(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 103608776 B
(45)授权公告日 2017.09.05

(21)申请号 201180060199.1

(22)申请日 2011.12.09

(65)同一申请的已公布的文献号
申请公布号 CN 103608776 A

(43)申请公布日 2014.02.26

(30)优先权数据
61/423,465 2010.12.15 US
13/287,418 2011.11.02 US

(85)PCT国际申请进入国家阶段日
2013.06.14

(86)PCT国际申请的申请数据
PCT/US2011/064172 2011.12.09

(87)PCT国际申请的公布数据
WO2012/082557 EN 2012.06.21

(73)专利权人 超威半导体公司
地址 美国加利福尼亚州

(72)发明人 本杰明·托马斯·桑德
迈克尔·休斯顿 牛顿·张
基思·洛韦里

(74)专利代理机构 上海胜康律师事务所 31263
代理人 李献忠

(51)Int.Cl.
G06F 9/46(2006.01)

(56)对比文件
US 2004054999 A1,2004.03.18,全文.

审查员 宋秀之

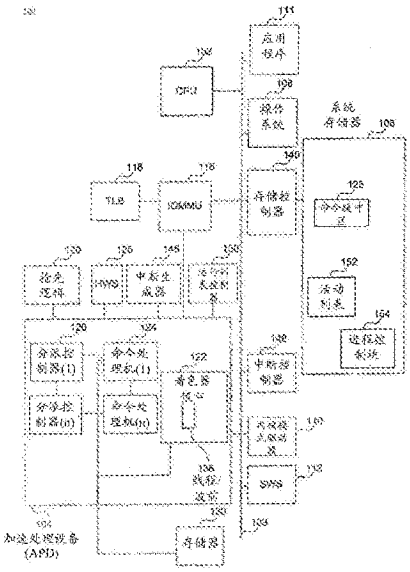
权利要求书3页 说明书13页 附图8页

(54)发明名称

异构型处理设备上的动态工作划分

(57)摘要

一种用于平衡异构型处理设备上的工作负荷的方法、系统以及制品。所述方法包括:由与不同类型的处理器相关的离队实体存取一种类型的处理器的存储器;从在所述存储器内的多个任务中识别可以由所述不同类型的所述处理器处理的任务;对能够存取所述存储器的多个离队实体进行同步;以及使所述任务从所述存储存储器中离队。



CN 103608776 B

1.一种用于平衡异构型处理设备上的工作负荷的方法,其包括:

将多个任务存储在共享队列中,所述共享队列与一个或多个第一类型的第一处理器和一个或多个第二类型的第二处理器通信连接,其中所述多个任务中的每个包括指向用于在所述一个或多个第一处理器上执行的微代码的第一指针和指向用于在所述一个或多个第二处理器上执行的所编译的代码的第二指针,

其中所述一个或多个第一处理器中的每个包括:

第一离队模块,以及

一个或多个第一处理核心;

其中所述一个或多个第二处理器中的每个包括:

同步模块,

第二离队模块,以及

一个或多个第二处理核心;

通过所述一个或多个第二处理器的至少一个同步模块,利用用于同步的原子操作同步所述一个或多个第一处理器中的每个的所述第一离队模块和所述一个或多个第二处理器中的每个的所述第二离队模块对所述共享队列的存取;

通过所述一个或多个第二处理器的至少一个第二离队模块从所述多个任务中读取特定的任务;

在所述特定的任务适于在所述一个或多个第二处理器上执行的条件下通过所述一个或多个第二处理器的至少一个第二离队模块使所述特定的任务从所述共享队列离队;

通过所述一个或多个第二处理器的至少一个第二离队模块指示所述一个或多个第二处理核心执行由所述第二指针指向的所述被编译的代码。

2.如权利要求1所述的方法,其中所述特定的任务被确定为基于存储在所述共享队列中的任务参数适于在所述一个或多个第二处理器上执行。

3.如权利要求1所述的方法,其中所述一个或多个第二处理器包括中央处理器(CPU)并且所述一个或多个第一处理器包括加速处理设备(APD)。

4.如权利要求3所述的方法,其中所述一个或多个第二处理器的所述第一离队模块是硬件设备。

5.如权利要求3所述的方法,其中所述一个或多个第一处理器的所述第二离队模块是软件模块。

6.如权利要求1所述的方法,其中所述一个或多个第二处理器的所述第二离队模块使之离队的任务的数量基于所述一个或多个第二处理器的类型。

7.如权利要求1所述的方法,还包括:

通过所述一个或多个第一处理器的至少一个第一离队模块从所述多个任务中读取另一个特定的任务;

在所述另一个特定的任务适于在所述一个或多个第一处理器上执行的条件下通过所述至少一个第一离队模块使所述另一个特定的任务从所述共享队列离队;

通过所述至少一个第一离队模块指示所述一个或多个第一处理核心执行由所述第一指针指向的所述微代码。

8.如权利要求1所述的方法,其中所述多个任务中的每一个还包括指向中间语言表示

的第三指针，所述中间语言表示包括在预先确定的事件发生之后变为可执行的依赖关系信息。

9.一种用于平衡异构型处理设备上的工作负荷的系统，其包括：

一个或多个第一类型的第一处理器；

一个或多个第二类型的第二处理器；以及

共享队列，其与所述一个或多个第一处理器和所述一个或多个第二处理器通信连接；

其中所述一个或多个第一处理器中的每个包括：

第一离队模块，以及

一个或多个第一处理核心；

其中所述一个或多个第二处理器中的每个包括：

同步模块，

第二离队模块，以及

一个或多个第二处理核心；

其中所述共享队列存储多个任务，其中所述多个任务中的每个包括指向用于在所述一个或多个第一处理器上执行的微代码的第一指针和指向用于在所述一个或多个第二处理器上执行的所编译的代码的第二指针，

其中所述一个或多个第二处理器的每个同步模块在从所述共享队列中移除特定的任务之前利用用于同步的原子操作同步所述一个或多个第一处理器中的每个的所述第一离队模块和所述一个或多个第二处理器中的每个的所述第二离队模块对所述共享队列的存取，以及

其中所述一个或多个第二处理器的每个第二离队模块：

从所述多个任务中读取所述特定的任务，

在所述特定的任务适于在所述一个或多个第二处理器上执行的条件下使所述特定的任务从所述共享队列离队，

指示所述一个或多个第二处理核心执行由所述第二指针指向的所述被编译的代码。

10.如权利要求9所述的系统，其中所述特定的任务被确定为基于存储在所述共享队列中的任务参数适于在所述一个或多个第二处理器上执行。

11.如权利要求9所述的系统，其中所述一个或多个第二处理器包括中央处理器(CPU))并且所述一个或多个第一处理器加速处理设备(APD)。

12.如权利要求9所述的系统，其中)所述一个或多个第一处理器的所述第二离队模块是硬件设备。

13.如权利要求9所述的系统，其中所述一个或多个第一处理器的每个第一离队模块：

从所述多个任务中读取另一个特定的任务；

在所述另一个特定的任务适于在所述一个或多个第一处理器上执行的条件下使所述另一个特定的任务从所述共享队列离队；

指示所述一个或多个第一处理核心执行由所述第一指针指向的所述微代码。

14.如权利要求9所述的系统，其中所述多个任务中的每一个还包括指向中间语言表示的第三指针，所述中间语言表示包括在预先确定的事件发生之后变为可执行的依赖关系信息。

15.如权利要求9所述的系统，其中所述至少一个第二离队模块使之离队的任务的数量基于所述一个或多个第二处理器的类型。

16.一种用于平衡异构型处理设备上的工作负荷的系统，其包括：

一个或多个第一类型的第一处理器；

一个或多个第二类型的第二处理器；以及

共享队列，其与所述一个或多个第一处理器和所述一个或多个第二处理器通信连接；

其中所述一个或多个第一处理器中的每个包括：

第一离队模块，以及

一个或多个第一处理核心；

其中所述一个或多个第二处理器中的每个包括：

同步模块，

第二离队模块，以及

一个或多个第二处理核心；

其中所述共享队列存储多个任务，其中所述多个任务中的每个包括指向用于在所述一个或多个第一处理器上执行的微代码的第一指针和指向用于在所述一个或多个第二处理器上执行的所编译的代码的第二指针，

其中所述一个或多个第二处理器的每个同步模块在从所述共享队列中移除特定的任务之前利用用于同步的原子操作同步所述一个或多个第一处理器中的每个的所述第一离队模块和所述一个或多个第二处理器中的每个的所述第二离队模块对所述共享队列的存取，以及

其中所述一个或多个第一处理器的每个第一离队模块：

从所述多个任务中读取所述特定的任务，

在所述特定的任务适于在所述一个或多个第一处理器上执行的条件下使所述特定的任务从所述共享队列离队，

指示所述一个或多个第一处理核心执行由所述第一指针指向的所述微代码。

17.如权利要求16所述的系统，其中所述多个任务中的每一个还包括指向中间语言表示的第三指针，所述中间语言表示包括在预先确定的事件发生之后变为可执行的依赖关系信息。

异构型处理设备上的动态工作划分

技术领域

[0001] 本发明大体上是针对计算机系统。更具体来说，本发明是针对一种用于统一计算机系统内部的计算部件的体系结构。

背景技术

[0002] 对使用图形处理单元(GPU)来进行一般计算的渴望在最近由于GPU的示例性每单位功率性能和/或成本而变得更加显著。一般来说，GPU的计算能力已以超过对应中央处理器(CPU)平台的计算能力的速率增长。随着移动计算市场(例如，笔记本计算机、移动智能电话、平板计算机等)和其所必需的支持服务器/企业系统的蓬勃发展，这种增长已被用来提供指定品质的所需用户体验。因此，组合使用CPU和GPU来执行具有数据并行内容的工作负荷正在成为一项体积技术(volume technology)。

[0003] 然而，GPU传统上已在约束程序设计环境中进行操作，其可主要用于图形的加速。这些约束由以下事实而引起：GPU并不具有与CPU一样丰富的程序设计生态系统。因此，它们的使用已主要限于2D和3D图形以及少数前沿的多媒体应用，这些多媒体应用已被习惯地用于处理图形和视频应用程序设计接口(API)。

[0004] 随着多厂商支持的OpenCL™和DirectCompute®、标准API和支持工具的出现，GPU在传统应用中的限制已被扩展到传统图形的范围之外。虽然OpenCL和DirectCompute是有希望的开端，但是在创建允许将CPU和GPU组合来像CPU一样流畅地用于大多数程序设计任务的环境和生态系统方面仍存在着许多障碍。

[0005] 现有的计算系统常常包括多个处理设备。例如，一些计算系统包括在独立芯片上的CPU和GPU(例如，CPU可能位于母板上，而GPU可能位于图形卡上)或在单个芯片封装中的CPU和GPU。然而，这两种布置仍包括与以下各项相关的重大挑战：(i)独立的存储系统、(ii)有效调度、(iii)提供进程之间的服务质量(QoS)保证、(iv)程序设计模型以及(v)编译至多个目标指令集体系结构(ISA)—全部都要同时使功耗降到最小。

[0006] 例如，离散的芯片布置迫使系统和软件体系结构设计者利用芯片间接口来使每一个处理器存取存储器。虽然这些外部接口(例如，芯片间接口)对用于配合异构型处理器的存储器等待时间和功耗具有负效应，但是独立的存储系统(即，独立的地址空间)和驱动器管理的共享存储器产生开销，所述开销对细粒卸荷(fine grain offload)来说变得不可接受。

[0007] 尽管CPU和GPU传统上执行不同任务，但许多类型的工作负荷可以使用CPU或GPU来执行。当CPU或者GPU空闲时，如果工作负荷可以在处理器之间重新分布，那么计算环境会受益。

[0008] 在处理之前，工作负荷被分为许多个离散任务。每个任务被指派给与CPU或者GPU相关的工作队列。一旦任务被指派给CPU或GPU来处理，包括CPU和GPU的常规计算环境就不允许工作重新分布给不同类型的处理设备。常规系统允许CPU将任务重新分布给其它CPU，而GPU不具有重新分布工作的功能性。这一点还妨碍了处理，因为CPU可能在GPU空闲时为繁

忙的，反之亦然。这些失衡的处理导致了低效率和次最优性能，特别是当任务可以在任一个处理设备上进行处理时。

发明内容

[0009] 因此，所需要的是其中CPU和GPU能够在它们自己之间重新分布并且平衡任务的系统和方法。

[0010] 虽然GPU、加速处理单元(APU)以及通用用途的图形处理单元(GPGPU)是这个领域中常用的术语，但是表述“加速处理设备(APD)”被认为是更广义的表述。例如，APD是指硬件和/或软件的任何配合集合，与常规CPU、常规GPU、软件和/或其组合相比，所述任何配合集合以加速方式完成与加速图形处理任务、数据并行任务或嵌套数据并行任务相关的那些功能和计算。

[0011] 在一些情况下，本发明的实施方案包括一种用于平衡异构型处理设备上的工作负荷的方法、系统以及制品。所述方法包括：由与不同类型的处理器相关的离队实体存取一种类型的处理器的存储器；从在存储器内的多个任务中识别可以由所述不同类型的处理器处理的任务；对能够存取所述存储器的多个离队实体进行同步；以及使这个任务从所述存储器中离队。

[0012] 以下参照附图详细地描述本发明的额外特征和优点，连同本发明的各种实施方案的结构和操作。应该指出，本发明不限于本文所描述的具体实施方案。本文所提出的这类实施方案仅用于说明性目的。基于本文所包括的教义，一个或多个相关领域的技术人员将会明白额外的实施方案。

附图说明

[0013] 并入本文并且形成本说明书的一部分的附图示出本发明，并且与描述一起，进一步用于解释本发明的原理并且用于使相关领域技术人员能够制作并使用本发明。以下参照附图描述本发明的各种实施方案，在所述附图中，相似参考数字自始至终用于指示相似元件。

[0014] 图1A是根据本发明的实施方案的处理系统的说明性方框图。

[0015] 图1B是图1A中所示的APD的说明性方框图图解。

[0016] 图2是一种排队系统的说明性方框图，在所述排队系统中，CPU和APD被熔铸在同一硅片上。

[0017] 图3是一种处于离散系统环境中的排队系统的说明性方框图。

[0018] 图4是为多个CPU和APD平衡任务的多个队列的说明性方框图。

[0019] 图5是APD使任务从存储任务的队列中离队以用于在融合环境中的CPU上进行处理的说明性流程图。

[0020] 图6是CPU使任务从存储任务的队列中离队以用于在APD上进行处理的说明性流程图。

[0021] 图7是CPU使任务从存储任务的队列中离队以用于在离散环境中的CPU上进行处理的说明性流程图。

[0022] 将参看附图描述本发明。一般来说，元件第一次出现所在的附图通常由对应元件

符号的最左边的数字指示。

具体实施方式

[0023] 在以下详述中，对“一个实施方案”、“实施方案”、“示例实施方案”等的参考指示所描述的实施方案可以包括具体特征、结构或特点，但是每个实施方案可能并没有必要包括所述具体特征、结构或特点。此外，这类短语没有必要是指同一实施方案。另外，当结合一个实施方案描述具体特征、结构或特点时，所主张的是本领域技术人员知道结合无论是否被明确地描述的其它实施方案来实现这种特征、结构或特点。

[0024] 术语“本发明的实施方案”不要求本发明的所有实施方案都包括所论述的特征、优点或操作模式。在不脱离本发明的范围的情况下，可以设计出替代实施方案，并且可能并未详细描述或可能省略本发明的众所周知的元件，以免模糊本发明的相关细节。另外，本文所使用的术语仅出于描述具体实施方案的目的，而并不意图限制本发明。例如，如本文所使用，单数形式的“一个(种)”以及“所述”也意图包括复数形式，除非上下文另有明确指示。还将进一步理解的是，术语“包括”、“包括了”、“包含”和/或“包含了”在本文中使用时指明所陈述的特征、整体、步骤、操作、元件和/或部件的存在，但是并不排除一个或多个其它特征、整体、步骤、操作、元件、部件和/或其群组的存在或添加。

[0025] 图1A是包括两个处理器(即，CPU 102和APD 104)的统一计算系统100的示例性图解。CPU 102可以包括一个或多个单核或多核CPU。在本发明的一个实施方案中，系统100被形成在单个硅芯片或封装上，组合CPU 102和APD 104以提供统一的程序设计和执行环境。这个环境使得APD 104能够像CPU 102一样流畅地用于一些程序设计任务。然而，CPU 102和APD 104被形成在单个硅芯片上并不是本发明的绝对要求。在一些实施方案中，CPU和APD有可能被单独地形成并且被安装在相同或不同的衬底上。

[0026] 在一个实施例中，系统100还包括存储器106、操作系统108以及通信基础设施109。以下更详细地论述操作系统108和通信基础设施109。

[0027] 系统100还包括内核模式驱动器(KMD)110、软件调度器(SWS)112，以及存储器管理单元116，如输入/输出存储器管理单元(IOMMU)。系统100的部件可以被实施为硬件、固件、软件或其任何组合。本领域技术人员将会理解，除了图1A中所示的实施方案中所示的各项之外，系统100可以包括一个或多个软件、硬件以及固件部件，或与图1A中所示的实施方案中所示的各项不同的一个或多个软件、硬件以及固件部件。

[0028] 在一个实施例中，驱动器(如KMD 110)典型地通过计算机总线或通信子系统来与设备进行通信，硬件连接至所述计算机总线或通信子系统。当调用程序调用驱动器中的例程时，所述驱动器向设备发出命令。一旦设备将数据发送回到驱动器，所述驱动器就可以调用原始调用程序中的例程。在一个实施例中，驱动器是与硬件有关的并且是操作系统特定的。所述驱动器常常提供任何必要的异步的时间有关的硬件接口所需要的中断处理。

[0029] 设备驱动器，特别是在现代Microsoft Windows®平台上的，能够以内核模式(环0)或以用户模式(环3)进行运行。以用户模式运行驱动器的主要益处是改进的稳定性，因为写入不良的用户模式设备驱动器不会通过盖写内核存储器来使系统崩溃。另一方面，用户/内核模式转换常常强加相当大的性能开销，从而针对短等待时间和高吞吐量要求禁止用户模式驱动器。内核空间可以由用户模块仅通过使用系统调用来存取。最终用户程序，像UNIX

操作系统外壳或其它基于GUI的应用程序，是用户空间的一部分。这些应用程序通过内核支持的功能来与硬件进行交互。

[0030] CPU 102可以包括(未图示)控制处理器、现场可编程门阵列(FPGA)、专用集成电路(ASIC)或数字信号处理器(DSP)中的一个或多个。例如，CPU 102执行控制计算系统100的操作的控制逻辑，所述控制逻辑包括操作系统108、KMD 110、SWS 112以及应用程序111。在这个说明性实施方案中，根据一个实施方案，CPU 102通过例如以下操作来发起并且控制应用程序111的执行：在CPU 102和其它处理资源(如APD 104)上分配与那个应用程序相关的处理。

[0031] 尤其是APD 104执行用于所选定的功能的命令和程序，所述所选定的功能如图形操作和可能例如特别适用于并行处理的其它操作。一般来说，APD 104可以被频繁地用于执行图形流水线操作(如像素操作、几何计算)，并且将图像渲染至显示器。在本发明的各种实施方案中，APD 104还可以基于从CPU 102所接收的命令或指令来执行计算处理操作(例如，与图形无关的那些操作，例如像视频操作、物理模拟、计算流体动力学等)。

[0032] 例如，命令可以被认为是典型地不是定义在指令集体系结构(ISA)中的特殊指令。可以通过特殊处理器(如分派处理器、命令处理器或网络控制器)来执行命令。另一方面，指令可以被认为是例如计算机体系结构内部的处理器的单一操作。在一个实施例中，当使用两个ISA集时，一些指令被用于执行x86程序，而一些指令被用于执行APD单元上的内核。

[0033] 在一个说明性实施方案中，CPU 102将所选定的命令传输至APD 104。这些所选定的命令可以包括图形命令和服从并行执行的其它命令。可以大致上独立于CPU 102来执行还可以包括计算处理命令在内的这些所选定的命令。

[0034] APD 104可以包括其自己的计算单元(未图示)，如但不限于一个或多个SIMD处理核心。如本文所提及，SIMD是流水线或程序设计模型，其中通过每个处理元件自己的数据和共享的程序计数器，在多个处理元件中的每一个上同时地执行内核。所有处理元件执行一个完全相同的指令集。预测的使用使得工作项目能够参与或不参与每个所发出的命令。

[0035] 在一个实施例中，每个APD 104计算单元可以包括一个或多个标量和/或向量浮点单元和/或算术逻辑单元(ALU)。APD计算单元还可以包括专用处理单元(未图示)，如反平方根单元和正弦/余弦单元。在一个实施例中，APD计算单元在本文中统称为着色器核心122。

[0036] 一般来说，具有一个或多个SIMD使得APD 104理想地适用于数据并行任务(如在图形处理中常见的那些)的执行。

[0037] 一些图形流水线操作(如像素处理)和其它并行计算操作可能要求对输入数据元素的流或集合执行同一命令流或计算内核。同一计算内核的相应实例化可以在着色器核心122中的多个计算单元上同时地执行，以便并行地处理这类数据元素。如本文所提及，例如，计算内核是含有在程序中陈述并且在APD上执行的指令的功能。这个功能还被称为内核、着色器、着色器程序或程序。

[0038] 在一个说明性实施方案中，每个计算单元(例如，SIMD处理核心)可以执行特定工作项目的相应实例化来处理传入数据。工作项目是由命令在设备上所调用的内核的并行执行的集合中的一个。工作项目可以由一个或多个处理元件执行为在APD计算单元上执行的工作群组的一部分。

[0039] 工作项目通过其全局ID和局部ID来与所述集合内的其它执行区别开。在一个实施

例中，一起同时在SIMD上执行的工作群组中的工作项目子集可以被称为波前136。波前的宽度是计算单元(例如，SIMD处理核心)的硬件的特点。如本文所提及，工作群组是在单一计算单元上执行的相关工作项目的集合。群组中的工作项目执行同一内核并且共享本地存储器和工作群组屏障。

[0040] 在示例性实施方案中，在同一SIMD处理核心上处理来自工作群组的所有波前。一次一个地发出波前上的指令，并且在所有工作项目遵循同一控制流时，每个工作项目执行同一程序。波前还可以被称为弯曲、向量或线程。

[0041] 执行掩码和工作项目预测用于使得发散的控制流能够在一个波前内，其中每个单独的工作项目实际上可以采取通过内核的唯一的代码路径。当工作项目的全集不可在波前开始时间使用时，可以处理部分填充的波前。例如，着色器核心122可以同时执行预定数量的波前136，每个波前136包括多个工作项目。

[0042] 在系统100内，APD 104包括其自己的存储器，如图形存储器130(但存储器130不限于仅供图形使用)。图形存储器130提供用于在APD 104中进行计算期间使用的本地存储器。着色器核心122内的单独计算单元(未图示)可以具有其自己的本地数据储存器(未图示)。在一个实施方案中，APD 104包括存取本地图形存储器130以及存取存储器106。在另一个实施方案中，APD 104可以包括存取动态随机存取存储器(DRAM)或直接附接至APD 104并且与存储器106分离的其它此类存储器(未图示)。

[0043] 在所示实施例中，APD 104还包括一个或“n”数量个命令处理器(CP)124。CP 124控制APD 104内的处理。CP 124还从存储器106中的命令缓冲区125检索待执行的命令，并且对这些命令在APD 104上的执行进行协调。

[0044] 在一个实施例中，CPU 102将基于应用程序111的命令输入适当的命令缓冲区125中。如本文所提及，应用程序是将在CPU和APD内的计算单元上执行的程序部分的组合。

[0045] 多个命令缓冲区125可以用被调度来在APD 104上执行的每个进程来维护。

[0046] CP 124可以用硬件、固件或软件或其组合来实施。在一个实施方案中，CP 124被实施为具有用于实施包括调度逻辑在内的逻辑的微代码的精简指令集计算机(RISC)引擎。

[0047] APD 104还包括一个或“n”数量个分派控制器(DC)126。在本申请中，术语“分派”是指由分派控制器执行的命令，所述分派控制器使用上下文状态来为计算单元集合上的工作群组集合发起内核的执行的开始。DC 126包括用以发起着色器核心122中的工作群组的逻辑。在一些实施方案中，DC 126可以被实施为CP 124的一部分。

[0048] 系统100还包括用于从运行列表150选择进程以在APD 104上执行的硬件调度器(HWS)128。HWS 128可以使用循环法、优先级或基于其它调度策略来从运行列表150选择进程。例如，可以动态地确定优先级。HWS 128还可以包括用以管理运行列表150的功能性，例如通过添加新的进程以及通过从运行列表150删除现有进程来管理。HWS 128的运行列表管理逻辑有时被称为运行列表控制器(RLC)。

[0049] 在本发明的各种实施方案中，当HWS 128发起执行来自RLC 150的进程时，CP 124开始从对应的命令缓冲区125检索并且执行命令。在一些情况下，CP 124可以生成待在APD 104内部执行的一个或多个命令，这些命令对应于从CPU 102接收的命令。在一个实施方案中，CP 124与其它部件一起对APD 104上的命令进行区分优先次序并且调度，其方式为改进或最大化对APD 104和/或系统100的资源的利用率。

[0050] APD 104可以存取或可以包括中断生成器146。中断生成器146可以由APD 104配置来在APD 104遇到如页面错误等中断事件时中断操作系统108。例如，APD 104可以依赖于IOMMU 116内的中断生成逻辑来产生以上所指出的页面错误中断。

[0051] APD 104还可以包括用于抢先取得当前正在着色器核心122内运行的一个进程的抢先和上下文切换逻辑120。例如，上下文切换逻辑120包括用以停止所述进程并且保存其当前状态(例如，着色器核心122状态和CP 124状态)的功能性。

[0052] 如本文所提及，术语“状态”可以包括初始状态、中间状态和/或最终状态。初始状态是机器根据程序设计次序处理输入数据集以产生数据输出集合的开始点。存在例如需要在几个点处被存储以使得处理能够向前进的中间状态。这个中间状态有时被存储来允许当由某一其它进程中断时在稍后时间处继续执行。还存在可以被记录为输出数据集的一部分的最终状态。

[0053] 抢先和上下文切换逻辑120还可以包括用以将另一个进程上下文切换至APD 104中的逻辑。用以将另一个进程上下文切换成在APD 104上运行的功能性可以包括例如通过CP 124和DC 126来实例化所述进程以在APD 104上运行，为这个进程恢复任何先前保存的状态，并且开始其执行。

[0054] 存储器106可以包括非永久性存储器，如DRAM(未图示)。存储器106可以在执行应用程序或其它处理逻辑的若干部分期间存储例如处理逻辑指令、常量值以及变量值。例如，在一个实施方案中，用以在CPU 102上执行一个或多个操作的控制逻辑的若干部分可以在由CPU 102执行操作的相应部分期间驻留在存储器106内。

[0055] 在执行期间，相应的应用程序、操作系统功能、处理逻辑命令以及系统软件可以驻留在存储器106中。对操作系统108很重要的控制逻辑命令在执行期间通常将驻留在存储器106中。包括例如KMD 110和软件调度器112在内的其它软件命令在系统100的执行期间也可以驻留在存储器106中。

[0056] 在这个实施例中，存储器106包括由CPU 102使用来将命令发送至APD 104的命令缓冲区125。存储器106还包含进程列表和进程信息(例如，活动列表152和进程控制块154)。这些列表以及信息由在CPU 102上执行的调度软件使用来将调度信息传递至APD 104和/或相关调度硬件。存取存储器106可以由耦合到存储器106的存储器控制器140管理。例如，来自CPU 102或来自其它设备的对从存储器106读取或写入存储器106的请求由所述存储器控制器140管理。

[0057] 转回参看系统100的其它方面，IOMMU 116是一个多上下文存储器管理单元。

[0058] 如本文所使用，上下文可以认为是内核在其中执行的环境和在其中定义同步与存储器管理的领域。上下文包括设备集合、可由这些设备存取的存储器、对应的存储器特性以及用来调度一个或多个内核的执行或在存储器对象上的操作的一个或多个命令队列。

[0059] 转回参看图1A中所示的实施例，IOMMU 116包括用以执行用于包括APD 104在内的设备的存储器页面存取的虚拟至物理地址翻译的逻辑。IOMMU 116还可以包括用以生成中断的逻辑，例如当由如APD 104等设备的页面存取导致页面错误时生成中断。IOMMU 116还可以包括或能够存取翻译旁视缓冲区(TLB)118。作为实例，TLB 118可以在内容可寻址存储器(CAM)中实施，以便应由APD 104对存储器106中的数据所做出的请求而加速逻辑(即，虚拟)存储器地址至物理存储器地址的翻译。

[0060] 在所示实施例中，通信基础设施109视需要互连系统100的部件。通信基础设施109可以包括(未图示)外围部件互连(PCI)总线、扩展的PCI(PCI-E)总线、高级微控制器总线体系结构(AMBA)总线、加速图形端口(AGP)或其它此类通信基础设施中的一个或多个。通信基础设施109还可以包括以太网，或类似网络，或满足应用程序的数据传送速率要求的任何适当物理通信基础设施。通信基础设施109包括用以互连包括计算系统100的部件在内的部件的功能性。

[0061] 在这个实施例中，操作系统108包括用以管理系统100的硬件部件以及用以提供常见服务的功能性。在各种实施方案中，操作系统108可以在CPU 102上执行，并且提供常见服务。这些常见服务可以包括例如调度用于在CPU 102内部执行的应用程序、错误管理、中断服务以及处理其它应用程序的输入和输出。

[0062] 在一些实施方案中，基于由如中断控制器148的中断控制器生成的中断，操作系统108调用适当的中断处理例程。例如，在检测到页面错误中断之后，操作系统108可以即刻调用中断处理程序来起始将相关页面加载到存储器106中，并且更新对应页面表。

[0063] 操作系统108还可以包括用以通过确保以下操作来保护系统100的功能性：存取硬件部件是通过操作系统管理的内核功能性来进行调解。事实上，操作系统108确保了应用程序(如应用程序111)在用户空间中在CPU 102上运行。操作系统108还确保了应用程序111调用由操作系统提供的内核功能性，以便存取硬件和/或输入/输出功能性。

[0064] 举例来说，应用程序111包括用以执行用户计算的各种程序或命令，这些用户计算也在CPU 102上执行。CPU 102能够无缝地发送所选定的命令以用于在APD 104上处理。

[0065] 在一个实施例中，KMD 110实施应用程序设计接口(API)，通过所述应用程序设计接口，CPU 102或在CPU 102上执行的应用程序或其它逻辑可以调用APD 104功能性。例如，KMD 110可以使来自CPU 102的命令排队到命令缓冲区125，APD 104随后将从命令缓冲区检索这些命令。此外，KMD 110可以与SWS 112一起执行待在APD 104上执行的进程的调度。例如，SWS 112可以包括用以维护待在APD上执行的进程的已区分优先次序的列表的逻辑。

[0066] 在本发明的其它实施方案中，在CPU 102上执行的应用程序可以在对命令进行排队时完全绕过KMD 110。

[0067] 在一些实施方案中，SWS 112维护待在APD 104上执行的进程的在存储器106中的活动列表152。SWS 112还在活动列表152中选择进程子集来由硬件中的HWS 128管理。关于在APD 104上运行每个进程的信息通过进程控制块(PCB)154而从CPU 102传递至APD 104。

[0068] 用于应用程序、操作系统以及系统软件的处理逻辑可以包括在如C语言的程序设计语言中和/或如Verilog、RTL或网表的硬件描述语言中指定的命令，以便使得能够最终通过掩模作品(maskwork)/光掩模的产生而配置制造过程，从而生产体现本文所描述的本发明的方面的硬件设备。

[0069] 本领域技术人员在阅读本描述后将了解，计算系统100可以包括比图1A中所示更多或更少的部件。例如，计算系统100可以包括一个或多个输入接口、非易失性储存器、一个或多个输出接口、网络接口以及一个或多个显示器或显示器接口。

[0070] 图1B为示出图1A中所示的APD 104的更详细的图解的实施方案。在图1B中，CP 124可以包括CP流水线124a、124b以及124c。CP 124可以被配置来处理命令列表，这些命令列表被提供为来自图1A中所示的命令缓冲区125的输入。在图1B的示例性操作中，CP输入0

(124a)负责将命令驱动到图形流水线162中。CP输入1和2(124b和124c)将命令转发到计算流水线160。还提供了用于控制HWS128的操作的控制器机构166。

[0071] 在图1B中,图形流水线162可以包括块集合,本文称为有序流水线164。作为一个实例,有序流水线164包括顶点群组翻译器(VGT)164a、图元汇编器(PA)164b、扫描变换器(SC)164c以及着色器输出后期渲染单元(SX/RB)176。有序流水线164内的每个块可以表示图形流水线162内的不同图形处理级。有序流水线164可以是固定功能硬件流水线。

[0072] 可以使用也将在本发明的精神和范围内的其它实施方式。尽管只有少量数据可以被提供为到图形流水线162的输入,但这些数据将在被提供为从图形流水线162的输出时被放大。图形流水线162还包括用于在从CP流水线124a接收的工作项目群组内的整个范围中进行计数的DC 166。通过DC 166提交的计算工作与图形流水线162是半同步的。

[0073] 计算流水线160包括着色器DC 168和170。所述DC168和170中的每一个被配置来在从CP流水线124b和124c接收的工作群组内的整个计算范围中进行计数。

[0074] 在图1B中示出的DC 166、168以及170接收输入范围,将这些范围分解成工作群组,然后将这些工作群组转发到着色器核心122。

[0075] 由于图形流水线162通常是固定功能流水线,因而难以保存并恢复其状态,并且因此,图形流水线162难以进行上下文切换。因此,在大多数情况下,如本文所论述,上下文切换不涉及在图形进程之间进行上下文切换。一个例外是对于在着色器核心122中的图形工作,它可以进行上下文切换。

[0076] 在图形流水线162内部的工作处理已经完成之后,通过后期渲染单元176处理所完成的工作,所述后期渲染单元进行深度和色彩计算,并且然后将其最终结果写入存储器130。

[0077] 着色器核心122可以由图形流水线162和计算流水线160共享。着色器核心122可以是被配置来运行波前的一般处理器。

[0078] 在一个实施例中,在计算流水线160内部的所有工作是在着色器核心122中进行处理的。着色器核心122运行可编程的软件代码,并且包括各种形式的数据,例如状态数据。

[0079] 图2是一种排队系统200的说明性方框图,在所述排队系统中,工作负荷受到平衡并且被重新分布以用于在CPU和APD处理设备上进行处理。排队系统200包括队列202、任务204、信号量块206、CP 124(本文所描述)、一个或多个SIMD调度器208、着色器核心122、CPU同步模块210、CPU离队模块212以及CPU核心214。

[0080] CPU 102包括如本文所描述的一个或多个CPU核心214。每个CPU核心214处理CPU 102中的计算机指令和数据。

[0081] 队列202是从系统存储器106分配的一段存储器。队列根据先进先出("FIFO")原则来操作。也就是,先入队到队列的工作负荷是先从队列离队的工作负荷。此外,本领域技术人员将了解的是,以实例方式而非限制方式给出对具体队列数据结构的论述,并且可以使用其它存储器存储数据结构。

[0082] 队列202是公用队列。公用队列可以由如CPU 102和APD 104等处理设备来存取。队列202存储根据FIFO原则而入队到队列202并且从其离队的多个任务204。任务204是独立作业,这些作业包括为在APD 104或CPU 102上进行处理而调度的操作系统指令、应用程序指令、图像以及数据。一个作业根据"粒度"而分成多个任务204,其中粒度表示任务204的大

小。粒度的大小针对为APD 104和CPU 102处理器调度的任务204而变化。例如，针对在CPU102上处理的任务204的粒度大小通常小于针对在APD 104上处理的任务204的粒度大小。

[0083] 任务204包括含有指向需要处理的数据的信息指令和/或指针的数据结构。例如，含有用于任务204的信息的数据结构可以被定义为MyTask结构。在一个非限制性实施例中，MyTask结构可以包括以下参数：

```
struct MyTask {
    MyPtr myCodePtr
        myCPUCodePtr : 指向代码的指针(x86 二进制格式)
        myAPDCodePtr :
            //指向代码的指针(着色器二进制格式)
```

[0084]

```
    MyPtr myDataPtr :
        myExecRange:
            //全局网格尺寸
            //局部网格尺寸
        myArgSize
        myArgs {(变量大小)}
    MyNotification
        //指向通知机制的指针
```

[0085]

```
    MyAffinity
        //处理优选项
}
```

[0086] MyTask结构包括指向存储在系统存储器106或另一个存储器设备中的所编译的CPU代码和APD微代码的指针。在以上实施例中，MyPtr myCodePtr将指向在CP 124上执行的微代码的指针定义为myAPDCodePtr，并且将指向在CPU 102上执行的所编译的源代码的指针定义为myCPUCodePtr。myAPDCodePtr指向包括一个功能的微代码，着色器核心122使用所述功能来执行任务204中的数据。例如，如果任务204在APD 104上执行，则APD 104存取其地址被存储在myAPDCodePtr中的功能。如果任务204在CPU 102上执行，则CPU 102存取其地址被存储在myCPUCodePtr中的功能。在一个实施方案中，myCodePtr还可以指向中间语言表示，所述中间语言表示包括在预先确定的事件发生之后变为可执行的依赖关系信息。

[0087] 在以上实施例中，MyTask结构可以包括MyPtr myDataPtr。myDataPtr是指向系统存储器106中的任务204需要处理的数据的位置的指针。此外，myDataPtr包括多个参数，所述参数包括与任务204中的数据相关的信息。例如，参数myArgs包括一列自变量，myArgSize包括自变量的数量，并且myExecRange包括数据网格的尺寸。

[0088] 在本发明的实施方案中，MyTask结构还包括MyAffinity参数。MyAffinity的值确定执行任务204的处理设备。例如，MyAffinity参数的值可以为如CPU102 或APD 104的处理设备指示优选项、要求、提示等。

[0089] 本领域技术人员将了解到，如MyTask的数据结构还可以包括其它参数。

[0090] CPU离队模块212和CP 124充当离队实体。离队实体使任务从队列202中离队或移除以用于在处理设备上进行处理。

[0091] CPU离队模块212为存取队列202并且移除任务204以用于在CPU 102上处理的软件模块。在一个实施方案中，CPU离队模块212在CPU 102需要处理任务204时将任务从与APD 104相关的队列202中移除。例如，当与CPU 102相关的队列202为空时，而与APD 104相关的队列202则存储需要处理的任务204。

[0092] 通常，CPU离队模块212使用FIFO原则来检索任务204。在移除一个或多个任务204之前，CPU离队模块212存取MyAffinity参数以确定任务204是否适于在CPU 102上处理。例如，CPU离队模块212在MyAffinity参数未被设置成作为要求而在APD 104上进行处理的情况下使一个或多个任务204离队。在另一个实施例中，CPU离队模块212在MyAffinity参数未被设置成作为优选项而在APD 104上进行处理的情况下使一个或多个任务204离队。通常，包括可以由并行处理器执行的数学上复杂的操作的一个或多个任务204可以具有被设置成作为优选项或要求的APD 104处理的MyAffinity参数。

[0093] 在多CPU核心214环境中，CPU离队模块212对应于具体CPU核心214。

[0094] CP 124存取队列202并且移除任务204以用于在APD 104上进行处理。CP 124是将任务204从队列202中移除以用于在APD 104上进行处理的硬件模块。类似于CPU离队模块212，当与APD 104相关的队列202为空而与CPU 102相关的队列202存储需要处理的任务204时，CP 124可以将任务204从与CPU 102相关的队列202中移除。

[0095] CP 124根据FIFO原则来检索任务204。在移除一个或多个任务204之前，CP 124使用MyAffinity参数来确定任务204是否适于在APD 104上进行处理。例如，CP 124在MyAffinity参数未被设置成作为要求而在CPU 102上进行处理的情况下使一个或多个任务204离队。在另一个实施例中，CP 124在MyAffinity参数未被设置成作为优选项而在CPU 102上进行处理的情况下使一个或多个任务204离队。通常，包括枝状代码的一个或多个任务204可以具有被设置成作为优选项或要求的CPU 102处理的MyAffinity参数。

[0096] 在CP 124移除任务204之后，CP 124将任务204转发至一个或多个着色器管插补器(SPI)208。SPI 208准备任务204来在着色器核心122上进行处理。在一个实施方案中，SPI 208确定工作项目的数量和被需要来处理任务204的着色器核心122的数量。

[0097] 在CPU离队模块212和CP 124将任务204从队列202移除之前，对这些任务进行同步。同步确保了在任务204被移除时连续并且独占地存取队列202。CPU同步模块210在CPU离队模块212将任务204从队列202移除之前将CPU离队模块212与队列202和APD 104进行同步。CPU同步模块210保证了CPU离队模块212在其尝试移除任务204以用于在CPU 102上进行处理时独有地存取队列202。

[0098] CPU同步模块210使用原子操作来确保CPU离队模块212独占地存取队列202。本领域技术人员将了解，原子操作阻止进程或硬件设备从一个存储器位置读取或对其写入，直到存取所述存储器位置的另一个进程或硬件设备完成存取为止。

[0099] 在移除任务204以用于在APD 104上进行处理之前，信号量块206将CP 124与队列202和CPU 102进行同步。信号量块206还为CP 124保证了独占存取队列202。在一个实施方案中，信号量块206使用原子操作来确保CP 124独占地存取队列202。在另一个实施方案中，信号量块206使用事件通知机制来保证独占地存取队列202。本领域技术人员将了解，事件通知机制通知一个进程或硬件设备：具体存储器位置正由另一个进程或硬件设备存取。

[0100] APD 104和CPU 102从队列202中检索不同数量的任务204。本领域技术人员将了解的是，因为APD 104能够并行处理更多任务204，所以APD 104会检索更多任务204。因此，当CP 124和CPU离队模块212从队列202中检索任务204时，每个离队设备从队列202中移除的任务204的数量取决于是APD 104还是CPU 102请求处理。

[0101] 在离散处理器环境中，信号量块216可能不能够直接对队列202进行同步并且需要额外部件。图3是用于在离散处理环境中重新分布工作负荷的排队系统的方框图。除本文所描述的部件之外，在离散系统环境中，APD 104包括APD驱动器模块302和用以使任务204从队列202离队的APD离队模块304。APD驱动器模块302是在APD 104上控制总执行的软件模块。APD离队模块302是从队列202检索任务204的基于软件的模块。

[0102] 当APD 104请求工作时，信号量块206与APD驱动器模块302进行通信。APD驱动器模块302与APD离队模块304进行通信。APD离队模块304从队列202移除任务204并且将任务204提交给CP124。

[0103] 图4是包括与CPU 102和APD 104进行通信的多个队列202的操作环境400的方框图。

[0104] 尽管每个队列202可以与多个CPU 102和APD 104进行通信，但是队列202可以主要地为特定CPU 102、特定CPU核心214或特定APD 104存储任务。

[0105] CP 124可以将任务204从与CPU 102相关的多个队列202移除，并且将任务204转发至APD 104以用于处理，如本文所描述。类似地，CPU离队模块212可以将任务204从与APD 104相关的多个队列202移除以用于在CPU 102上进行处理，如本文所描述。

[0106] 图5是CP 124将任务204从队列202移除的示例性实施方案的流程图500。

[0107] 在操作502处，APD 104请求需要处理的任务204。

[0108] 在操作504处，CP 124存取队列202。

[0109] 在操作506处，CP 124识别需要处理并且可以在APD 104上处理的一个或多个任务204。例如，CP 124识别一个或多个任务204中的MyAffinity参数的值。在一个实施方案中，CP 124识别被调度来从队列202离队的任务204中的MyAffinity参数。如果CP 124识别一个或多个任务204，那么流程图进行到操作508。否则，流程图500结束。

[0110] 在操作508处，信号量块206对队列202与CPU同步模块210进行同步。

[0111] 在操作510处，CP 124使一个或多个任务204从队列202离队。

[0112] 在操作512处，CP 124将一个或多个任务204发送至SPI 208。

[0113] 在操作514处，SPI 208确定在着色器核心122上处理一个或多个任务204所需要的资源。

[0114] 在操作516处，在着色器核心122上处理任务204。

[0115] 图6是APD离队模块210将任务204从队列202移除的示例性实施方案的流程图600。

[0116] 在操作602处，CPU 102向队列202请求一个或多个任务204。

[0117] 在操作604处，CPU离队模块212识别需要处理并且可以在CPU 102上进行处理的一个或多个任务204。例如，CP 124识别一个或多个任务204中的MyAffinity参数的值。在一个实施方案中，CP 124识别被调度来从队列202离队的一个或多个任务204中的MyAffinity参数。如果CPU离队模块212识别一个或多个任务204，那么流程图进行到操作606。否则，流程图结束。

[0118] 在操作606处，CPU同步模块212对队列202与APD 104进行同步，以使得仅有CPU离队模块212存取队列202。

[0119] 在操作608处，CPU离队模块212如本文所描述地将任务204从队列202移除，并且发送任务204以用于在CPU 102上进行处理。

[0120] 在操作610处，CPU 102处理任务204。

[0121] 图7是APD离队模块304将任务204从队列202移除以用于在离散环境中的APD 104上进行处理的示例性实施方案的流程图700。

[0122] 在操作702处，APD 104请求需要处理的一个或多个任务204，如在操作502中所描述。

[0123] 在操作706处，APD 104将对一个或多个任务204的请求发送至APD驱动器模块302。

[0124] 在操作708处，APD驱动器模块302将请求发送至APD离队模块304。

[0125] 在操作710处，APD离队模块304识别需要处理并且可以在APD 104上处理的一个或多个任务，如在操作506中所描述。

[0126] 在操作712处，信号量块206对队列202与CPU同步模块210进行同步，如在操作508中所描述。

[0127] 在操作714处，APD离队模块304如在操作510中所描述地使任务204离队以用于在APD 104上进行处理并且将任务204发送至APD驱动器模块302。

[0128] 在操作716处，APD驱动器模块302将任务204发送至APD 104，其中如在操作508至512中所描述地处理任务204。

[0129] 本发明的各种方面可以通过软件、固件、硬件或其组合来实施。举例来说，由图5的流程图500、图6的流程图600、图7的流程图700所示的方法可以在图1的统一计算系统100中实施。根据这个实例性统一计算系统100来描述本发明的各种实施方案。相关领域技术人员将明白如何使用其它计算机系统和/或计算机体系结构来实施本发明。

[0130] 在本文献中，术语“计算机程序介质”和“计算机可用介质”用以大体上指代如可装卸存储单元或硬盘驱动器等介质。计算机程序介质和计算机可用介质还可以指代存储器，如系统存储器106和图形存储器130，其可以是存储器半导体(例如，DRAM等)。这些计算机程序产品是用于向统一计算系统100提供软件的手段。

[0131] 本发明还针对包括存储在任何计算机可用介质上的软件的计算机程序产品。这些软件当在一个或多个数据处理设备中执行时致使数据处理设备如本文中描述那样来操作，或者，如上文提到，允许合成和/或制造计算设备(例如，ASIC或处理器)来执行本文中所描述的本发明的实施方案。本发明的实施方案采用现在已知或将来知道的任何计算机可用或可读介质。计算机可用介质的实例包括(但不限于)主要存储设备(例如，任何类型的随机存取存储器)、次要存储设备(例如，硬盘驱动器、软盘、CD ROM、ZIP磁盘、磁带、磁性存储设备、光学存储设备、MEMS、纳米技术存储设备等)以及通信介质(例如，有线和无线通信网络、局

域网、广域网、企业内部网等)。

[0132] 尽管上文已经描述了本发明的各种实施方案,但应当了解,仅以实例方式而非限制方式来呈现所述各种实施方案。相关领域技术人员将了解,可以在不脱离如所附权利要求中所界定的本发明的精神和范围的情况下在其中做出各种形式和细节改变。应了解,本发明不限于这些实施例。本发明适用于如本文所描述那样进行操作的任何元件。因此,本发明的宽度和范围不应被任何上文描述的示例性实施方案所限制,而是应该仅根据所附权利要求及其等效物来限定。

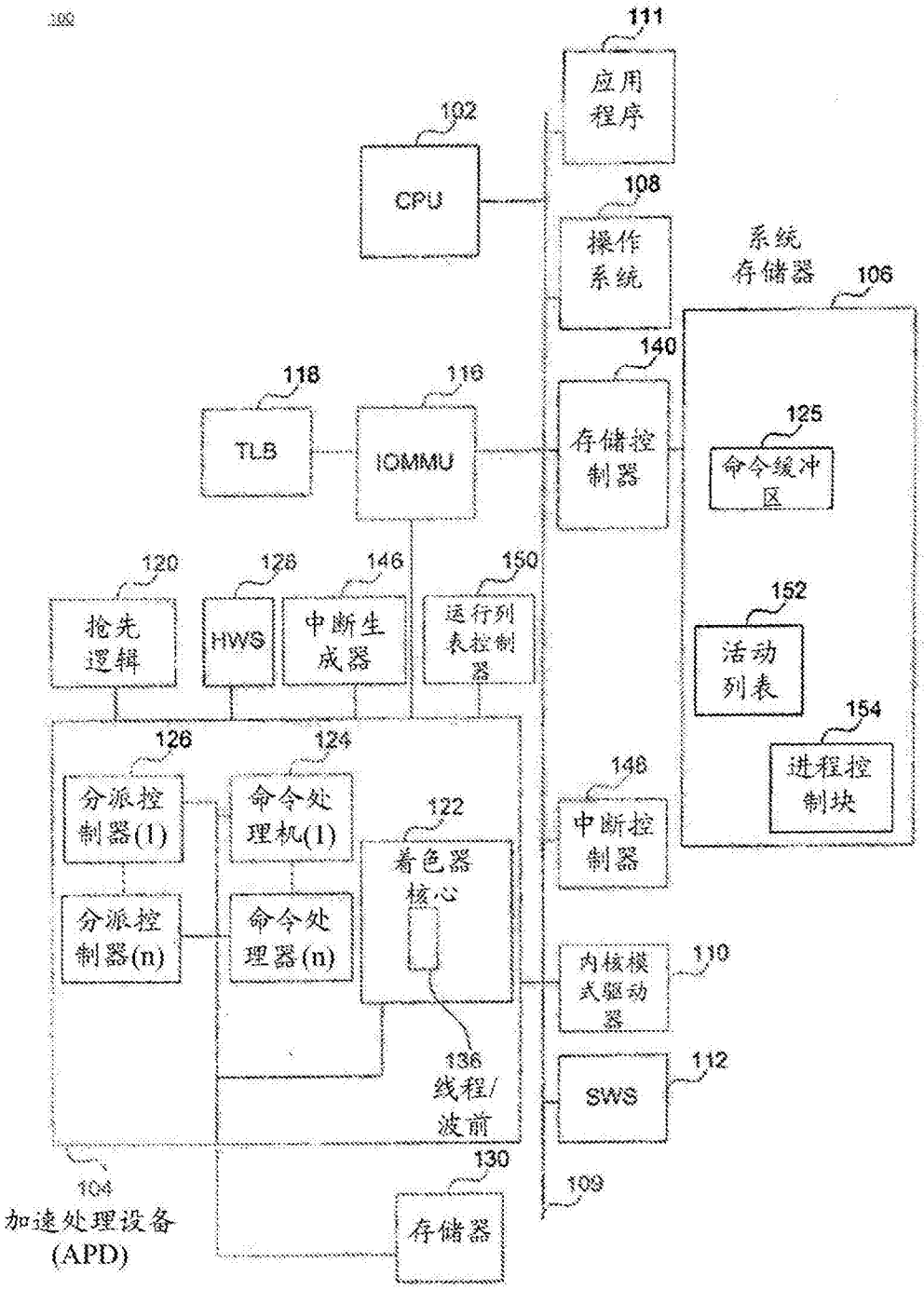

100
111
应用程序
102
CPU
108
操作系统
系统存储器
106
140
存储控制器
118
TLB
116
IOMMU
125
命令缓冲区
120
抢先逻辑
128
HWS
146
中断生成器
150
运行列表控制器
152
活动列表
154
进程控制块
126
分派控制器(1)
124
命令处理机(1)
122
着色器核心
分派控制器(n)
命令处理器(n)
148
中断控制器
110
内核模式驱动器
112
SWS
136
线程/波前
104
加速处理设备(APD)
130
存储器
109

图1A

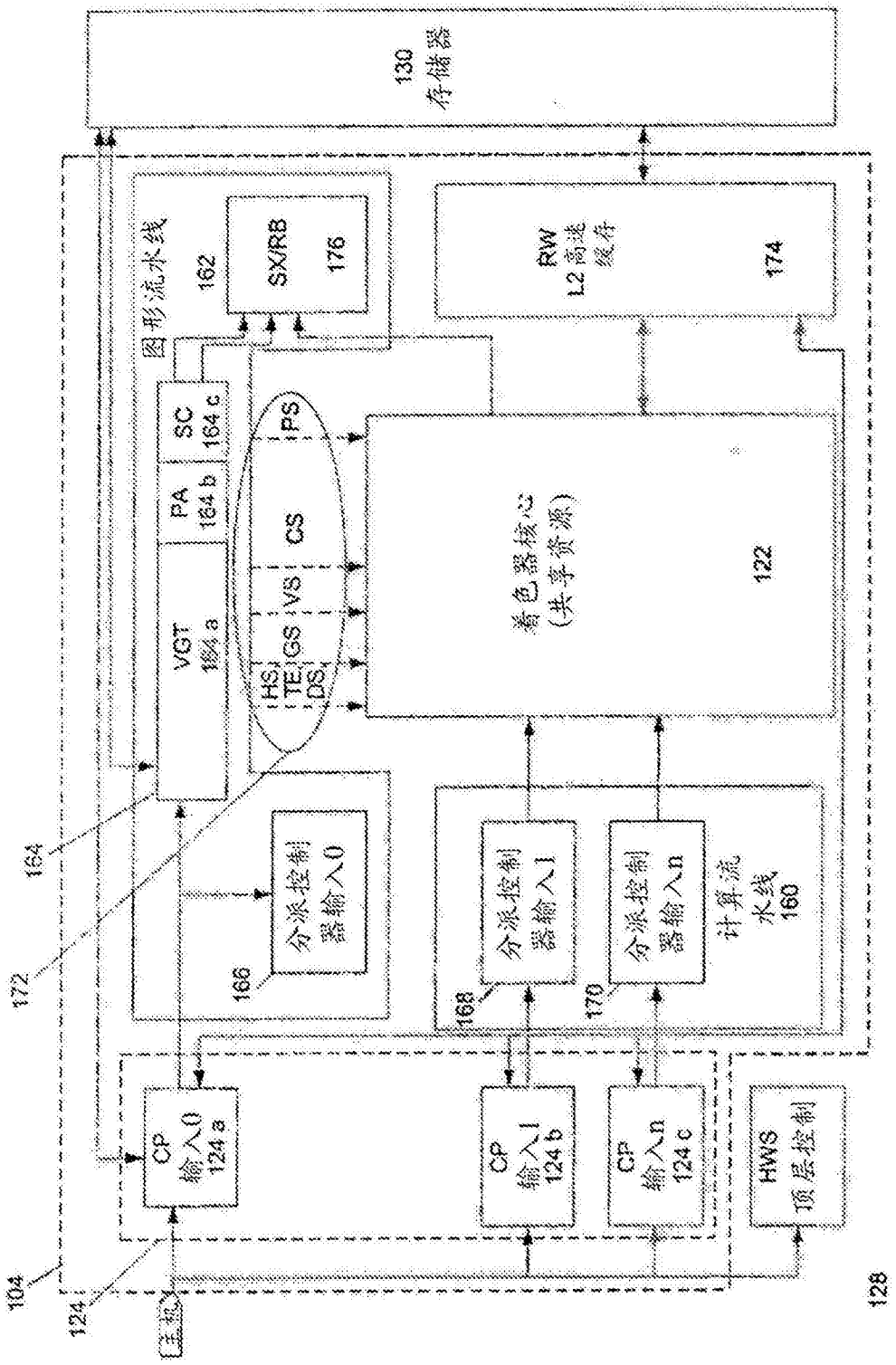

130
存储器
图形流水线
162
SX/RB
176
RW
L2 高速缓存
174
SC
164 c
PA
164 b
VGT
164 a
PS
CS
VS
GS
HS
TE
DS
着色器核心
(共享资源)
122
分派控制器输入0
分派控制器输入1
分派控制器输入n
计算流水线
160
166
168
170
164
172
CP
输入0
124 a
CP
输入1
124 b
CP
输入n
124 c
HWS
顶层控制
104
124
128

图1B

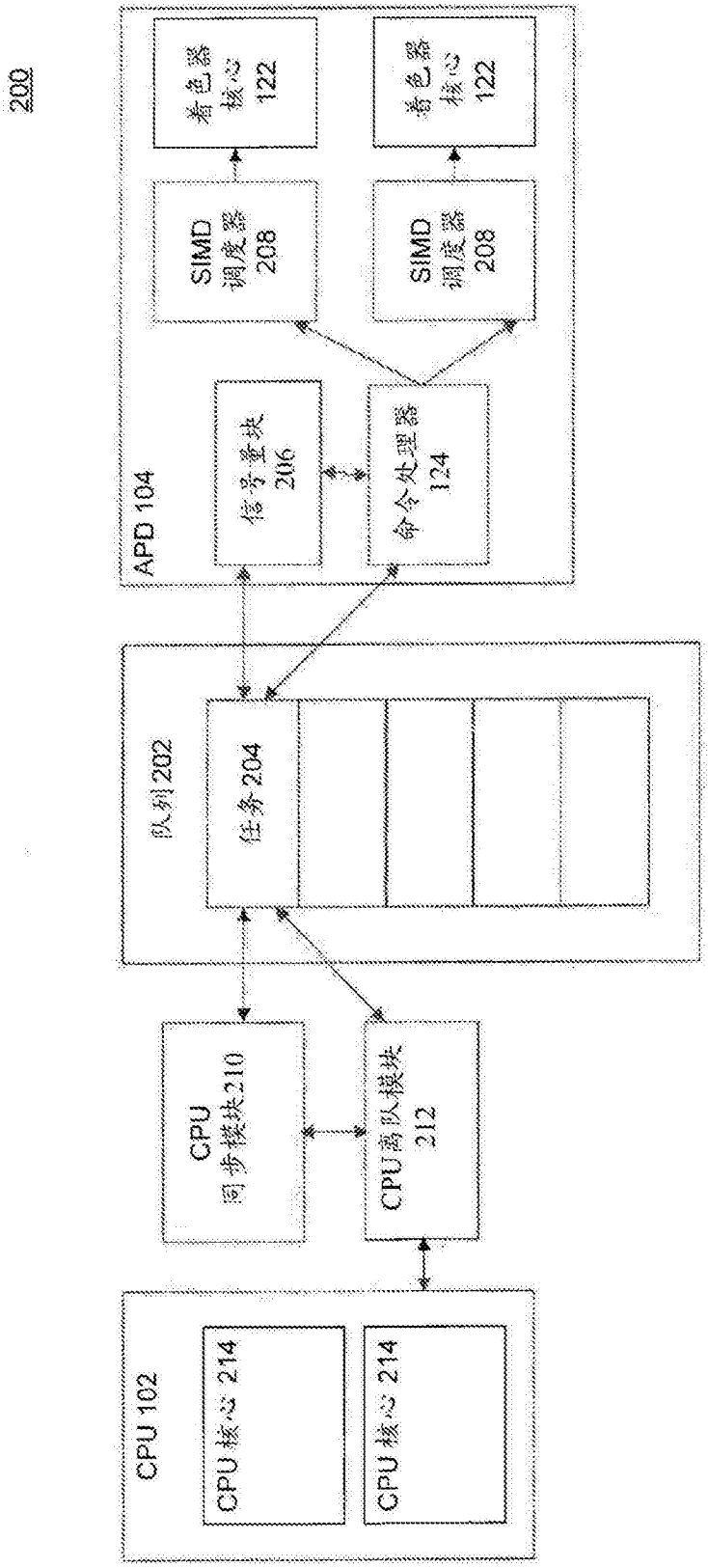

200
APD 104
着色器核心 122
着色器核心 122
SIMD 调度器 208
SIMD 调度器 208
信号量块 216
命令处理器 124
队列 202
任务 204
CPU 同步模块 210
CPU 离队模块 212
CPU 102
CPU 核心 214
CPU 核心 214

图2

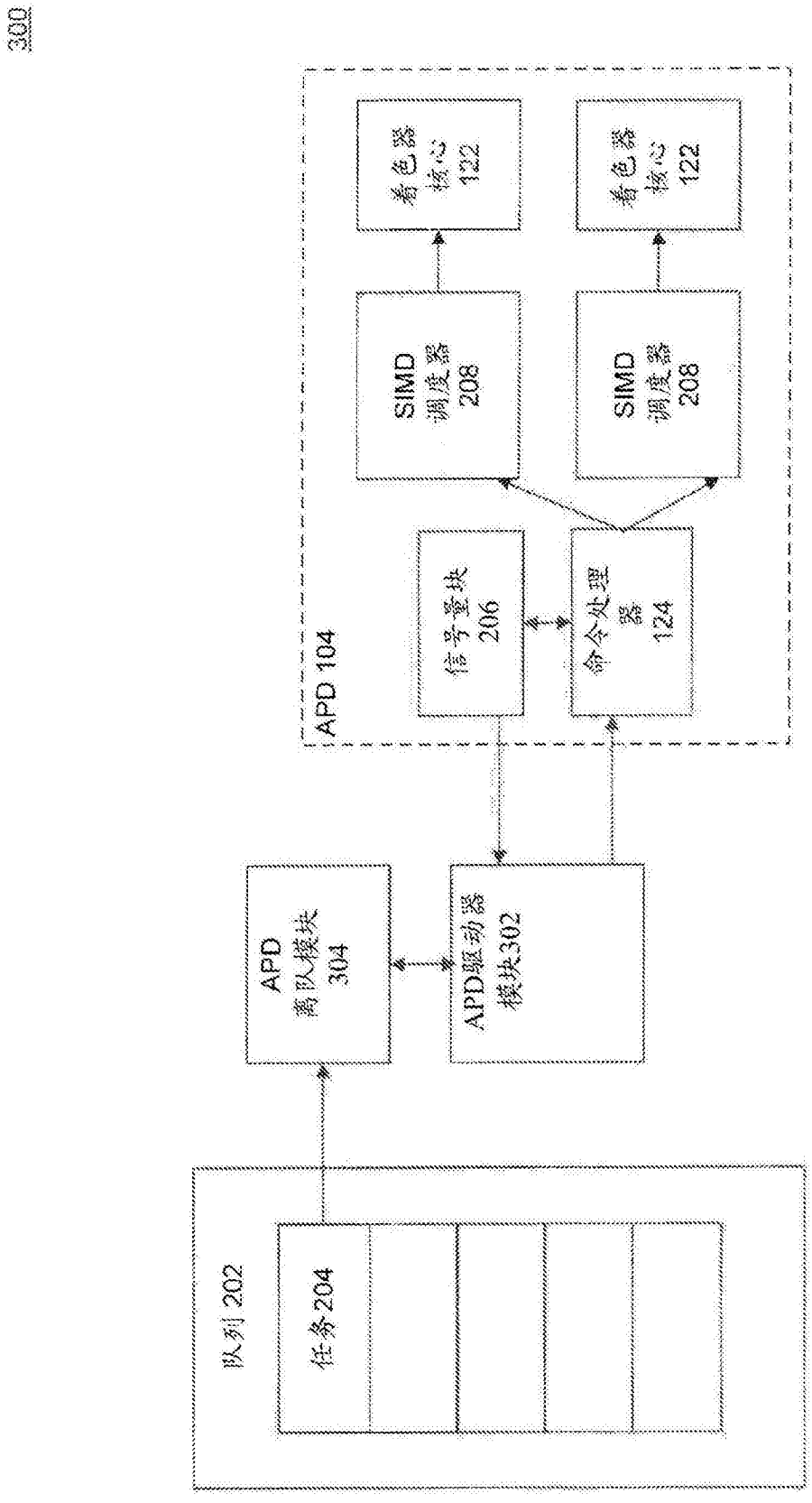

300
APD 104
着色器核心 122
着色器核心 122
SIMD 调度器 208
SIMD 调度器 208
信号量块 206
命令处理器 124
APD 离队模块 304
APD驱动器模块302
队列 202
任务204

图3

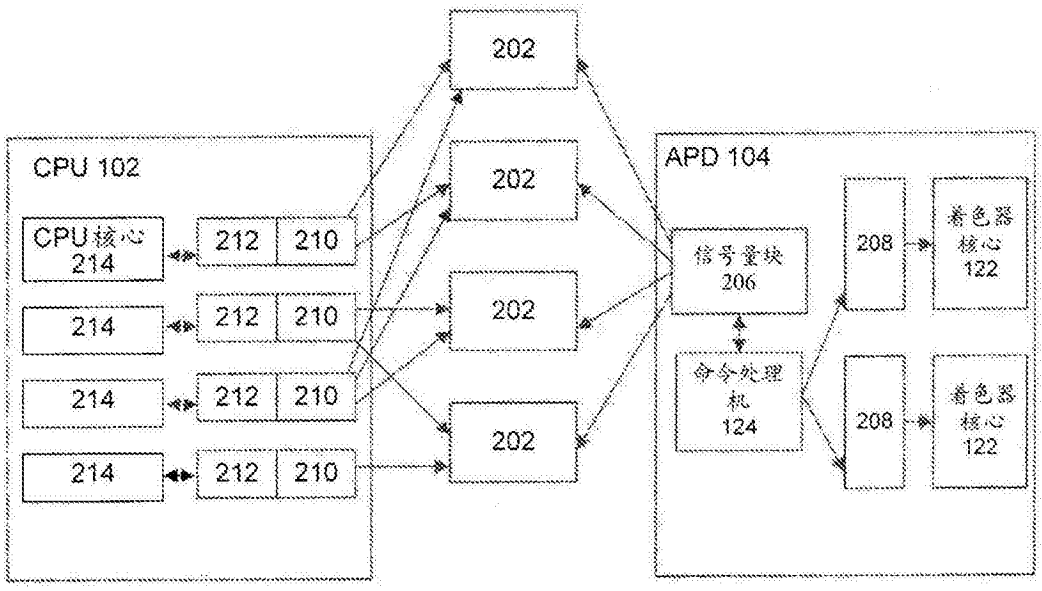

CPU 102
CPU核心 214
214
214
214
212
210
212
210
212
210
212
210
202
202
202
202
APD 104
信号量块 206
命令处理机 124
208
208
着色器核心 122
着色器核心 122

图4

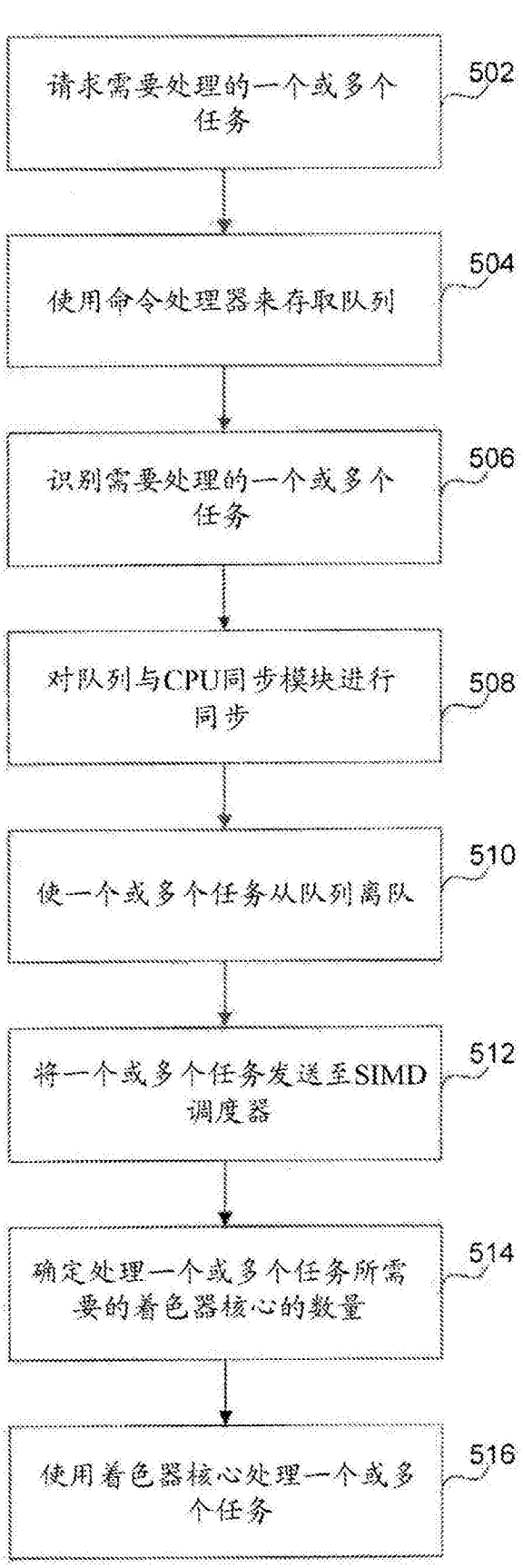

500
请求需要处理的一个或多个任务
502
使用命令处理器来存取队列
504
识别需要处理的一个或多个任务
506
对队列与CPU同步模块进行同步
508
使一个或多个任务从队列离队
510
将一个或多个任务发送至SIMD调度器
512
确定处理一个或多个任务所需要的着色器核心的数量
514
使用着色器核心处理一个或多个任务
516

图5

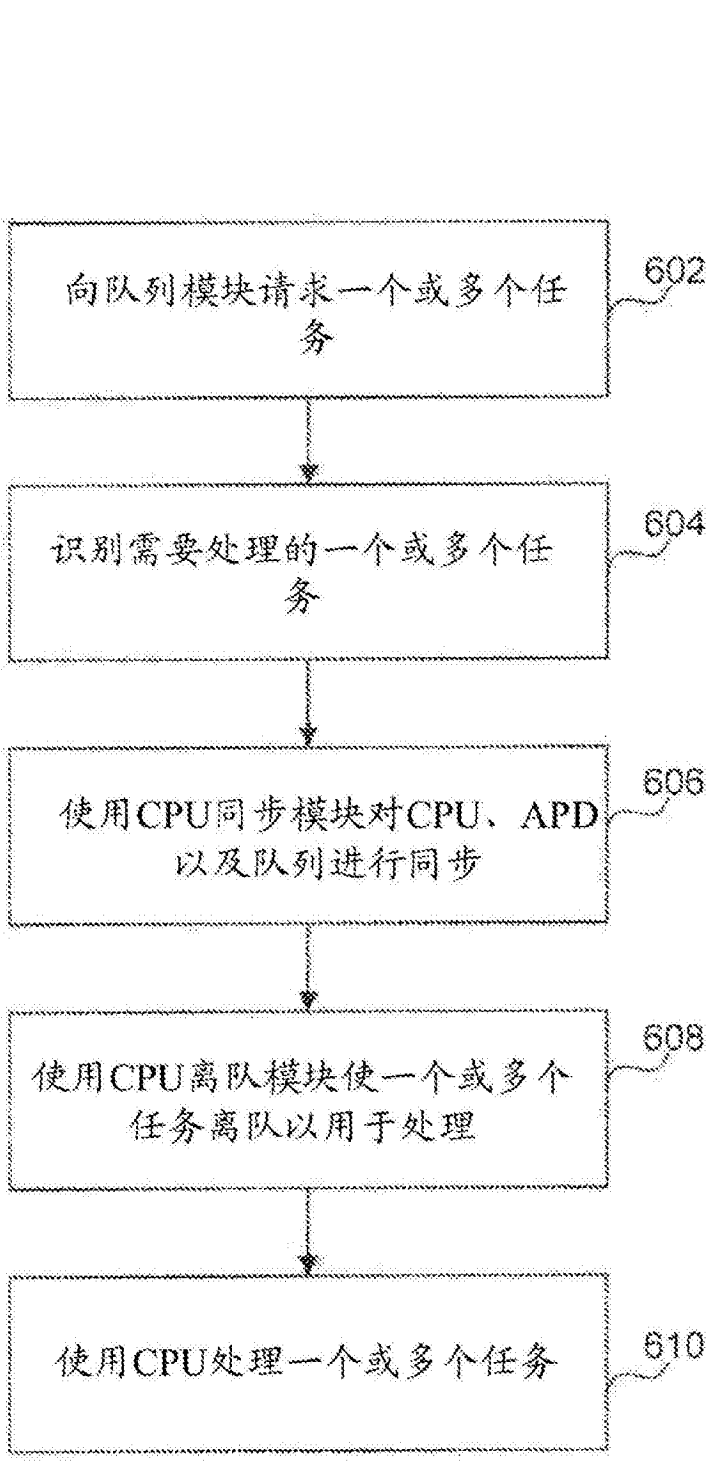

600
向队列模块请求一个或多个任务
602
识别需要处理的一个或多个任务
604
使用CPU同步模块对CPU、APD以及队列进行同步
606
使用CPU离队模块使一个或多个任务离队以用于处理
608
使用CPU处理一个或多个任务
610

图6

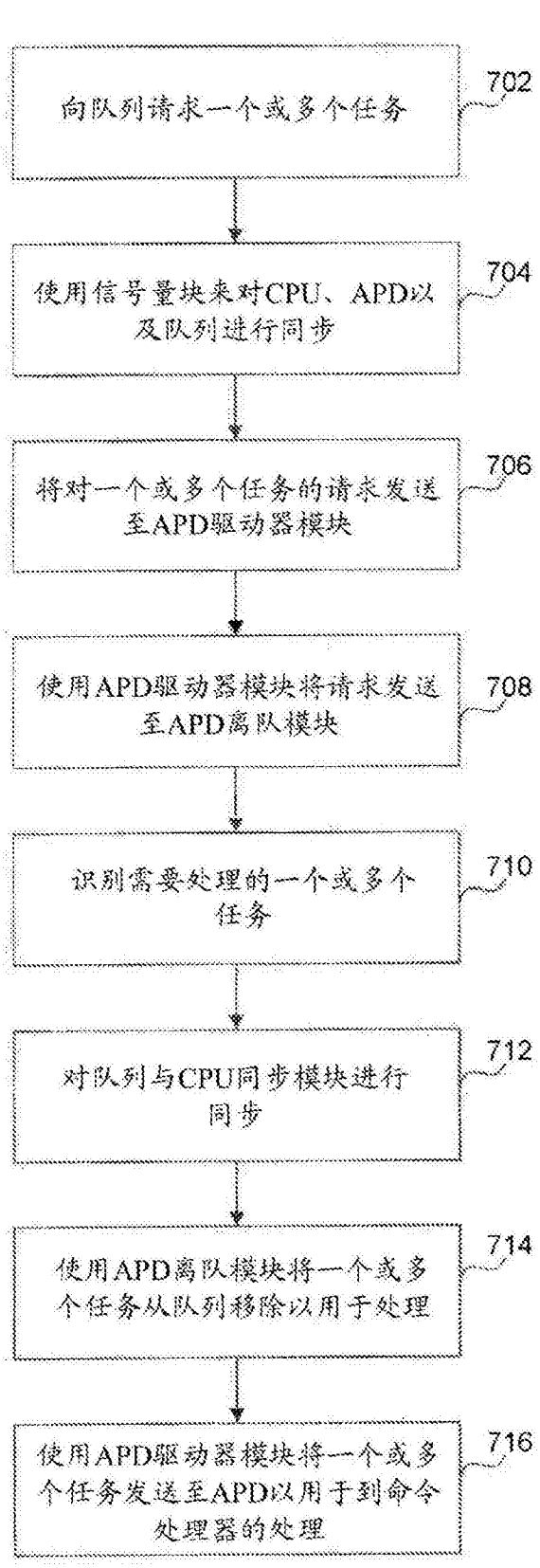

700
向队列请求一个或多个任务
702
使用信号量块来对CPU、APD以及队列进行同步
704
将对一个或多个任务的请求发送至APD驱动器模块
706
使用APD驱动器模块将请求发送至APD离队模块
708
识别需要处理的一个或多个任务
710
对队列与CPU同步模块进行同步
712
使用APD离队模块将一个或多个任务从队列移除以用于处理
714
使用APD驱动器模块将一个或多个任务发送至APD以用于到命令处理器的处理
716

图7