多缓冲区以使在着色器核心上不同类型工作能够异步并行分派的指令处理器的图形处理单元

技术领域

[0001] 本揭示发明大致关于在计算机系统中执行的计算作业，尤其是关于一种例如图形处理单元 (graphic-processing unit，GPU) 的处理单元，该处理单元执行计算作业及其应用程序。

背景技术

[0002] GPU 为适用于执行例如图形处理工作的数据并行计算工作复杂的集成电路。举例而言，GPU 可执行例如视讯游戏应用程序之终端使用者应用程序所要求的图形处理工作。该 GPU 可为离散的（亦即，分开的）装置及 / 或封装件，或可被包含在相同的装置及 / 或封装件中作为另一处理器（例如，中央处理器 (central processing unit，CPU)）。举例而言，GPU 频繁地被整合至路由 (routing) 或桥接 (bridge) 装置中，例如北桥装置。

[0003] 终端使用者应用程序与 GPU 之间具有数个软件层。该终端使用者应用程序与应用程序编程接口 (Application programming interface，API) 沟通。API 允许该终端使用者应用程序以标准化的格式输出图形数据及命令，而非取决于 GPU 的格式。数种 API 是商业市场上可取得的，包含位于华盛顿州雷蒙德的微软公司 (Microsoft Corporation of Redmond，Washington) 所发展的DirectX®以及科纳斯组织 (Khronos Group) 所发布的OpenGL®。API 与驱动程序沟通。该驱动程序将接受自该 API 的标准编码 (code) 转译为该 GPU 可了解的指令的原生格式。该驱动程序典型地是由该 GPU 的制造者所撰写。该 GPU 随后执行来自该驱动程序的指令。

[0004] 由 GPU 所执行的图形处理工作典型地包含复杂的数学计算，例如矩阵和向量运算。为了执行一个单一的图形处理工作，GPU 可能需要执行多个不同的线程（指令的序列）。各线程可包括一着色程序 (shader program)，例如几何着色程序、像素着色程序、顶点着色程序等等。各线程（例如，着色程序）典型地与状态数据集（例如，纹理处理、着色常数、转换矩阵等等）相关联，该状态数据集区域性地储存在该 GPU 的数据储存单元中。该区域性的储存的状态数据集称为上下文 (context)。

[0005] 为了有效率地执行单一图形处理工作的各种线程（例如，着色程序），GPU 包含一处理组件所组成的矩阵，称为着色器核心 (shader core)。该处理组件所组成的矩阵被组织成单指令多数据 (single-instruction，multiple-data，SIMD) 装置。多个线程（例如，着色程序）可同时被发出至该着色器，伴随着将执行各个线程（例如，着色程序）所需要的数据平行地分配给该着色器的不同处理组件。该不同处理组件可随后平行地执行该数据上的作业。以此方式，GPU 可较典型的中央处理器 (CPU) 更快速地执行图形处理工作所要求的复杂数学计算。因此，若计算系统包含 GPU，图形处理工作（及其它类型的数据平行处理工作）典型地传送至该 GPU 而非 CPU。

[0006] 为了传送工作至 GPU，操作系统 (operation system，OS) 排程器将该工作储存在命令缓冲区 (command buffer) 中。习知的 GPU 一次处理一个缓冲区。该 OS 排程器逐次地

将工作放置该命令缓冲区中，且该 GPU 典型地依据各该工作被放置在命令缓冲区中的顺序处理各该工作。然而，在某些例子中，该 GPU 可不依据各该工作被放置在命令缓冲区中的顺序处理各该工作。举例而言，GPU 可中断第一工作的执行，以实行在该命令缓冲区中被放置在该第一工作之后的更重要的（例如，低延迟 (low-latency)）工作。

[0007] 为了于该第一工作在 GPU 的着色器核心中结束完成前执行更重要的（例如，低延迟）的工作，习知的 GPU 执行上下文切换 (context switch)。亦即，将关联于该第一工作的线程的状态数据交换至该习知的 GPU 所维持的备用储存单元中，再将关联于该更重要的（例如，低延迟）的工作的线程（例如，着色程序）的新状态数据撷取并放置到该着色器核心的数据储存单元中。该着色器核心随后基于在该数据储存单元中的该新状态数据执行该更重要的（例如，低延迟）工作。在该更重要的（例如，低延迟）工作完成执行之后，从数据储存单元中清空在关联于该更重要的（例如，低延迟）工作的线程的状态数据，且将来自该第一工作的线程的状态数据交换回该着色器核心的数据储存单元中。然后，该着色器核心可继续执行该第一工作的线程。

[0008] 虽然上下文切换使 GPU 能够不依据各该工作被放置在命令缓冲区中的顺序处理各该工作，但因为数个理由，上下文切换仍是有问题的。就初步的问题而言，执行上下文切换需要大量的时间，因此限制了 GPU 的表现。此外，上下文切换需要额外的区域内存（例如，备用储存单元）以储存被切换的上下文。该额外的区域内存占用宝贵的芯片面积，导致了较大的 GPU。

[0009] 除需要大量的时间和面积之外，上下文切换使 GPU 在处理低延迟、高度优先的工作上变得无效率。为了准备着色器核心以执行该低延迟、高度优先的工作，习知的 GPU 必须执行上下文切换。关联于上下文切换的时间（例如，数百个时脉循环）使得用以执行该低延迟、高度优先的工作的有效时间 (effective time) 相对的长，即使执行该低延迟、高度优先的工作的实际时间 (actual time) 可能相对的短（例如，数十个时脉循环）。

[0010] 综上所述，需要不使用上下文切换，而可有效率地处理重要（例如，低延迟）工作的处理单元。

发明内容

[0011] 本发明之实施例通过提供方法、设备以及系统以能够进行异步工作分派及其应用以满足前述的需求。

[0012] 举例而言，本发明之实施例提供一种包含多个虚拟引擎和着色器核心的处理单元。该多个虚拟引擎组构成用以 (i) 接收来自操作系统 (OS) 的实质上彼此平行的多个工作，以及 (ii) 加载关联于该多个工作的每一个的状态数据集。该着色器核心组构成基于关联于该多个工作的每一个的该状态数据集以执行该实质上平行的多个工作。该处理单元还可包含排程模块，用以排程将发出给该着色器核心的该多个工作。

[0013] 在另一实施例中，处理单元界定在软件中。在本实施例中，一种包括含有指令的计算机可读取储存媒体的计算机程序产品，该指令若在计算装置上执行时，界定一处理单元。

[0014] 在又一实施例中，该处理单元被包含在一计算系统中。在本实施例中，该计算系统包含内存、第一处理单元、第二处理单元以及耦合至该内存、该第一处理单元及该处理单元的总线。一范例计算系统可包含，但不限于超级计算机、桌上型计算机、笔记型计算机、游戏

机(video-game console)、嵌入式装置、手持装置(例如,手机、智能型手机、MP3播放器、相机、GPS装置等等)或其它包含有或配置成含有处理单元的装置。

[0015] 本发明的另一实施例提供一种用以在处理单元中处理工作的计算机实行方法。该计算机实行方法包含数个作业。在第一作业中,接收来自操作系统(OS)的彼此平行的多个工作。在第二作业中,加载关联于该多个工作的每一个的状态数据集,在第三作业中,在着色器核心中基于关联于该多个工作的每一个的该状态数据集以执行该实质上平行的多个工作。该计算机实行方法还可包含排程将发出给该着色器核心的该多个工作。

[0016] 本发明的又一实施例提供一种用以提供工作给处理单元的计算机实行方法。该方法包含数个作业。在第一作业中,接收来自一个或更多应用程序的多个工作,其中,各工作包含优先类型之指示。在第二作业中,提供该多个工作及关联于各工作的该优先类型之指示至该处理单元。在一实施例中,若指令由一计算装置所执行,储存在计算机程序产品的计算机可读取媒体上的指令可引发该计算装置执行本方法。

[0017] 本发明的特征及优势,连带本发明的各种实施例的结构和运作,在以下参考附加图标作详细地描述。应该了解到本发明并不以此处所描述的特定实施例为限。此处所揭示之该些特定实施例仅为了例示之目的。依据此处教示,其它实施例对于熟悉相关技术领域者而言是显而易见的。

附图说明

[0018] 合并至且构成本说明书的一部份的附加图标系例示本发明之实施例,连同前述发明内容提供的一般叙述以及下列具体实施方式提供的详细说明,以协助解释本揭示发明的原理且使熟悉相关技术领域者能够制作及使用本发明。

[0019] 图1例示根据本揭示发明的实施例的范例计算系统的方块图。

[0020] 图2例示根据本揭示发明的实施例的范例GPU的方块图。

[0021] 图3A及3B系例示根据本揭示发明的实施例的用以发出工作至GPU的虚拟引擎的范例工作流程的方块图。

[0022] 图4例示根据本揭示发明的实施例的用以发出工作至GPU的虚拟引擎的更详细范例工作流程的方块图。

[0023] 图5描绘本揭示发明的实施例可被实现的范例计算系统。

[0024] 本发明的特性及优点将伴随图式透过以下更详尽的说明变得更清楚明白,其中,相似的参考标号定义全文中相对应的组件。附图中,相同、功能类似及/或结构上类似的组件大体用相同的参考数字表示。该组件首次出现的该附图由参考数字最左边的数字(或数个)所表示。

具体实施方式

[0025] 一、概述

[0026] 本发明的实施例提供一能够进行异步工作分派及其应用的处理单元。在下列的详细说明中,引用的“一个实施例”、“一实施例”或“一具体实施例”等表示所述的实施例可包含一特别特征、结构或特性,但并非每个实施例都必定包含该特别特征、结构或特性。此外,此等用语不一定指相同的实施例。进一步,当结合一实施例以描述一特别特征、结构或特性

时，在熟悉本领域技术者的知识范围内可结合其它已清楚说明或未清楚说明之实施例实现该等特别特征、结构或特性。

[0027] 依据一实施例，一处理单元包括包含在单一着色器核心上的多个虚拟引擎。各虚拟引擎组构成用以接收数据平行处理工作（例如，图形处理工作和一般计算工作）以及在该单一着色器核心上独立地执行这些工作。以此方法，该处理单元可执行二个或更多不同处理工作串流－例如，低延迟处理工作的第一串流以及标准图形处理工作的第二串流－而不需要上下文切换。执行二个或更多不同处理工作串流提供了在没有关联于停止和清除该处理单元的数据的开销 (overhead) 下的上下文切换的低延迟好处。事实上，本发明的实施例使多个上下文能够在单一着色器核心上存在和同时地（实质上地）被执行。

[0028] 仅是为了说明的目的，且并非为了限制，此处叙述的本发明的实施例是以 GPU 而言。然而，相关领域之技术人员将了解本发明之实施例可应用在用以接收处理工作串流的其它类型处理单元，例如中央处理单元和共处理器 (coprocessors)。这些其它类型的处理器被视为在本发明的精神及范围内。

[0029] 在实施例中，该 GPU 处理多个命令缓冲区。举例而言，低延迟处理工作可被安排在第一命令缓冲区，且标准图形处理工作可例如被安排在第二命令缓冲区。该 GPU 的第一虚拟引擎撷取 (retrieve) 该低延迟处理工作，且该 GPU 的第二虚拟引擎撷取该标准图形处理工作。然后将来自各虚拟引擎的工作实质上彼此平行地发出给单一着色器核心。

[0030] 为了使该单一着色器核心能够（实质上地）同时地处理来自二个或更多不同虚拟引擎的工作，该着色器核心的资源在空间及／或时间上被分割。为了达成空间的分割，举例而言，将来自第一虚拟引擎的第一（例如，低延迟）工作发出给该着色器核心的处理组件 (SIMDs) 的第一子集合，且将来自第二虚拟引擎的第二（例如，标准图形）工作发出给该着色器核心的处理组件 (SIMDs) 的第二子集合。为了达到时间的分割，举例而言，第一和第二工作分享着色器核心的处理组件 (SIMDs) 的一部分时间。在一实施例中，该 GPU 包含排程模块用以排程来自该二个或更多不同虚拟引擎的工作以在该着色器核心上执行。

[0031] 依据本发明的实施例分享该 GPU 的资源以提供多个虚拟引擎增进了该 GPU 资源的使用，特别是在大芯片上。可将二个或更多工作的串流发出给单一着色器核心，使该 GPU 有效率地使用计算及输入／输出设备。举例而言，该 GPU 着色器核心的资源（例如，SIMDs）可基于需求、优先级及／或预设的限制，在平行的工作间被分割－同时暂时性地使任何一个工作能够（实质上地）完全地耗用该 GPU 的资源。

[0032] 依据本发明的实施例的范例 GPU 的进一步详细内容将在以下描述。在提供此些详细内容之前，描述在其中此等 GPU 可被实施的范例系统是有帮助的。

[0033] 二、范例系统

[0034] 图 1 为根据一实施例的计算系统 100 的方块图。计算系统 100 包含 CPU 102、GPU 110 以及可选择性地包含共处理器 112。在图 1 的实施例中，CPU 102 及 GPU 110 显示为分开的方块。此仅为了说明的目的，且并非限制。熟悉相关领域的人士将了解 CPU 102 及 GPU 110 可被包含在分开的封装件中或可被合并在单一的封装件或集成电路中。

[0035] 计算系统 100 亦包含可被 CPU 102、GPU 110 以及共处理器 112 访问的系统内存 104。在实施例中，计算系统 100 可包括超级计算机、桌上型计算机、笔记型计算机、游戏机、嵌入式装置、手持装置（例如，手机、智能型手机、MP3 播放器、相机、GPS 装置等等）或其它

包含有或组构成含有 GPU 的装置。

[0036] GPU 110 通过执行某些特殊功能（例如，图形处理工作以及数据平行、一般计算工作）协助 CPU 102，通常快于 CPU 102 在软件内可执行者。GPU 110 包含分享单一着色器的资源的多个虚拟引擎。以此方法，GPU 110 的该多个虚拟引擎可实质上平行地执行多个工作。在实施例中，GPU 110 可被整合至芯片组及 / 或 CPU 102。以下提供 GPU110 额外的详细内容。

[0037] 共处理器 112 亦协助 CPU 102。共处理器 112 可包括，但不限于浮点 (floating point) 共处理器、GPU、网络连结共处理器以及其它类型的共处理器与处理器，这对于熟悉本技术领域的人士是显而易见的。

[0038] GPU 110 与共处理器 112 与 CPU 102 及该系统内存透过总线 114 沟通。总线 114 可为使用在计算机系统内的任何种类的总线，包含周边组件界面 (peripheral component interface，PCI) 总线、加速图形端口 (accelerated graphics port，AGP) 总线、快捷周边组件界面 (PCI Express，PCIE) 总线，或无论是目前可得的或在未来所发展的其它种类的总线。

[0039] 除了系统内存 104 外，计算系统 100 进一步包含区域内存 106 以及区域内存 108。区域内存 106 耦合至 GPU 110 且亦可耦合至总线 114。区域内存 108 耦合至共处理器 112 且亦可耦合至总线 114。为了提供某些数据（例如频繁地被使用的数据）比若该数据储存在系统内存 104 内更快的访问，区域内存 106 与 108 分别地对 GPU 110 与共处理器 112 为可用的。

[0040] 在一实施例中，GPU 110 与共处理器 112 和 CPU 102 同时将指令译码 (decode) 且仅执行该些用于 GPU 110 与共处理器 112 的指令。在另一实施例中，CPU 102 送出用于 GPU 110 与共处理器 112 的指令至个别的命令缓冲区。

[0041] 虽然在图 1 中未特别显示，计算系统 100 亦可包含或耦接至一显示装置（例如，阴极射线管、液晶显示器、电浆显示器等等）。该显示装置用以显示内容给使用者（例如，当计算系统 100 包括计算机、游戏机或手持装置时）。

[0042] 三、范例 GPU

[0043] 如上所述，GPU 110 包括包含实施在着色器核心上的多个虚拟引擎。各虚拟引擎组构成用以执行由 OS 排程器所提供的处理工作串流，其中，一给定的串流中的各处理工作可包含多个单独的处理线程。因为 GPU 110 包含多个虚拟引擎，GPU 110 不需要上下文切换即可执行来自该 OS 排程器的不同处理工作串流。事实上，在实施例中，GPU 110 通过在工作间分享着色器核心的资源而在单一着色器核心内（实质上）同时执行来自多个串流的工作，该多个串流对应于多个不同的上下文。

[0044] 图 2 例示 GPU 110 的范例硬件组件的方块图。请参照图 2，GPU 110 包含命令处理器 230、输入逻辑（包含顶点分析器 (vertex analyzer)208、扫描转换器 212 以及仲裁逻辑 (arbitration logic)222)、着色器核心 214、输出逻辑 224 以及内存系统 210。以下描述各该组件。

[0045] A. 命令处理器

[0046] 命令处理器 230 接收来自由该 OS 排程器所填入的一个或更多命令缓冲区的工作（例如，图形处理工作及一般计算工作）。如图 3 所示，命令处理器 230 包含多个虚拟引擎，

该多个虚拟引擎共享 GPU 110 的资源。该命令处理器 230 的不同虚拟引擎处理不同类型的工作。

[0047] 在图 2 的实施例中，命令处理器 230 包含第一背景引擎 202A、第二背景引擎 202B、实时低延迟引擎 202C、主要 3D 引擎 202D 以及低延迟 3D 引擎 202E。背景引擎 202 处理低优先级的工作。然而，应了解的是命令处理器 230 可包含其它种类的虚拟引擎。背景引擎 202 仅在无其它虚拟引擎使用 GPU 110 的资源时接管该 GPU 110 的资源。为了处理高优先级的工作，实时低延迟引擎 202C 具有对该 GPU 110 资源的优先访问。主要 3D 引擎 202D 处理标准图形处理工作，且低延迟 3D 引擎 202E 处理高优先级的图形处理工作。低延迟 3D 引擎 202E 具有对该 GPU 110 图型处理资源的优先访问。

[0048] 在被发出给 GPU 110 的着色器核心 214 之前，将来自命令处理器 230 的工作提供给输入逻辑。

[0049] B. 输入逻辑

[0050] 输入逻辑仲裁哪个工作要发出给着色器核心 214。在一实施例中，输入逻辑依据该着色器核心 214 资源的可用性及该各种工作的相对优先级实施一软件例行工作以排程该工作在着色器核心 214 上执行。在图 3 的实施例中，输入逻辑包含图形前处理逻辑（其准备用以发布至着色器核心 214 的图形处理工作）以及仲裁逻辑 222（其提供工作至着色器核心 214）。

[0051] 图形前处理逻辑包含顶点分析器 208 以及扫描转换器 212。来自主要 3D 引擎 202D 以及低延迟 3D 引擎 202E 的工作是送至该图形前处理逻辑。先进先出 (first-in, first-out, FIFO) 缓冲区 204A 接收来自主要 3D 引擎 202D 的工作，且 FIFO 缓冲区 204B 接收来自低延迟 3D 引擎 202E 的工作。多任务器 (multiplexer)206 将来自 FIFO 缓冲区 204 中的一者的工作提供给顶点分析器 208。

[0052] 顶点分析器 208 识别关联于图形处理及 / 或一般计算工作的着色器程序，并基于将为可用的输入和输出数据排程何时可在着色器核心 214 中激活各着色器程序。除了排程着色器程序以激活外，顶点分析器 208 亦产生指针给顶点缓冲区且包含连接性数据 (connectivity data)。该指针是用以读取来自顶点缓冲区的顶点。若顶点已被处理且被储存在顶点缓冲区中，顶点分析器 208 可读取来自顶点缓冲区的该顶点，以便一顶点仅会被处理一次。该连接性数据明确说明顶点如何配合在一起以制作基元 (primitive)（例如，三角），以便该基元可被适当地栅格化 (rasterized)。

[0053] 顶点分析器 208 将图形处理工作送给扫描转换器 212，且将一般计算工作送给仲裁逻辑 222。扫描转换器 212 遍历 (traverse) 该基元以判定将被着色器核心 214 处理的画素 (pixels)。扫描转换器 212 再将该画素送给仲裁逻辑 222。仲裁逻辑 222 包含多个多任务器以将来自命令处理器 230 的不同虚拟引擎的工作提供给着色器核心 214。

[0054] C. 着色器核心

[0055] 着色器核心 214 包含用以执行提供给 GPU 110 的工作的多个处理组件 220。处理组件 220 安排为 SIMD 装置，使着色器核心 214 能够（实质上地）同时执行多个数据平行处理工作。为了使着色器核心 214 能够（实质上地）同时处理来自命令处理器 230 的多个虚拟引擎的工作，处理组件 220 在空间及 / 或时间上被分割。

[0056] 为了达成空间的分割，处理组件 220 的不同子集合组构成用以执行不同的工作。

举例而言，来自第一虚拟引擎（例如，实时低延迟引擎202C）的第一（例如，低延迟）工作可被发出给着色器核心214的处理组件220的第一子集合，且来自第二虚拟引擎（例如，主要3D引擎202D）的第二（例如，标准图形）工作可被发出给着色器核心214的处理组件220的第二子集合。处理组件220的各子集合再独立地执行其接收到的工作。

[0057] 为了达成时间上的分割，不同虚拟引擎的不同处理工作分享着色器核心214的处理组件220的一部分时间。由上述的范例中，该第一及第二工作分享着色器214的处理组件220的一部分时间。

[0058] 着色器214亦包含一个或更多区域数据共享(local data shares，LDS)228，该区域数据共享228用以储存由处理组件220所使用以执行由该OS排程器所提供的处理工作的数据。LDS 228储存关联于将由着色器核心214所执行的各工作的状态数据。在一实施例中，LDS 228储存多个不同上下文的状态数据，使着色器核心214能够（实质上地）同时执行来自OS排程器且关联于多个不同上下文的多个工作而不需要上下文切换。

[0059] 处理组件220的中间结果可在着色器核心214中重新处理。举例而言，处理组件220可实现多个不同的着色器程序（例如，几何着色器、顶点着色器、画素着色器、曲面细分(tessellation)着色器等等）以完成由该OS排程器所提供的单一图形处理工作。将该不同着色器程序的中间结果送回顶点分析器208及/或扫描转换器212，且最后再回到处理组件220。在处理组件220完成由该OS排程器所提供的工作之后，将该最终结果提供给输出逻辑224。

[0060] D. 输出逻辑

[0061] 输出逻辑224包含多个缓冲区，包括写入组合高速缓存、深度缓冲区以及颜色缓冲区。该写入组合高速缓存结合数据以被写入片外(off-chip)内存中，使片外内存能够有效率地访问。该深度缓冲区缓冲结果以用于z测试(z-testing)。该颜色缓冲区缓冲结果以用于颜色混合(color blending)。在执行关联于该写入组合高速缓存、深度缓冲区以及颜色缓冲区的程序之后，输出逻辑224将该结果提供给内存系统210。

[0062] E. 内存系统

[0063] 内存系统210包含一个或更多片上(on-chip)高速缓存以及一个或更多片外内存接口。内存系统210耦接至命令处理器230、顶点分析器208、扫描转换器212、着色器核心214以及输出逻辑各者。当此些组件之任何一者需要数据以执行着色器程序时，会对内存系统210的该片上高速缓存发出要求。若在该片上高速缓存中为命中(hit)（亦即，要求的数据在该片上高速缓存中），发送该数据给提出要求的该组件。若在该片上高速缓存中为未命中(miss)（亦即，要求的数据不在该片上高速缓存中），所需要的数据必须从片外内存（例如，图1的系统内存104）透过该内存系统210的片外内存接口撷取。在该数据由片外内存重新撷取之后，发送该数据给提出要求的该组件。此外，该数据亦使用本领域之技术人员所熟知的高速缓存内存技术储存在该片上高速缓存(on-chip cache)中。

[0064] 四、范例操作

[0065] GPU 110组构成用以执行由一OS排程器所提供的多个处理工作串流。该多个处理工作串流可由单一应用程序或多于一个的应用程序产生。

[0066] 图3A显示了着色器核心214（实质上地）同时执行二个（或更多）不同处理工作串流的范例，其中，该处理工作串流由单一应用程序302所产生。应用程序302可为，举例而

言，用以在GPU上执行的产生图形处理工作的终端使用者应用程序（例如，视讯游戏应用程序、计算机辅助设计（computer-aided design，CAD）应用程序等等）或产生一般计算工作的终端使用者应用程序（例如，数学算法、物理仿真等等）。请参照图3A，应用程序302产生第一工作308A以及第二工作308B。应用程序302所产生的各工作308包含优先级类型(priority type)。举例而言，应用程序302可指出该第一工作308A为低延迟、高优先级工作，且该第二工作308B为标准优先级工作。OS排程器310接收该应用程序302所产生的工作且将该工作发出给GPU 110的不同虚拟引擎。举例而言，OS排程器310将第一工作308A发出给第一虚拟引擎312A，且将第二工作308B发出给第二虚拟引擎312B。之后，来自各虚拟引擎312的工作由GPU 110的着色器核心214（实质上地）同时执行。

[0067] 图3B显示了着色器核心214（实质上地）同时执行二个（或更多）不同处理工作串流的范例，其中，该处理工作串流由不同应用程序所产生。如图3B所示，第一处理工作330A由第一应用程序302A所产生，且第二处理工作330B由第二应用程序302B所产生。各工作330包含优先级类型。OS排程器310接收该工作330且将其发出给GPU 110的不同虚拟引擎。举例而言，OS排程器310将第一工作330A发出给第一虚拟引擎332A，且将第二工作330B发出给第二虚拟引擎332B。之后，来自各虚拟引擎332的工作由GPU 110的着色器核心214（实质上地）同时执行。

[0068] 在图3A及3B的范例中，GPU 110接收该工作及该优先级类型两者。为了提供优先级类型给GPU 110，应用程序302提供位(bits)给API以指出各该工作的优先级类型。该API依序提供此信息给GPU 110的驱动程序。在一实施例中，GPU 110包含排程模块，其至少一部份地依据该应用程序所明确指定的优先级类型排程将在着色器核心214上执行的该工作。

[0069] 虽然图3A及3B的范例显示了着色器核心214仅执行二个处理工作串流，但此仅为了说明的目的，且并非限制。应该了解到着色器核心214可（实质上地）同时执行由一个或更多应用程序所产生的二个（或更多）不同处理工作串流。

[0070] 举例而言，图4为范例工作流程，显示了运行在计算系统（例如，计算系统100）与包含在该计算系统内的GPU 110上的一个或更多应用程序之间的软件与硬件的各种层。在图4的实施例中，有二种不同的处理单元类型，CPU 102以及GPU 110。此外，二个应用程序正运行在该计算系统上－第一应用程序402A以及第二应用程序402B。CPU102提供了应用程序402所要求的主要功能性。举例而言，CPU 102可包含多个核心412A-N，其中，第一应用程序402A主要地运行在第一核心412A上，且第二应用程序402B主要地运行在第二核心412N上。

[0071] 在运行的过程中，应用程序402可产生多个工作404A-D以在GPU110而非CPU 102上执行。工作404可包含数据平行处理工作（例如，图形处理工作、一般计算工作等等），GPU 110可能可以在软件中以快于CPU 102的方式执行该数据平行处理工作。各工作404包含由该应用程序402所明确指定的该优先级类型（如同图3A及3B中所显示的工作中所包含的优先级类型）。在习知系统中，OS排程器将以连续的方式将工作404提供给单一命令缓冲区，且习知的GPU将连续地处理该工作。不同于此等习知系统，OS排程器310依据该应用程序402所明确指定的优先级类型将各工作404提供给多个工作缓冲区420A-N中之一者。举例而言，OS排程器310提供第一类工作（例如，高优先级工作）给第一命令缓

冲区 420A，第二类工作（例如，图形处理工作）给第二命令缓冲区 420B 等等。

[0072] GPU 110 包含多个虚拟引擎 432A-N，其各者组构成用以服务命令缓冲区 420A-N 中的一者。举例而言，第一虚拟引擎 432A 组构成用以服务第一命令缓冲区 420A；第二虚拟引擎 432B 组构成用以服务第二命令缓冲区 420B；以及第 N 虚拟引擎 432N 组构成用以服务第 N 命令缓冲区 420N。如上所述，来自虚拟引擎 432 的工作再（实质上地）同时由着色器核心 214 执行。在一实施例中，GPU 110 的排程模块 434 基于至少下列情况将针对由着色器核心 214 所执行的工作进行排程：(i) 由应用程序 402 所明确说明的优先级类型；(ii) 由虚拟引擎 432 所处理的工作 404 之间的相对优先级；以及 (iii) 着色器核心 214 内资源的可用性。举例而言，排程模块 434 可基于需求、优先级及 / 或预设的限制在平行的工作 404 间分割该着色器核心 214 的资源 - 同时暂时性地使工作 404 中的任何一者能够完全地耗用 GPU 110 的资源。通过在着色器 214 中执行二个或更多不同的处理工作串流，GPU 110 提供了在没有关联于储存、交换和清除关于上下文切换的数据的开销 (overhead) 下的上下文切换的低延迟好处。

[0073] 五、范例计算机实施

[0074] 本发明的实施例可利用硬件、软件或其组合据以实施，且可在一个或更多计算机系统或其它处理系统中被实施。一计算机系统 500 的范例显示在图 5。

[0075] 计算机系统 500 包含一个或更多处理器，例如处理器 504。处理器 504 可为通用处理器（例如，CPU 102）或专用处理器（例如，GPU 110）。处理器 504 连接至通讯基础设施 (communication infrastructure)506（例如，通信总线、交叉汇流条 (corss-over bar) 或网络）。各种软件的实施例就此范例计算机系统来描述。在阅读本说明后，如何使用其它计算机系统及 / 或架构以实施本发明对于熟习本领域者而言将变得清楚明白。

[0076] 计算机系统 500 包含显示接口 502 以发送图形、文字以及其它来自通讯基础设施 506（或来自未图标的框架缓冲器 (frame buffer)）的数据以显示在显示单元 530 上。

[0077] 计算机系统 500 亦包含主内存 508，较佳地是随机访问内存 (random access memory，RAM)，且复可包含辅助内存 510。举例而言，该辅助内存 510 可包含硬盘机 512 及 / 或可移式储存装置 514，以软盘机、磁带机、光驱等等为代表。该可移式储存装置 514 以众所周知的方法读取及 / 或写入可移式储存单元 518。可移式储存单元 518 表示为软盘、磁带、光盘等等，其由可移式储存装置 514 读取及写入。如将被了解的，该可移式储存单元 518 包含计算机可使用储存媒体，具有储存在其中的计算机软件及 / 或数据。

[0078] 在替代的实施例中，辅助内存 510 可包含其它相似的装置以令计算机程序或其它指令可被加载至计算机系统 500 中。此等装置可包含，例如，可移式储存单元 522 以及接口 520。此范例可包含程序储存匣 (program cartridge) 及储存匣接口（例如在视讯游戏装置中所发现者）、可移式记忆芯片（例如，可擦除可编程只读存储器 (erasable programmable read only memory，EPROM) 或可编程只读存储器 (programmable read only memory，PROM)）与伴随的插座以及其它可移式储存单元 522 以及接口 520，其令软件及数据由可移式储存单元 522 传输至计算机系统 500。

[0079] 计算机系统 500 亦可包含通信接口 524。通信接口 524 令软件及数据在计算机系统 500 与外部装置间传输。通信接口 524 的范例可包含调制解调器、网络接口（例如，以太网络卡）、通讯端口、个人计算机记忆卡国际协会 (Personal Computer Memory Card

International Association,PCMCIA) 插槽及卡等等。透过通信接口 524 传输的软件及数据为信号 528 的形式,其可为电子、电磁、光学或其它能够被通信接口 524 所接收的讯号。该讯号 528 透过通信路径 (例如,频道)526 提供给通信接口 524。该频道 526 携带讯号 528 且可使用电线 (wire) 或电缆 (cable)、光纤、电话线、蜂巢式链 (cellular link)、无线电频率 (radio frequency,RF) 链以及其它通信频道。

[0080] 在本文件中,该用语“计算机可读取储存媒体”是用来一般指例如可移式储存装置 514 以及安装在硬盘机 512 中的硬盘的媒体。这些计算机程序产品提供软件给计算机系统 500。

[0081] 计算机程序 (亦可称为计算机控制逻辑) 是储存在主内存 508 及 / 或辅助内存 510 中。计算机程序亦可透过通信接口 524 以接收。如本文中所讨论者,当此等计算机程序被执行时,其使该计算机系统 500 能够执行本发明的特征。特别是,当此等计算机程序被执行时,其使该处理器 504 能够执行本发明的特征。因此,此等计算机程序代表该计算机系统 500 的控制器。

[0082] 在一实施例中,该软件可储存在计算机程序产品且使用可移式储存装置 514、硬盘机 512 或通信接口 524 加载至计算机系统 500 中。当该控制逻辑 (软件) 由该处理器 504 所执行时,致使该处理器 504 执行如本文所述的本发明的实施例的功能。

[0083] 六、范例软件实施

[0084] 除了处理单元 (例如,CPU 102 及 / 或 GPU 110) 的硬件实施之外,此等处理单元亦可被实施在配置的软件中,例如,在组构成用以储存该软件 (例如,计算机可读取程序编码) 的计算机可读取媒体中。该计算机编码导致本发明的实施例的可能 (enablement),包含下列实施例 :(i) 本文所叙述的该系统的功能及技术 (例如,提供工作给 GPU 110、排程在 GPU 110 中的工作、执行在 GPU 110 中的工作等等);(ii) 本文所叙述的该系统的制造及技术 (例如,GPU 110 的制造);或 (iii) 本文所叙述的该系统的功能与制造的组合及技术。

[0085] 举例而言,该软件可透过一般程序语言 (例如,C 或 C++)、包含 Verilog HDL、VHDL、Altera HDL(AHDL) 等等的硬件描述语言 (hardware-description languages,HDL) 或其它可用的编程及 / 或简图撷取 (schematic-capture) 工具 (例如,电路撷取 (circuit-capture) 工具) 加以完成。该程序编码可被配置在任何已知的计算机可读取媒体中,包含半导体、磁盘或光盘 (例如,CD-ROM、DVD-ROM)。同样地,该编码可透过包含网际网络 (the Internet) 和互联网 (internets) 的通信网路传输。应该了解到由该系统所完成的功能及 / 或所提供的结构以及上述的技术可被表现在核心 (例如,GPU 核心) 中,该核心以程序编码实施并且可被转换为作为集成电路产品的一部份的硬件。

[0086] 七、结论

[0087] 应了解,系意图以具体实施方式部分而非发明内容与摘要部分解释权利要求。该发明内容与摘要部分可阐明一个或更多但并非所有发明人所思及的本发明的范例实施例,且因此,并非意欲以任何方式限制本发明以及所附加的权利要求。

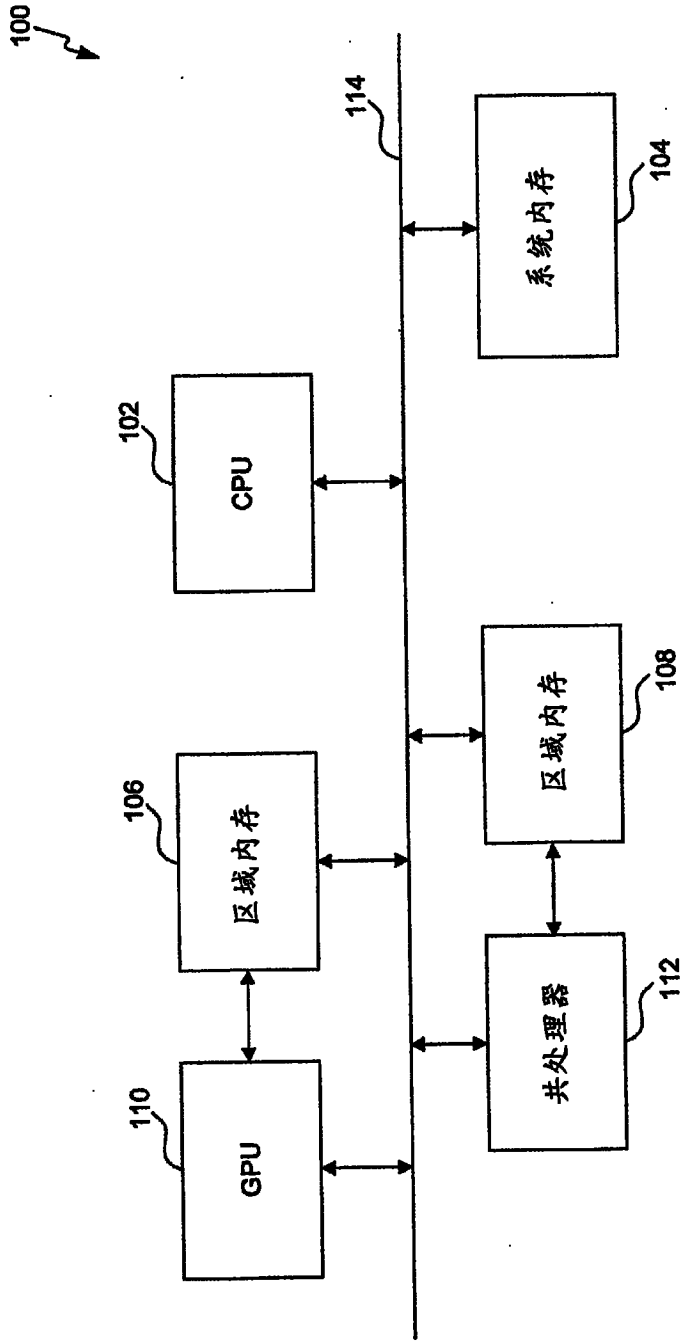


图 1

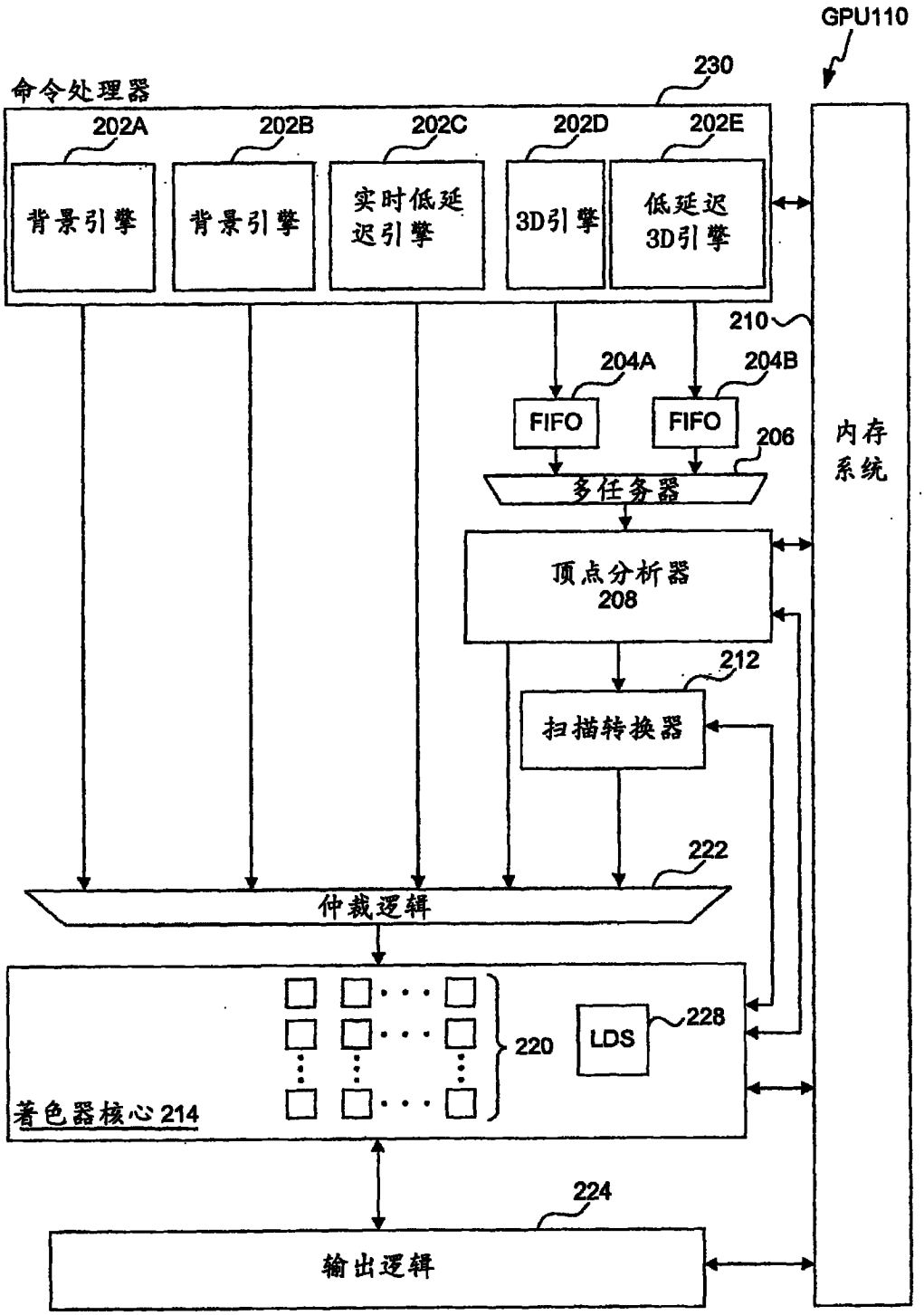

GPU110
命令处理器
230
202A
202B
202C
202D
202E
背景引擎
背景引擎
实时低延迟引擎
3D引擎
低延迟3D引擎
210
204A
204B
FIFO
FIFO
206
多任务器
内存系统
顶点分析器
208
212
扫描转换器
222
仲裁逻辑
220
LDS
228
着色器核心 214
224
输出逻辑

图 2

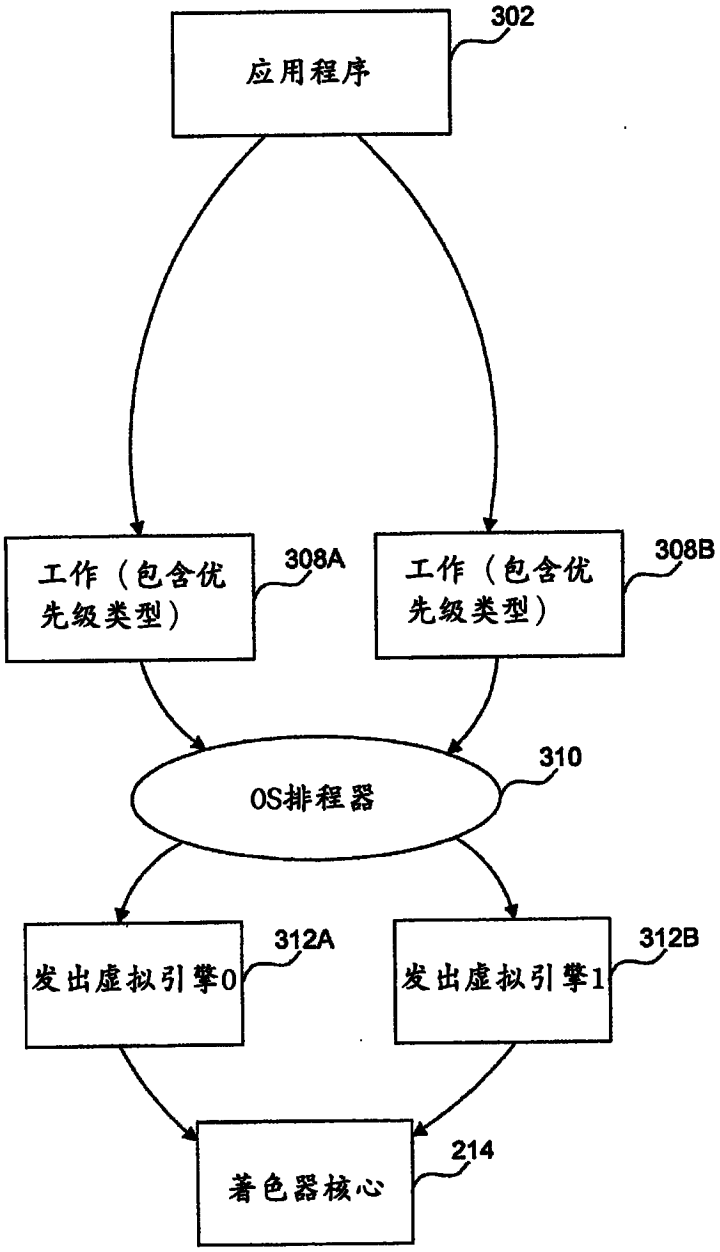

应用程序
302
工作（包含优先级类型）
308A
工作（包含优先级类型）
308B
OS排程器
310
发出虚拟引擎0
312A
发出虚拟引擎1
312B
著色器核心
214

图 3A

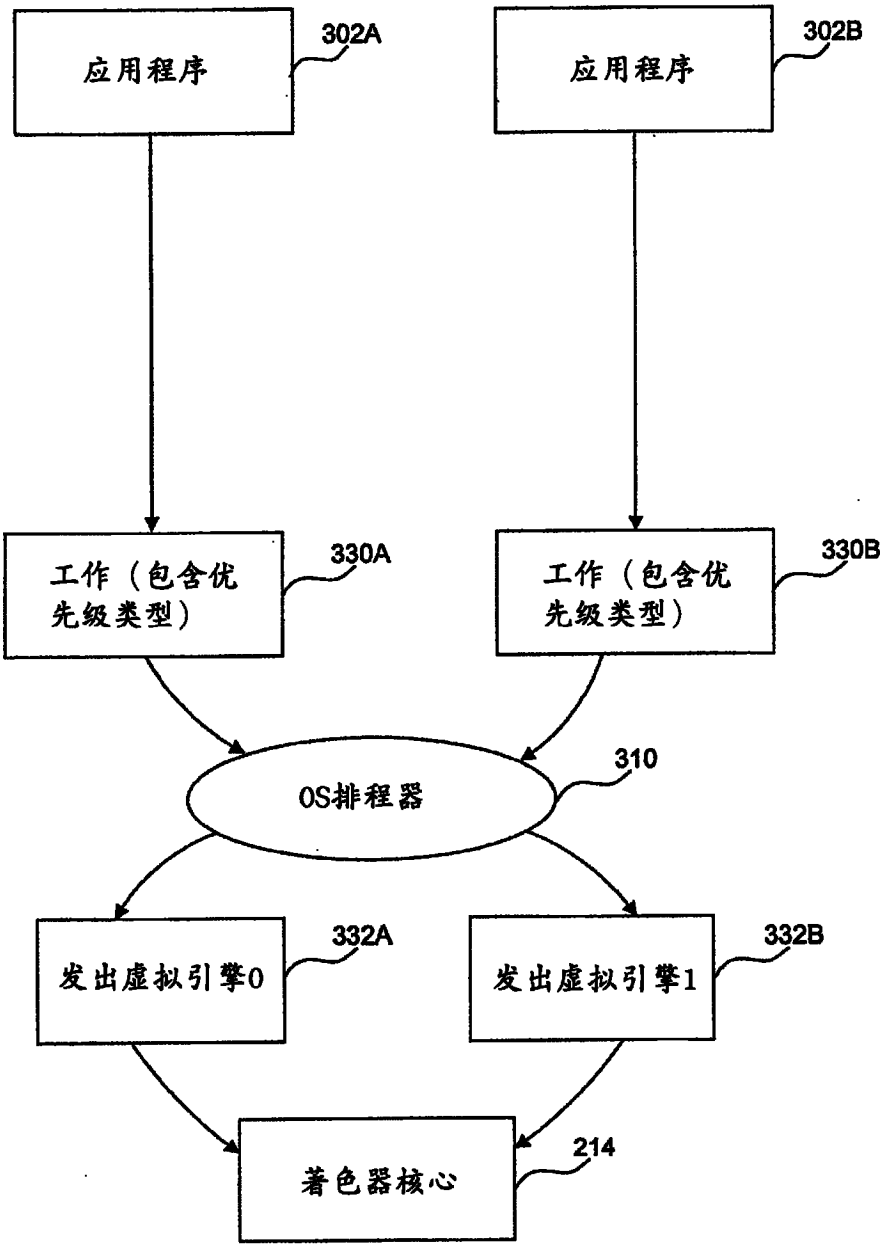

应用程序
302A
应用程序
302B
工作（包含优先级类型）
330A
工作（包含优先级类型）
330B
OS排程器
310
发出虚拟引擎0
332A
发出虚拟引擎1
332B
着色器核心
214

图 3B

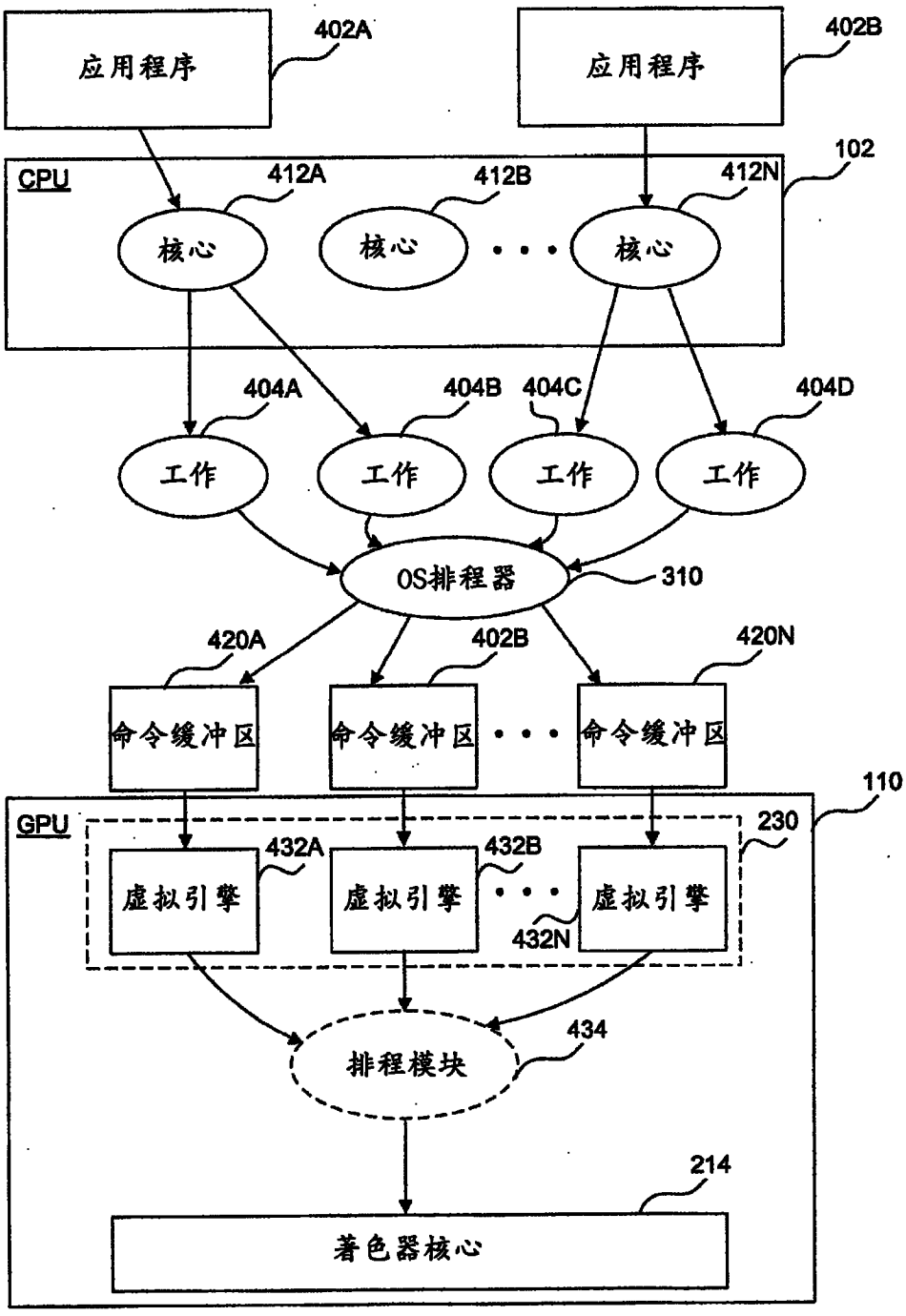

应用程序
402A
应用程序
402B
CPU
102
412A
412B
412N
核心
核心
核心
404A
404B
404C
404D
工作
工作
工作
工作
OS排程器
310
420A
402B
420N
命令缓冲区
命令缓冲区
命令缓冲区
110
GPU
230
432A
432B
虚拟引擎
虚拟引擎
虚拟引擎
432N
排程模块
434
214
著色器核心

图 4

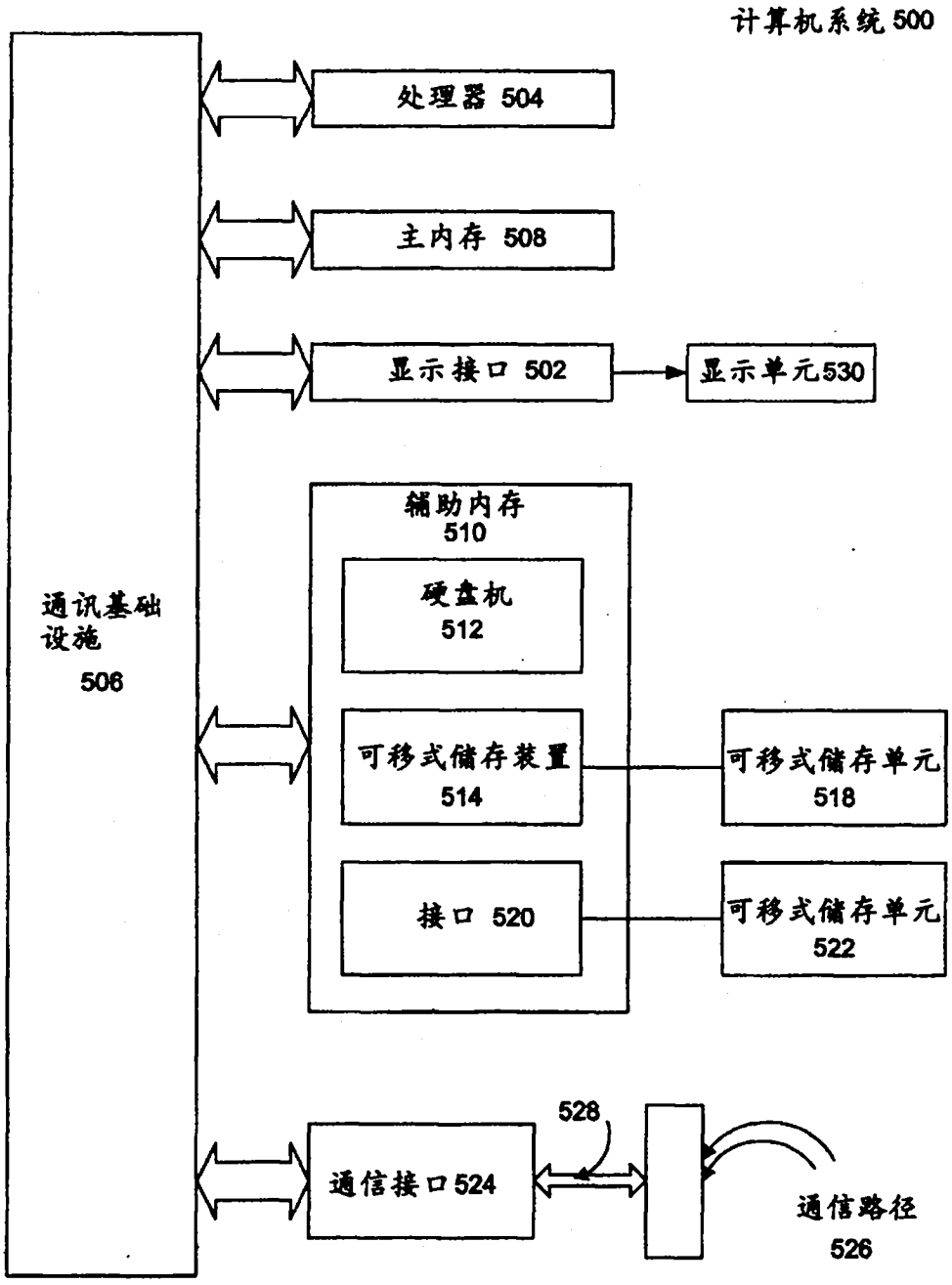

计算机系统 500
处理器 504
主内存 508
显示接口 502
显示单元530
辅助内存
510
硬盘机
512
通讯基础
设施
506
可移式储存装置
514
可移式储存单元
518
接口 520
可移式储存单元
522
528
通信接口524
通信路径
526

图 5