(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 101918982 B
(45)授权公告日 2015.01.21

(21)申请号 200880122341.9

(22)申请日 2008.12.23

(30)优先权数据
61/016,746 2007.12.26 US
12/341,028 2008.12.22 US

(85)PCT国际申请进入国家阶段日
2010.06.22

(86)PCT国际申请的申请数据
PCT/US2008/014011 2008.12.23

(87)PCT国际申请的公布数据
WO2009/085264 EN 2009.07.09

(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·曼特 R·E·麦克拉里

(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.
G06T 15/00(2011.01)
G06T 1/20(2006.01)

(56)对比文件

US 5920326 A,1999.07.06, 说明书第8栏第27行至第第56行,第12栏第7行至第15栏第67行,以及附图1,2,4.

US 5920326 A,1999.07.06, 说明书第8栏第27行至第第56行,第12栏第7行至第15栏第67行,以及附图1,2,4.

CN 101057261 A,2007.10.17, 说明书第4页第16行至第23行,第8页第12行至第10页第20行,第12页第2行至第13页第1行,附图2,3.

US 6462743 B1,2002.10.08, 全文.

US 2007/0103475 A1,2007.05.10, 全文.

WO 02/101649 A1,2002.12.19, 全文.

审查员 石爽

权利要求书1页 说明书9页 附图4页

(54)发明名称

图形流水线的有效状态管理

(57)摘要

本发明提供一种复杂ASIC的有效状态管理系统及其应用。在一实施例中,基于计算机的系统执行依赖状态进程。该基于计算机的系统包括命令处理器(CP)以及多个处理模块。该命令处理器接收命令流内的命令,并管理响应于该命令流内的全局背景事件的全局状态。该多个处理模块接收该命令流内的所述命令,并管理响应于该命令流内的模块背景事件的各个模块状态。每一处理模块分别依据该全局状态及该各个处理模块的模块状态在一数据流内的数据上执行进程。

CN 101918982 B

1. 一种用以执行状态依赖进程的基于计算机的系统，包括：

命令处理器，该命令处理器被配置以接收命令，该命令包括全局状态更新、模块状态更新以及绘图命令中的至少一个，该命令处理器并被配置以管理响应于该全局状态更新的全局状态；

多个处理模块，该处理模块被配置以接收命令流内的命令，并管理响应于该模块状态更新的各个模块状态，其中，每一处理模块分别依据该全局状态及该各个处理模块的该模块状态在数据流内的数据上执行进程；

第一接口模块，该第一接口模块被配置以管理该多个处理模块中的一个处理模块的计数器，其中，该计数器包括模块状态值，指示存储在该多个处理模块的该处理模块中的该模块状态的数量，并被配置以暂缓该多个处理模块接收的该命令流内的命令，以响应该模块状态值超过一阈值；以及

第二接口模块，经由数据总线和控制总线耦合到该多个处理模块的每一处理模块，其中，该第二接口模块被配置以将该模块状态更新写进至该控制总线。

2. 如权利要求 1 所述的基于计算机的系统，其中，该多个处理模块的第一处理模块被配置以管理第一数量的模块状态，以及该多个处理模块的第二处理模块被配置以管理第二数量的模块状态，其中，该第一数量不等于该第二数量。

3. 如权利要求 1 所述的基于计算机的系统，其中，该命令处理器被配置以管理具有指示该全局状态的数量的全局状态值的全局状态计数器，且响应该全局状态值超过另一阈值，该命令处理器被配置以暂缓该多个处理模块接收的该命令。

4. 如权利要求 1 所述的基于计算机的系统，其中，该命令处理器在接收到空闲信号后被配置以重新分配该全局状态的第一全局状态，该空闲信号来自该多个处理模块中最终的处理模块。

5. 如权利要求 1 所述的基于计算机的系统，其中，该基于计算机的系统包括图形处理单元。

6. 一种在专用集成电路 ASIC 内执行状态依赖进程的方法，包括：

管理该专用集成电路 ASIC 的全局状态，该全局状态响应于全局状态更新；

管理多个处理模块内的每一处理模块的模块状态，该模块状态响应于模块状态更新；

依据该全局状态及该各个处理模块的该模块状态，在该多个处理模块的每一处理模块内的数据流内的数据上执行进程；

使用第一接口模块，暂缓该多个处理模块接收的命令，以响应指示该全局状态的数量超过第一阈值的全局状态值、指示该模块状态的数量超过第二阈值的模块状态值、或其组合；以及

使用第二接口模块，将该模块状态更新写进至该控制总线，其中，该第二接口模块经由数据总线和控制总线耦合到该多个处理模块的每一处理模块。

7. 如权利要求 6 所述的方法，其中，管理该模块状态包括管理该多个处理模块的第一处理模块内的第一数量的该模块状态以及该多个处理模块的第二处理模块内的第二数量的该模块状态，其中，该第一数量不等于该第二数量。

8. 如权利要求 6 所述的方法，其中，管理该全局状态包括在接收到空闲信号后重新分配该全局状态的第一全局状态，该空闲信号来自该多个处理模块内的最终的处理模块。

图形流水线的有效状态管理

技术领域

[0001] 本发明涉及执行在计算机系统内的计算操作。

背景技术

[0002] 图形处理单元(GPU)是一被专门设计成执行图形处理任务的专用集成电路(ASIC)。GPU可例如执行终端用户应用所要求的图形处理任务,该终端用户应用例如为视频游戏应用。于此例中,该终端用户应用与该GPU之间有数个软件层。该终端用户应用与应用程序接口(API)交流。API允许该终端用户应用以标准化格式输出图形数据及命令,而不是以依赖该GPU的格式输出图形数据及命令。数个类型的API市面有售,包括微软公司开发的DirectX®以及硅谷图形公司开发的OpenGL®。该API与驱动器(driver)交流。该驱动器将接收自该API的标准码翻译成该GPU理解的指令的原始格式。该驱动器典型地由该GPU制造商写出。接着该GPU执行来自该驱动器的指令。

[0003] 许多GPU包括执行所述指令的图形流水线。图形流水线包括多个处理模块,所述处理模块同时工作于一指令的不同阶段。流水线操作能使GPU利用存在于执行该指令所需的所述阶段中的并行性。因此,GPU可在更短的时间周期内执行更多的指令。

[0004] 该图形流水线的输出依赖该图形流水线的状态。该图形流水线的状态依据状态封包更新,所述状态封包即特定背景常量(context-specific constant)(例如纹理句柄(texture handle)、着色常量(shader constant)、变换矩阵(transform matrix)等等),其被该图形流水线局部存储。由于所述特定背景常量是局部保存,所以可被该图形流水线快速存取。

[0005] 该图形流水线保存的状态封包的数量依赖于该GPU耦接(couple)的API。所述状态封包与普通的API有关,所述状态封包可储存在相对小数量的寄存器内,例如8个寄存器。然而,与普通的API不同,较新的API(例如DirectX® 10)需要相对大数量的频繁背景切换,所述切换关于该流水线的某些方面。与这些频繁背景切换有关的所述状态封包的数量不能被普通图形流水线所保存的相对小数量寄存器所支持。

[0006] 处理与较新API有关的较大数量状态封包的潜在解决方案是仅仅增加该图形流水线支持的状态封包的数量。然而,由于需要额外的寄存器以处理额外的状态封包,这种解决方案将会显着增加晶片面积。此外,如果所述状态封包的数量超过该流水线的储存能力,该图形流水线将会暂缓,因此,这种解决方案可能产生时间问题(timing issue)。

[0007] 另一潜在解决方案将尝试使用软件对增加的状态封包数量进行补偿。例如该驱动器或该终端用户应用可尝试将发送给该GPU的工作重定序以减少状态改变数量(增加每一状态改变所发送的工作)。然而这种解决方案具有至少两个缺点。第一缺点为,这种解决方案仅对一些工作负荷(workload)(一些固有太多状态改变的工作负荷)起作用。第二缺点为,这种解决方案显着增加该CPU检索及分类输入事务(inputtransaction)的工作负荷。

[0008] 综上所述,需要一种提供有效状态管理系统的方法及系统。

发明内容

[0009] 透过提供一有效状态管理系统及其应用，本发明满足上述确定的要求。

[0010] 根据本发明一实施例，提出一执行状态依赖进程（state-depedentprocess）的基于计算机的系统（computer-based system）。该基于计算机的系统包括命令处理器以及多个处理模块。该命令处理器接收命令流内的命令，并管理响应于该命令流内的全局背景事件的全局状态。该多个处理模块接收该命令流内的所述命令，并管理响应于该命令流内的模块背景事件的各个模块状态。该多个处理模块依据该全局状态及所述各个处理模块的模块状态在一数据流内的数据上执行进程。

[0011] 根据本发明另一实施例，提出一在一 ASIC 内执行状态依赖进程的方法。该方法包括管理响应于一命令流内的全局背景事件的该 ASIC 的全局状态、以及分别管理响应于该命令流内的模块背景事件的多个处理模块内的每一处理模块的模块状态。该多个处理模块依据该全局状态及所述各个处理模块的模块状态在一数据流内的数据上执行进程。

[0012] 根据本发明的再一实施例，提出一有形计算机可读储存媒介，包括具体表现在软件内的 ASIC，其中，该 ASIC 执行状态依赖进程。该 ASIC 包括命令处理器以及多个处理模块。该命令处理器接收命令流内的命令，并管理响应于该命令流内的全局背景事件的全局状态。该多个处理模块接收该命令流内的所述命令，并管理响应于该命令流内的模块背景事件的各个模块状态。该多个处理模块依据该全局状态及所述各个处理模块的模块状态在一数据流内的数据上执行进程。

[0013] 以下参考附图详细描述本发明更多的特征及优点和本发明的各种实施例的结构及操作。注意本发明并非限制于本文描述的所述特定实施例。本文介绍的所述实施例仅作说明目的。依据本文包含的所述教导，本领域的技术人员将明了附加的实施例。

附图说明

[0014] 所述附图包含在本文内并形成该说明书的一部分，所述附图与该描述一起阐述本发明，进一步担任解释本发明的原理并使本领域技术人员能制造及使用本发明。

[0015] 图 1 描绘一阐述一范例系统的方块图，该范例系统包括根据本发明一实施例的 GPU ；

[0016] 图 2 描绘一阐述图 1 的 GPU 特征的方块图 ；

[0017] 图 3 描绘一阐述包含在图 1 的 GPU 内的图形流水线细节的方块图 ；以及

[0018] 图 4 描绘一表，该表阐述根据本发明一实施例的全局状态及模块状态的管理。

[0019] 从以下提出的结合附图的实施方式中，本发明的特征及优点将变得更明了。于所述附图中，相同的参考符号始终识别相关的元件。于所述图式内，相同的参考数字大体标示相同的、功能相似及 / 或结构相似的元件。对应参考数字内的该最左数位标示元件首次出现其中的该图式。

具体实施方式

[0020] I. 概述

[0021] 本发明提供一用于复杂 ASIC 的有效状态管理系统及其应用。在以下详细描述中，关于 " 一个实施例 (one embodiment)" ，" 一实施例 (anembodiment)" ，" 一具体实施例

(an example embodiment)″等等表明该所述实施例可包括特定特征、结构或特色，但每一实施例可不必然包括该特定特征、结构或特色。而且，这样的短语不必然涉及相同实施例。此外，当结合一实施例描述特定特征、结构或特色时，我们认为在本领域的技术人员的知识内能够结合其他实施例以变更这样的特征、结构或特色，不论所述实施例是否被明确描述。

[0022] 为阐述目的而非做为限制，根据本发明实施例的有效状态管理系统在本文中从GPU的角度描述，该GPU根据该GPU的图形流水线的全局状态与模块状态执行绘图命令流(steam of draw command)。然而，需明白的是这样的有效状态管理系统可应用于其他类型的系统，所述其他类型的系统根据所述各个系统的状态执行命令流。根据本文所述，本领域技术人员将了解如何在其他类型系统内实施本发明的实施例。

[0023] 如上所述，根据本发明一实施例的有效状态管理系统独立管理一图形流水线的全局状态及模块状态。该全局状态包括预定数量的可能的全局状态(possible global state)，例如，8个可能的全局状态。所述模块状态各自包括预定数量的可能的模块状态，例如2、4、8、16、32、64或一些其他数量的可能的模块状态。对于每一模块，所述可能的模块状态的数量可能不同。该全局状态可相当于与不能频繁改变的场景内所描绘的特征有关的绘画常量，例如该场景内的光线方向。所述模块状态可相当于与该频繁改变的场景内所描绘的特征有关的绘画常量，例如该场景所描绘的由于风所导致的草吹动的方向。

[0024] 如本文所使用，该术语“模块(block)”指包含在ASIC、CPU的执行流水线、及/或GPU的图形流水线内的处理模块。这样的处理模块可包括但并非限制于算术逻辑单元、乘法/除法单元、浮点单元、色彩缓冲器、顶点着色器(vertex shader)、像素着色器、限幅单元(clippingunit)、Z缓冲器、或本领域技术人员明了的一些其他处理模块。

[0025] 由于根据本发明的实施例的图形流水线独立管理全局状态及模块状态，这样的图形流水线可执行绘图命令，该绘图命令是以比普通图形流水线更大数量的不同值为基础。例如普通图形流水线可典型地执行一基于多达8个不同值的绘图命令。相反地，根据本发明的一实施例的图形流水线可执行一基于多达例如256个不同值的绘图命令。

[0026] 该全局状态被一命令处理器(command processor，简称CP)管理。该CP保持一全局状态已使用位(globe-state dirty bit)及一全局状态识别(ID)位。该全局状态已使用位标示该图形流水线的该全局状态是否需要更新。该全局状态ID位标示该图形流水线的全局状态。一全局背景事件(globe context event)后，该全局状态ID位递增。全局背景事件是一输入(数据)流(例如数据封包)内的事件，该输入流标示该全局状态待被更新(to be updated)。该全局状态已使用位由于一绘图呼叫(draw call)而被清除。

[0027] 所述模块状态被该图形流水线的所述各个模块管理。该图形流水线内的每一模块保存一模块状态已使用位及一模块状态ID位。该模块状态已使用位标示一模块的该模块状态是否应该更新。符合一模块的该模块状态ID位标示那个模块的模块状态。在一模块背景事件上，若下一模块状态更新符合那个模块，一模块的模块状态ID位递增。模块背景事件是该输入(数据)流(例如数据封包)内的事件，该输入流标示一模块状态待被更新。于一实施例中，该图形流水线内的模块根据该下一模块状态更新(next block state update)的地址区间确定该下一模块状态更新符合该模块。与该全局状态已使用位相似，该模块状态已使用位由于一绘图呼叫而被清除。

[0028] 一有效状态管理系统可表达在计算机系统内，该计算机系统包括GPU。以下更详细

描述这样的有效状态管理系统的一范例硬件实施及其操作。还描述这样的有效状态管理系统的软件实施。

[0029] II. 有效状态管理系统的一范例硬件实施

[0030] 如上所述，根据本发明一实施例的有效状态管理系统可实施在一 ASIC 内，该 ASIC 执行一命令流。为阐述目的，并非限制，这样的有效状态管理系统是从一范例图形流水线角度描述，该图形流水线包括在一计算机系统内，以下详细描述该计算机系统。

[0031] A. 范例计算机系统概述

[0032] 图 1 是根据一实施例的计算机系统 100 的方块图。系统 100 包括中央处理单元 (CPU)102、图形处理单元 (GPU)110，且可选择地包括协同处理器 (coprocessor)112。此外，计算机系统 100 包括系统存储器 104，该系统存储器 104 可被 CPU 102、GPU 110 以及协同处理器 112 存取。GPU 110 以及协同处理器 112 通过总线 114 与 CPU 102 及系统存储器 104 交流。总线 114 可以是使用在计算机系统内的任何类型总线，包括但并非限制于周边元件扩展接口 (PCI) 总线、加速图形接口 (AGP) 总线以及高速周边元件扩展接口 (PCIE) 总线。GPU 110 及协同处理器 112 透过执行某些特殊功能帮助 CPU 102，GPU 110 及协同处理器 112 执行所述特殊功能通常比 CPU 102 以软件形式执行所述特殊功能快。协同处理器 112 可包括但并非限制于浮点协同处理器、GPU、联网协同处理器以及本领域技术人员明了的其他类型的协同处理器及处理器。

[0033] 系统 100 还包括第一局部存储器 106 及第二局部存储器 108。第一局部存储器 106 耦接至 GPU 110 以及也耦接至总线 114。第二局部存储器 108 耦接至协同处理器 112 并也耦接至总线 114。第一及第二局部存储器 106、108 分别对 GPU 110 及协同处理器 112 有效以便提供尽可能更快的存取某些数据 (例如频繁使用的数据)，如果所述数据储存在系统存储器 104 内的话。

[0034] 于一实施例中，GPU 110 及协同处理器 112 与 CPU 102 一起并行解码指令，并且仅执行那些为它们准备的指令。于另一实施例中，CPU102 将为 GPU 110 及协同处理器 112 准备的指令发送至各自的命令缓冲器。

[0035] 例如，图 2 描绘一阐述一实施例的方块图，于该实施例中，CPU102 将为 GPU 110 准备的指令发送至一命令缓冲器 202。命令缓冲器 202 可位于例如系统存储器 104 内或可为耦接至总线 114 的独立存储器。如图 2 所示，GPU 110 包括程序机 204 以及图形流水线 206。程序机 204 从命令缓冲器 202 取回指令。程序机 204 转送所述指令至图形流水线 206 内的一命令处理器。以下详细描述图形流水线 206。

[0036] B. 范例图形流水线

[0037] 图 3 描绘一方块图，该方块图阐述根据本发明一实施例的图形流水线 206 的细节。如图 3 所示，图形流水线 206 包括命令处理器 (CP)310、寄存器总线管理器 (register bus manager，简称 RBM)320、数据总线传送 (data bus transfer，简称 DBT) 模块 330、模块 A 340a、模块 B 340b、模块 C 340c、以及模块 D 340d。以下详细描述所述元件中的每一元件。

[0038] CP 310 接收例如来自程序机 204 的命令流内的命令，所述命令包括绘图命令、全局状态更新以及模块状态更新。于一实施例中，每一命令包括寄存器地址。于该实施例中，可根据包括在该命令内的该地址是否落在预定地址范围内，将一命令解析 (parse) 为绘图命令、全局状态更新或模块状态更新。然而，本发明并非限制于该实施例。本领域技术人员

将明了在不悖离本发明的精神及范畴下可使用解析命令的其他机制。

[0039] CP 310 还保存二已使用位 :(1) 全局管理已使用 (GMD) 位 312 以标示全局状态更新是否已被接收 ;(2) 模块管理已使用 (BMD) 位 314 以标示模块状态更新是否已被接收。透过例如上电成 (power up as) 逻辑 1,GMD 位 312 及 BMD 位 314 将已使用 (组 (set)) 上电。透过一绘图命令清除该二位。

[0040] CP 310 还管理写进图形流水线 206 的模块 340 的寄存器的全局状态数量。为管理全局状态数量,CP 310 管理全局状态识别 (ID),该全局状态识别 (ID) 标示图形流水线 206 的该全局状态。该全局状态 ID 的范围从 0 变化至图形流水线 206 所承受的全局状态最大数量,例如 8 个全局状态。当 GPU 110 被上电,该全局状态 ID 被设置处于 0 状态。如下详细所述,一全局背景事件被接收以及一新的全局状态写进图形流水线 206 之后,该全局状态 ID 递增。图形流水线 206(例如模块 D340d) 内的一最后的模块发送一脉冲回至 CP 310 之后,一全局状态 ID 被释放 (且之后可被重新使用)。例如,CP 310 可将该全局状态 ID 从 0 递增至 1,但全局状态 ID 0 不能被重新使用直到该最后模块发送一脉冲回至 CP 310 标示与全局状态 ID 0 相对应的该背景已结束。

[0041] CP 310 耦接至 RBM 320。CP 310 写所述绘图命令、所述全局状态更新及所述模块状态更新至 RBM 320。如下详细所述,分配给图形流水线 206 的全局状态数量由 CP 310 管理。

[0042] RBM 320 耦接至 DBT 模块 330。RBM 320 写所述绘图命令、所述全局状态更新及所述模块状态更新至 DBT 模块 330。如下详细所述,分配给图形流水线 206 的模块状态数量由 RBM 320 及所述各个模块 340 共同管理。

[0043] DBT 模块 330 透过数据总线 380 及寄存器总线 (控制总线)390 耦接至模块 A 340a、模块 B 340b、模块 C 340c 以及模块 D 340d。数据总线 380 提供输入 (数据) 流至模块 340。模块 A 340a 到模块 D 340d 以串接链方式 (daisy chain manner) 耦接至数据总线 380。也就是说模块 A 340a 在该输入流内的数据上执行进程,接着将该数据发送至模块 B 340b ;模块 B 340b 在该输入流内的该数据上执行进程,接着将该数据发送至模块 C 340c ;模块 C 340c 在该输入流内的该数据上执行进程,接着将该数据发送至模块 D 340d ;以及模块 D 340d 在该输入流内的该数据上执行进程,接着将该数据输出作为图形流水线 206 的输出。模块 D 340d 在一给定全局背景的该数据上执行进程之后,模块 D 340d 发送一脉冲回至 CP 310 以标示与该给定全局背景相关的该全局状态 ID 可被重新分配。除提供一输入 (数据) 流之外,数据总线 380 提供全局背景事件及模块背景事件给模块 340。于一实施例中,数据总线 380 是一百位宽。

[0044] 寄存器总线 390 提供命令给模块 340,所述命令包括绘图命令、全局状态更新以及模块状态更新。一绘图命令根据分配给该绘图命令的该全局状态及该 (所述) 模块状态指示模块 340 在自数据总线 380 接收的该数据上执行绘图。

[0045] 全局状态更新被写进所述各个模块保持的局部寄存器。如图 3 所述,模块 A 340a 写所述全局状态更新至寄存器文件 A 342 内的寄存器 ;模块 B 340b 写所述全局状态更新至寄存器文件 B 352 内的寄存器 ;模块 C 340c 写所述全局状态更新至寄存器文件 C 362 内的寄存器 ;以及模块 D 340d 写所述全局状态更新至寄存器文件 D 372 内的寄存器。

[0046] 模块状态更新仅仅被写至图形流水线 206 内特定模块。例如寄存器总线 390 上的

一模块状态更新可仅为模块 B 340b 准备。于该实施例中，仅模块 B 340b 将会写该模块状态更新至寄存器文件 B 352 内的一寄存器内。所有其他模块将不会写该模块状态更新至它们各自保持的局部寄存器文件内的寄存器。于一实施例中，一模块状态更新包括地址。于该实施例中，每一模块 340 根据该模块状态更新的地址决定其是否应该写该模块状态更新至它的局部寄存器文件。然而，本领域技术人员将明了在不悖离本发明的精神及范畴下，可使用决定特定模块是否应该写一模块状态更新至它的局部寄存器的其他机制。

[0047] 给定图形流水线 206 结构，以下详细描述其操作。

[0048] III. 有效状态管理系统的范例操作

[0049] 如上所述，图形流水线 206 独立管理全局状态更新及模块状态更新，因此，使图形流水线 206 能够根据相对大数量的不同绘图常量执行绘图命令。首先，介绍全局管理状态及模块管理状态的概述。接着，介绍一限制分配该图形流水线 206 的全局状态及模块状态数量的范例方式。最后，介绍一系列范例事件以阐述根据本发明一实施例的有效状态管理系统的操作。

[0050] A. 全局及模块管理状态转变

[0051] CP 310 管理全局状态更新。如果 CP 310 检测到该命令流内的一全局状态更新以及如果图形流水线 206 内有可被写的全局状态寄存器，CP 310 将执行以下功能：(i) 设置 GMD 位 312（如果 GMD 位 312 还未被设置）；(ii) 发送一全局背景事件（即 " Global_Context_Done " ）给 RBM320，RBM 320 在数据总线 380 上提供该全局背景事件；(iii) 递增该全局状态 ID 位；(iv) 写给在 CP 310 内的 GBL_COPY_STATE 寄存器；以及 (v) 发送该全局状态更新给 RBM 320，RBM 320 在寄存器总线 390 上提供该全局状态更新。然而，如果 CP 310 检测到该命令流内的一全局状态更新，但图形流水线 206 内没有可用的全局状态寄存器，前述写给该 GBL_COPY_STATE 寄存器暂缓直到一背景可用。GMD 位 312 被用于标示一背景已为了这组状态被滚动。

[0052] 如果 BMD 位 314 未被设置且 CP 310 检测到该命令流内的模块状态更新，CP310 将设置 BMD 位 314，发送被所述模块状态更新跟随的 Block_Context_Done 事件。由 CP 310 及各个的模块 340 保存的所述已使用位被设置成标示一背景已为了这组状态被滚动。如果没有可用的模块状态寄存器，该目标模块（例如模块 B 340b）若不再接收寄存器更新，则该目标模块（例如模块 B 340b）为不能恢复空闲负责。

[0053] 藉由章节 III.C 所提供的一系列范例事件进一步阐述该全局及模块管理状态转变的操作。

[0054] B. 管理分配给本发明一实施例的图形流水线的全局状态及模块状态的数量。

[0055] 分配给图形流水线 206 的全局状态数量由 CP 310 管理，但是，分配给图形流水线 206 的模块状态数量由 RBM 320 及所述各个模块 340 共同管理。现在详细描述管理全局状态及模块状态数量的机制。

[0056] CP 310 不断地写绘图命令及模块状态更新给 RBM 320。然而，如果所分配的全局状态数量小于图形流水线 206 所承受的全局状态最大数量，CP 310 仅写全局状态更新给 RBM 320。即，分配给图形流水线 206 的全局状态数量由 CP 310 管理。

[0057] RBM 320 不断地写绘图命令及全局状态更新给 DBT 模块 330。然而，如果所分配的模块状态数量小于图形流水线 206 的模块 340 所承受的模块状态的最大数量，RBM 320 仅写

模块状态更新给 DBT 模块 330。DBT 模块 330 根据接收自模块 340 的信号决定模块状态数量是否小于该最大数量。即，RBM 320 及所述各个模块 340 管理分配给图形流水线 206 的模块状态更新数量。

[0058] 如上所述，RBM 320 及模块 340 共同起作用以调节 (throttle) 写进图形流水线 206 的模块状态更新数量。如图 3 所述，模块 340 分别透过迹线 344、354、364 及 374 连接至 RBM 320。只要分配给所述模块的模块状态数量小于每一模块的模块状态的最大数量，RBM 320 写模块状态更新给 DBT 模块 330，DBT 模块 330 依次 (in turn) 写所述模块状态更新至寄存器总线 390。然而如果一个模块 340 的模块更新数量达到那个模块的模块状态最大数量，那个模块不返回“空闲 (free)”信号给 RBM 320。因此，RBM 320 暂缓直到先前发布的绘图命令完成执行以及分配给那个绘图命令的该模块状态（或所述模块状态）被解除分配。

[0059] C. 一系列范例事件

[0060] 图 4 描绘表 400，表 400 阐述图形流水线 206 如何响应于一系列范例事件。如图 4 所示，表 400 包括 8 列。第一列 410 包括行号 (linenumber)，所述行号列举所接收到的该一系列事件。第二列 420 包括 CP 310 所接收的并被写至寄存器总线 390 的范例事件。第三列 430 包括该提供在数据总线 380 上的输入（数据）流。第四列 440 标示 CP 310 的 GMD 位 312 的状态。第五列 450 包括 CP 310 的 BMD 位 314 的状态。第六列 460 包括全局状态 ID，该全局状态 ID 标示图形流水线 206 的全局状态。第七列 470 标示一模块已使用位的状态及模块 A 340a 的模块背景标志位。以及第八列 480 标示模块 A 340a 的一模块状态 ID。第七列 470 及第八列 480 阐述模块 A 340a 如何响应于该一系列范例事件。本领域技术人员将了解模块 B 340b、模块 C 340c、模块 D 340d 以与模块 A 340a 相同的方式运行。现详细描述由第一列 410 内的行号所列举的该一系列事件。

[0061] 参考表 400 第一列 410，如第四列 440 及第五列 450 所分别描述，行 1 阐述 GMD 位 312 及 BMD 位 314 是被上电已使用 (power up dirty)（即逻辑 1）。此外，第七列 470 阐述模块 A 340a 的已使用位也被上电已使用（及逻辑 1）。于行 2 及 3 内，一模块状态及一全局状态被分别接收。所述状态写 (state write) 不能引起任何状态改变。

[0062] 于行 4 内，一 Draw_Initiator_in_Context_(Ctx)_0 命令被接收。如第四列 440、第五列 450 及第七列 470 分别所述，这个绘图命令清除 GMD 位 312、BMD 位 314 以及由模块 A 340a 管理的该已使用位。该绘图命令被执行在数据总线 380 上所提供的数据上。第三列 430 标示这个数据包括顶点 / 像素矢量 (vertex/pixel vectors)。

[0063] 于行 5 内，Global_Context_Done 命令被接收。这个命令标示全局背景 0 已经结束。因此，该全局背景事件 (Ctx Done Event) 被提供在数据总线 380 上以指示图形流水线 206 内的模块 340 该全局背景 0 已经结束。于一实施例内，该背景完成事件是一数据封包。

[0064] 于行 6 内，一写 GBL_COPY_STATE 命令被接收。这个命令标示一个新全局状态将被写进图形流水线 206 的所述寄存器。由于这个命令，如第四列 440 所标示，GMD 位 312 被设置。此外，如第六列 460 所标示，该全局状态 ID 位被递增至 1 以标示图形流水线 206 正进入一新的全局状态。

[0065] 于行 7 内，与全局背景 1 相对应的该全局状态更新被接收且被写进图形流水线 206 的所述寄存器。

[0066] 于行 8 内，Draw_Initiator_in_Ctx_1 命令被接收。这个绘图命令清除先前设置的

唯一的已使用位，即第四列 440 所述的 GMD 位 312。与行 4 所接收的 Draw_Initiator_in_Ctx_0 相似，该 Draw_Initiator_in_Ctx_1 命令被执行在数据总线 380 上携带 (carry) 的顶点 / 像素矢量上。

[0067] 于行 9 内，一 Block_Context_Done 事件被接收。这个命令标示所述模块背景中的一个模块背景已被完成。因此，BMD 位 314 被设置（如第五列 450 所述）且一模块背景事件 (Blk_Ctx_Done_Event) 被提供在数据总线 380 上。如第七列 470 所标示，模块 A 340a 检测这个事件，但并不初始设置它的已使用位。模块 A 340a 不初始设置其已使用位是因为仍然未确定这个模块事件是否与模块 A 340a 相对应。这个模块事件可与图形流水线 206 内的任一模块 340 相对应。

[0068] 于行 10 内，一模块状态 (Block_State)（包括 BlkA 状态）命令被接收，该命令标示这个模块事件与模块 A 340a 相对应。因此，分别如第七列 470 及第八列 480 所标示，模块 A 340a 设置它的已使用位并递增它的状态 ID。如上所述，于一实施例中，该命令流内的所述命令包括地址，且由于该 Block_State 的地址落入与模块 A 340a 相对应的预定地址范围，模块 A 340a 决定该 Block_State 应该被写进寄存器文件 A 342 内的一寄存器。

[0069] 于行 11 内，Global_Context_Done 命令被接收。这个命令标示全局背景 1 已结束。因此，一全局背景事件 (Ctx_Done_Event) 被提供在数据总线 380 上，以指示图形流水线 206 内的模块 340 全局背景 1 已结束。

[0070] 于行 12 内，写 GBL_COPY_STATE 命令被接收，与行 6 所接收的写 GBL_COPY_STATE 命令相似，该接收在行 12 内的命令标示一新的全局状态将被写进图形流水线 206 的所述寄存器。由于这个命令，如第四列 440 所标示，GMD 位 312 被设置。此外，如第六列 460 所标示，该全局状态 ID 位被递增至 2，以标示图形流水线 206 正进入一个新的全局状态。

[0071] 于行 13 内，与全局背景 2 相对应的该全局状态更新被接收且写进图形流水线 206 的所述寄存器。

[0072] 于行 14 内，Draw_Initiator_in_Ctx_2 命令被接收，这个绘图命令清除先前设置的所述已使用位，所述已使用位即分别如第四列 440、第五列 450 及第七列 470 所述的 GMD 位 312、BMD 位 314 以及由模块 A 340a 管理的该已使用位。与行 4 所接收的 Draw_Initiator_in_Ctx_0 及行 8 所接收的 Draw_Initiator_in_Ctx_1 相似，该 Draw_Initiator_in_Ctx_2 命令被执行在提供于数据总线 380 上的顶点 / 像素矢量上。

[0073] 于行 15 内，Block_Context_Done 事件被接收。这个命令标示所述模块背景中的一个模块背景已完成。因此，BMD 位 314 被设置（如第五列 450 所述），且一模块背景事件 (Blk_Ctx_Done_Event) 被写进数据总线 380。如第七列 470 所标示，模块 A 340a 检测这个事件，但并不初始设置它的已使用位。

[0074] 于行 16 内，模块状态 (Block_State)（非 BlkA 状态）命令被接收，该命令标示这个模块事件与模块 A 340a 不对应。因此，分别如第七列 470 及第八列 480 所标示，模块 A 340a 不设置其已使用位，也不递增它的状态 ID。于一实施例中，因为这个 Block_State 的地址没有落入与模块 A 340a 相对应的预定地址范围，模块 A 340a 决定这个 Block_State 不应该被写入寄存器文件 A 342 内的一寄存器。

[0075] 于行 17 内，另一 Draw_Initiator_in_Ctx_2 命令被接收，这个绘图命令清除先前设置的所述已使用位，所述已使用位即分别如第四列 440、第五列 450 及第七列 470 所述的

GMD 位 312、BMD 位 314 以及由模块 A 340a 管理的该已使用位。与先前接收的其他 Draw_Initiator 命令相似，这个 Draw_Initiator_in_Ctx_2 命令被执行在于数据总线 380 上携带的顶点 / 像素矢量上。

[0076] 以与上述相似方式接收及处理附加事件。需注意上面介绍的该一系列范例事件仅作为说明目的，并非以此限制本发明。

[0077] IV. 有效状态管理系统的范例软件应用

[0078] 除了 GPU 110 的硬件应用之外，GPU 110 也可被具体表现在所处理的软件内，例如配置用以存储该软件的计算机可用（即可读）媒介（例如计算机可读程序代码）内。该程序代码促使本发明的实施例实现，包括以下实施例：(i) 本文所揭示的所述系统及技术（例如管理全局状态更新及模块状态更新）的所述功能；(ii) 本文所揭示的所述系统及技术的制造（例如 GPU 110 的制造）；或 (iii) 本文所揭示的所述系统及技术的功能及制造的组合。例如，这可透过使用一般的编程语言（例如 C 或 C++）、包括 Verilog HDL，VHDL，Altera HDL(AHDL) 等的硬件描述语言 (HDL)、或其他可获得的编程及 / 或图表捕捉工具 (schematiccapture tool)（例如电路捕捉工具）来完成。

[0079] 该程序代码可被任何已知计算机可读媒介处理，该任何已知计算机可读媒介包括半导体、磁碟、光碟（例如 CD-ROM，DVD-ROM）并作为具体表现在一计算机可使用（即可读）传播媒介（例如载波或包括数字的 (digital)、光的或以模拟为基础 (analog-based) 的媒介的任何其他媒介）内的一计算机数据信号。就这点而论，该代码可通过通讯网络传播，所述通讯网络包括互连网 (internet)。需了解所述实现的功能及 / 或上述系统及技术所提供的结构可被表现在一核心（例如 GPU 核心）内，该核心被具体表现在程序代码内并可被转变成硬件以作为集成电路产品一部分。

[0080] V. 结论

[0081] 需明白的是该实施方式章节（并非发明说明 (Summary) 及摘要 (Abstract) 章节）意图被使用以解释权利要求。该发明说明及摘要章节可阐明如发明人所周密考虑的本发明的一个或多个但并非全部具体实施例，因此，并非意图以任何方式限制本发明及附加的权利要求。

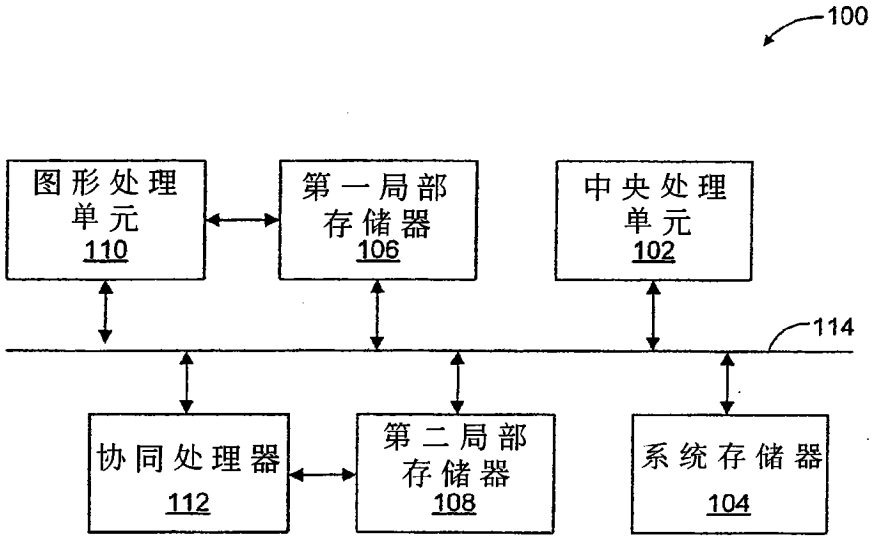

100
图形处理单元 110
第一局部存储器 106
中央处理单元 102
114
协同处理器 112
第二局部存储器 108
系统存储器 104

图 1

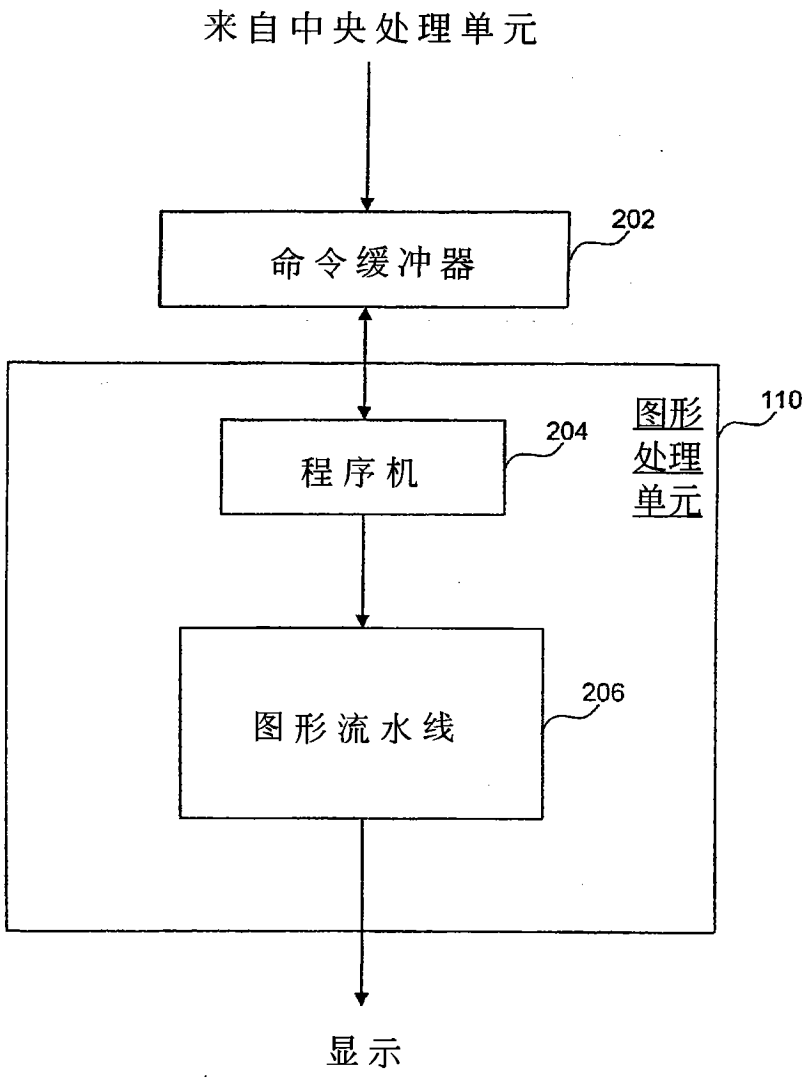


图 2

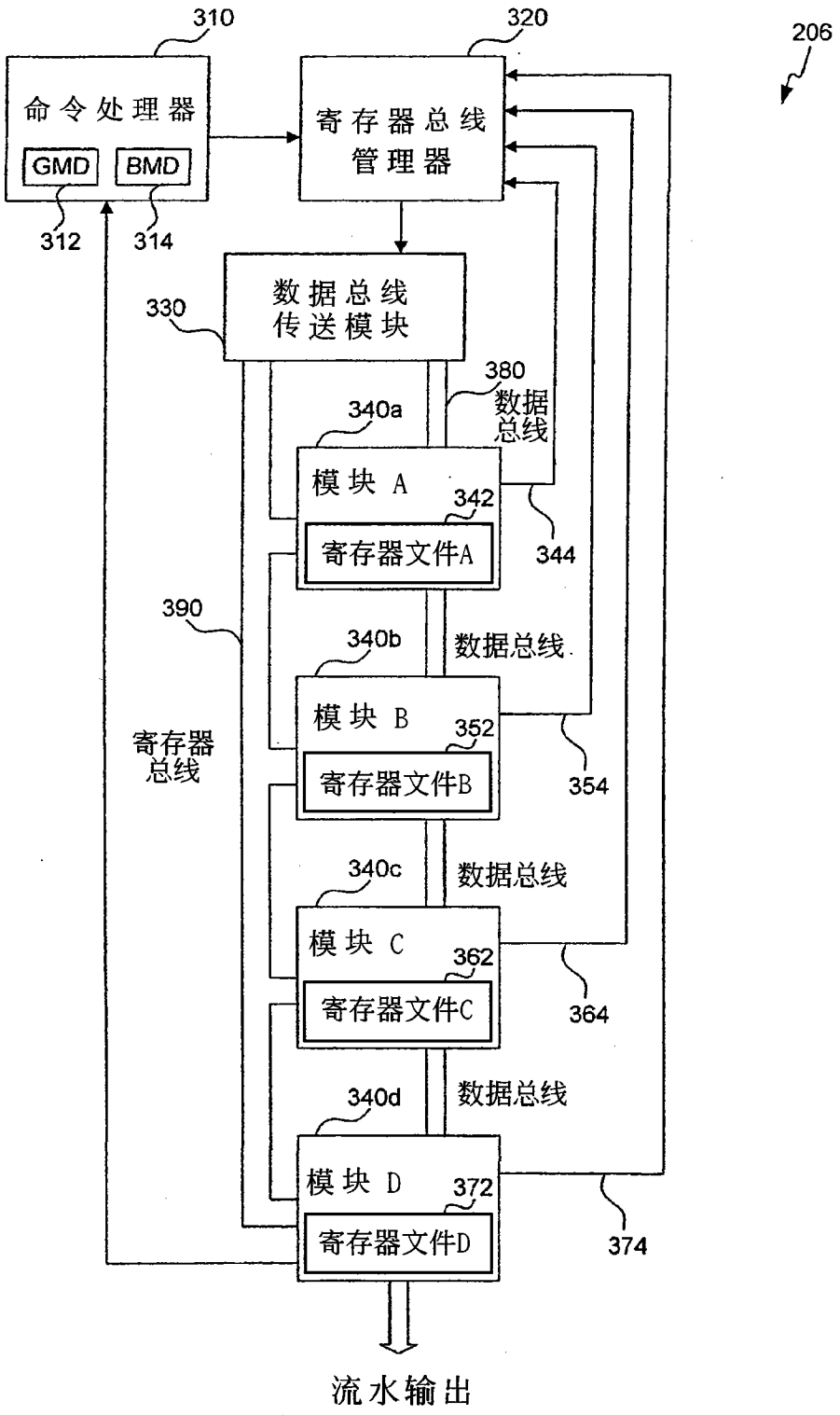
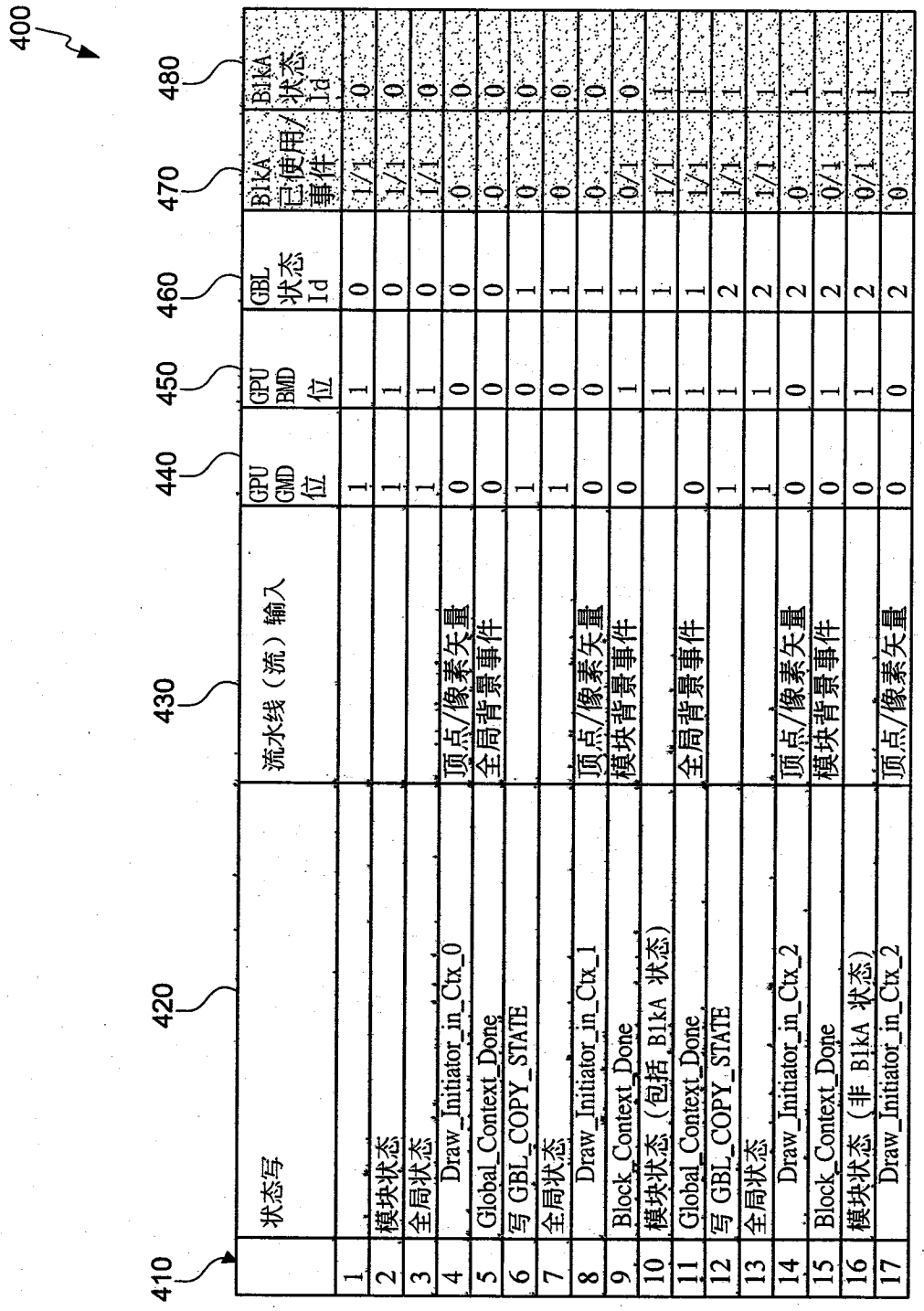

310
320
206
命令处理器
GMD
BMD
寄存器总线管理器
312
314
数据总线传送模块
330
380
数据总线
340a
模块 A
342
寄存器文件A
344
390
340b
数据总线
模块 B
352
寄存器文件B
354
寄存器总线
340c
数据总线
模块 C
362
寄存器文件C
364
340d
数据总线
模块 D
372
寄存器文件D
374
流水输出

图 3



	状态写	流水线（流）输入	GPU GMD 位	GPU BMD 位	GBL 状态 Id	B1kA 已使用/事件	B1kA 状态 Id
1			1	1	0	1/1	0
2	模块状态		1	1	0	1/1	0
3	全局状态		1	1	0	1/1	0
4	Draw_Initiator_in_Ctx_0	顶点/像素矢量	0	0	0	0	0
5	Global_Context_Done	全局背景事件	0	0	0	0	0
6	写 GBL_COPY_STATE		1	0	1	0	0
7	全局状态		1	0	1	0	0
8	Draw_Initiator_in_Ctx_1	顶点/像素矢量	0	0	1	0	0
9	Block_Context_Done	模块背景事件	0	1	1	0/1	0
10	模块状态（包括 B1kA 状态）			1	1	1/1	1
11	Global_Context_Done	全局背景事件	0	1	1	1/1	1
12	写 GBL_COPY_STATE		1	1	2	1/1	1
13	全局状态		1	1	2	1/1	1
14	Draw_Initiator_in_Ctx_2	顶点/像素矢量	0	0	2	0	1
15	Block_Context_Done	模块背景事件	0	1	2	0/1	1
16	模块状态（非 B1kA 状态）		0	1	2	0/1	1
17	Draw_Initiator_in_Ctx_2	顶点/像素矢量	0	0	2	0	1

图 4