(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 101080698 B
(45)授权公告日 2010.08.11

(21)申请号 200580043375.5

(22)申请日 2005.12.14

(30)优先权数据
11/019,414 2004.12.20 US

(85)PCT申请进入国家阶段日
2007.06.18

(86)PCT申请的申请数据
PCT/US2005/045834 2005.12.14

(87)PCT申请的公布数据
WO2006/068985 EN 2006.06.29

(73)专利权人 辉达公司
地址 美国加利福尼亚州

(72)发明人 邓肯·A·里亚赫 约翰·M·丹斯金
乔纳·M·阿尔本
迈克尔·A·奥格林茨
安东尼·迈克尔·塔马西

(74)专利代理机构 北京市磐华律师事务所 11336
代理人 董巍 顾珊

(51)Int.Cl.
G06F 9/46(2006.01)
G06F 15/80(2006.01)
G06F 15/00(2006.01)
G06T 1/60(2006.01)
G06F 15/16(2006.01)
G09G 5/399(2006.01)

(56)对比文件
US 2003/0006949 A1, 说明书第[0010]段.
US 6731288 B2, 2004.05.04, 全文.
US 6778188 B2, 2004.08.17, 说明书第12栏第1-12行.
US 5546530 A, 1996.08.13, 说明书第2栏第58-60行,第4栏第59行-第5栏第5行,第6栏第16-24行.
CN 1549964 A, 2004.11.24, 说明书第2页第6-24行,第3页倒数第4行-第4页第1行,第22页第8行-第27页第16行.
US 6243107 B1, 2001.06.05, 说明书第1栏第19-26行,第2栏第5-19行,第3栏第8行-第5栏第18行,图1.
US 2003/0084181 A1, 2003.05.01, 全文.
US 6452595 B1, 2002.09.17, 说明书第36栏第15-19行.

审查员 杨薇

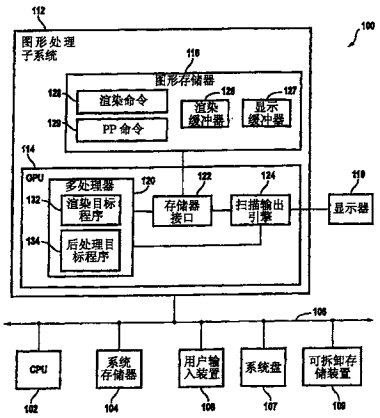
权利要求书 3 页 说明书 23 页 附图 10 页

(54)发明名称

图形处理器,图形处理系统及产生图像的方法

(57)摘要

在图形处理器中,渲染目标程序与后处理目标程序共享对具有可编程执行核心的主机处理器的存取。所述渲染目标程序根据几何数据产生图像的片段数据。所述后处理目标程序运行以根据所述片段数据产生像素数据帧并将所述像素数据存储于帧缓冲器中。与所述主机处理器的操作并行地,扫描输出引擎读取先前所产生的帧的像素数据并将所述像素数据提供至显示装置。所述扫描输出引擎周期性地触发所述主机处理器,以操作所述后处理目标程序来产生下一个帧。可对所述扫描输出引擎与所述后处理目标程序之间的定时进行控制,以便当所述扫描输出引擎结束对当前帧的读取时,所要显示的下一个帧在帧缓冲器中准备就绪。



CN 101080698 B

1. 一种图形处理器，其包括：

多处理器，其经配置以执行多个目标程序，包括：

渲染目标程序，其经配置以执行第一程序指令序列以用于产生图像数据且将所述图像数据写入渲染缓冲器；及

后处理目标程序，其经配置以：

从所述渲染缓冲器中读取所述图像数据；

执行第二程序指令序列以用于从所述图像数据中产生像素数据帧；及

将所述像素数据帧写入显示缓冲器；及

扫描输出引擎，其经配置以将所产生的像素数据帧等时地传输至显示端口，并经耦接以周期性地将触发器信号传输至所述多处理器，

其中所述多处理器进一步经配置以操作所述后处理目标程序，以响应于所述触发器信号而产生新的像素数据帧。

2. 如权利要求 1 所述的图形处理器，其中所述多处理器包括：

第一可编程处理引擎，其经配置以执行所述渲染目标程序；及

第二可编程处理引擎，其经配置以执行所述后处理目标程序。

3. 如权利要求 1 所述的图形处理器，其中所述多处理器包括：

可切换上下文的处理核心；

上下文管理器，其经配置以保持对应于所述多个目标程序中每一者的相应上下文；及

仲裁单元，其经配置以控制从由所述上下文管理器保持的所述上下文中选择当前上下文，以便响应于所述触发器信号而激活对应于所述后处理目标程序的上下文。

4. 根据权利要求 1 所述的图形处理器，其中所述多处理器包括仲裁单元，所述仲裁单元经配置以控制所述多个目标程序的执行，及

所述扫描输出引擎经配置以将所产生的像素数据帧等时地传输至显示端口，并经耦接以周期性地将触发器信号传输至所述仲裁单元；

其中所述仲裁单元进一步经配置以操作所述后处理目标程序，以响应于所述触发器信号而产生新的像素数据帧。

5. 一种图形处理器，其包括：

可编程执行核心，其经配置以可在多个上下文之间进行切换，且进一步经配置以执行与所述多个上下文中被切换到的一者相关联的程序指令，

仲裁单元，其耦接至所述可编程执行核心并经配置以控制所述可编程执行核心在所述多个上下文中不同上下文之间的切换，及

扫描输出引擎，其经配置以将所产生的像素数据帧等时地传输至显示端口，并经耦接以将触发器信号传输至所述仲裁单元，

其中所述多个上下文包括：

渲染上下文，其具有与其相关联的第一程序指令序列以用于产生图像数据；及

后处理上下文，其具有与其相关联的第二程序指令序列以用于从所述图像数据中产生像素数据帧；及

其中所述仲裁单元进一步经配置以响应于所述触发器信号而将所述可编程执行核心切换至所述后处理上下文。

6. 如权利要求 5 所述的图形处理器，其中所述扫描输出引擎进一步经配置以每一帧传输所述触发器信号一次。

7. 如权利要求 6 所述的图形处理器，其中所述扫描输出引擎进一步经配置以使所述触发器信号具有与帧结尾事件固定的时间关系。

8. 如权利要求 5 所述的图形处理器，其中所述仲裁单元进一步经配置以将所述可编程执行核心维持在所述后处理上下文中，直至在所述第二程序指令序列中检测到帧完成事件为止，且此后将所述可编程执行核心切换至所述渲染上下文。

9. 如权利要求 8 所述的图形处理器，其中所述帧完成事件对应于在所述第二程序指令序列中出现瞄准所述第二程序指令序列的起始点的无条件跳转指令。

10. 如权利要求 5 所述的图形处理器，其中所述第一程序指令序列进一步包括用于选择多个图像缓冲器中的一者并将所述图像数据写入至所述所选图像缓冲器的指令。

11. 如权利要求 10 所述的图形处理器，其中所述第二程序指令序列进一步包括用于从所述多个图像缓冲器中的一者或一者以上读取所述图像数据的指令。

12. 如权利要求 11 所述的图形处理器，其中 ：

所述第一程序指令序列进一步包括用于将所述所选图像缓冲器的索引值写入至索引位置内的指令，其中用于写入所述索引值的所述指令是在执行用于将所述图像数据写入至所述所选图像缓冲器的所述指令之后执行 ；及

所述第二程序指令序列进一步包括用于从所述索引位置读取所述索引值的指令。

13. 如权利要求 12 所述的图形处理器，其中所述第二程序指令序列进一步包括以从所述索引位置读取的所述索引值是否不同于先前从所述索引位置读取的旧索引值为条件的至少一个指令。

14. 如权利要求 13 所述的图形处理器，其中所述至少一个指令包括用于释放所述多个图像缓冲器中的一者以供所述渲染上下文随后使用的指令。

15. 如权利要求 5 所述的图形处理器，其中所述第二程序指令序列包括用于选择多个帧缓冲器中的一者并将帧的像素数据写入至所述所选帧缓冲器的指令。

16. 如权利要求 15 所述的图形处理器，其中 ：

所述扫描输出引擎进一步经配置以从第一位置读取显示索引值，所述显示索引值标识所述多个帧缓冲器中的一者，并从所述所标识的帧缓冲器读取帧的像素数据 ；及

所述第二程序指令序列进一步包括用于在完成将所述帧的像素数据写入至所述所选帧缓冲器之后将对应于所述所选帧缓冲器的所述显示索引值写入至所述第一位置的指令。

17. 根据权利要求 5 所述的图形处理器，其中采用所述可编程执行核心的同一电路来执行所述渲染上下文和所述后处理上下文。

18. 一种图形处理系统，其包括 ：

多个图像缓冲器，其每一者经配置以存储图像的片段数据 ；

多个帧缓冲器，其每一者经配置以存储帧的像素数据 ；

多处理器，其包括 ：

可编程执行核心，其经配置以可在多个上下文之间进行切换，以使所述可编程执行核心执行与所述多个上下文中被切换到的一者相关联的程序指令 ；及

仲裁单元，其耦接至所述可编程执行核心，并经配置以控制所述可编程执行核心在所

述多个上下文中的不同上下文之间的切换，及

扫描输出引擎，其经配置以将像素数据帧从所述帧缓冲器等时地传输至显示端口，并经耦接以将触发器信号周期性地传输至所述多处理器的所述仲裁单元，其中所述多个上下文包括：

渲染上下文，其具有与其相关联的第一程序指令序列，以用于产生多个图像的片段数据并将每一图像的所述片段数据写入至所述图像缓冲器中的一者；及

后处理上下文，其具有与其相关联的第二程序指令序列，以用于从所述图像缓冲器中的所述片段数据中产生像素数据帧并将所述帧的所述像素数据写入至所述帧缓冲器中的一者；及

其中所述仲裁单元进一步经配置以响应于所述触发器信号而将所述可编程执行核心切换至所述后处理上下文。

19. 一种用于产生图像的方法，所述方法包括：

在处理器的共享执行核心中操作渲染目标程序，所述渲染目标程序响应于第一程序指令序列而产生图像的片段数据；

与操作所述渲染目标程序并行地操作扫描输出引擎，以将像素数据帧等时地递送至显示装置，其中所述扫描输出引擎周期性地产生触发器信号；及

响应于所述触发器信号，在所述处理器的共享执行核心中操作后处理目标程序，所述后处理目标程序响应于第二程序指令序列而从一个或一个以上图像的所述片段数据中产生新的像素数据帧，所述新的像素数据帧可供所述扫描输出引擎使用。

20. 如权利要求 19 所述的方法，其中所述触发器信号具有与帧结尾事件固定的时间关系。

21. 如权利要求 19 所述的方法，其中在所述后处理目标程序的操作期间中止所述渲染目标程序的操作，并在所述后处理目标程序已完成所述新的像素数据帧的产生之后重新开始所述渲染目标程序的操作。

22. 如权利要求 19 所述的方法，其中所述第二程序指令序列包括用于对图像的所述片段数据进行下降过滤的指令。

23. 如权利要求 19 所述的方法，其中所述第二程序指令序列包括用于对图像的所述片段数据进行上升过滤的指令。

24. 如权利要求 19 所述的方法，其中所述第二程序指令序列包括用于为所述帧的每一像素计算 LCD 过激励值的指令。

25. 如权利要求 19 所述的方法，其中所述第二程序指令序列包括用于使用两个或两个以上不同图像的片段数据来形成合成图像的指令。

26. 如权利要求 19 所述的方法，其中所述第二程序指令序列包括用于将所述图像映射至表面上的指令。

图形处理器，图形处理系统及产生图像的方法

技术领域

[0001] 本发明大体而言涉及图形处理器，且具体而言，涉及使用可编程硬件进行的实时显示后处理。

背景技术

[0002] 基于计算机的图像渲染通常首先对景物进行几何表示。将各种物体描述为可置于景物中的“图元”（通常是例如三角形等简单的多边形，以及点及线）的集合。选择一取景座标系，并将这些图元变换至该座标系中。然后，将图元转换成二维 (2-D) “片段”阵列表示形式，其中每一片段均具有颜色并可具有其他属性，例如深度座标或曲面法线。可在图元及 / 或片段级上引入照明、纹理、灰雾及各种其他会增强视觉真实性的效果。在渲染过程结束时，将每一片段的数据（一般至少是颜色值）存储于图像缓冲器中。所述图像缓冲器是通过一“扫描输出”过程读出，所述“扫描输出”过程等时地运行，以便以规定的屏幕刷新速率将像素递送至显示装置。实时动画要求渲染过程以约 30Hz 的速率递送新图像。典型的显示装置以约 60-80Hz 的屏幕刷新速率运行。

[0003] 为满足这些处理速率要求，许多计算机系统包括专用图形协处理器，其对由中央处理器 (CPU) 提供的数据实施渲染操作且还执行同步扫描输出操作来驱动显示装置。典型的图形处理器包括彼此不同步运行的渲染目标与扫描输出引擎。渲染目标程序为“后面”图像缓冲器中的新图像产生片段数据，而扫描输出引擎则使用“前面”图像缓冲器中先前经过渲染的图像来驱动显示器。当完成对新图像的渲染时，切换“后面”与“前面”缓冲器，以便扫描输出引擎开始显示新渲染的图像，而渲染目标程序则向前移至下一图像。一般而言，在对下一图像的渲染完成之前，扫描输出引擎可对同一图像读取两次或三次。

[0004] 渲染目标程序与扫描输出引擎在实施方案方面通常差别很大。渲染目标程序一般是灵活性的及可编程的。典型的渲染目标程序包括具有功能单元的执行核心（或若干个并行的执行核心），可指令所述功能单元执行任意的操作序列。通过适当编程，可使执行核心执行各种渲染算法的任意组合来产生特定图像，并可根据需要改变这些算法。

[0005] 相比之下，扫描输出引擎通常具有有限的处理能力且不可编程。而是，扫描输出引擎具有一序列管线式专用处理电路，片段数据即流经这些管线式专用处理电路，其中这些处理电路实施各种操作来将片段数据变换成像素值。举例来说，某些扫描输出引擎支持：加法覆盖（例如光标或视频覆盖），其可以高于所渲染图像的速率进行更新；颜色修正（例如 γ 修正，以虑及显示响应中的非线性）；或者对片段数据的过滤，以与屏幕上的像素数量相匹配（例如以实现图形保真）。所述专用电路一般设计成以固定延迟工作，以确保等时地将像素数据递送至显示装置。

[0006] 在某些处理器中，可启用或禁用各种扫描输出一时间操作（例如可将覆盖接通或断开）或改变操作的参数（例如 γ 修正的参数或覆盖的位置）。但是，由于每一操作均是在不同的专用电路中实施，因而一般不可能在不建构不同扫描输出引擎的情况下在管线中添加新的操作、改变操作顺序、或改变用于实施特定操作的算法。因此，扫描输出引擎的重

新配置能力非常有限。添加新的特征一般需要对电路进行改动，而此可能会影响芯片面积及排程、最大成本及延迟。

[0007] 随着实时渲染技术的持续进步，越来越需要一种可在显示速率下添加各种效果的、功能更强大且更灵活的扫描输出引擎。此外，图形处理器可驱动的显示装置的范围也已最大；除传统的CRT监视器之外，图形处理装置还可用于驱动LCD监视器、数字微镜投影仪、等离子体监视器、等等。每一类型的显示装置均对其像素的驱动具有不同的要求，且很难在单个硬件管线中适应所有这些要求。因此，非常希望在片段－像素转换过程中具有更大的灵活性。

[0008] 因此，希望使图形处理器具有以显示步调来执行任意操作序列的能力。

发明内容

[0009] 本发明的实施例提供图形处理器，在所述图形处理器中，渲染目标程序与后处理目标程序通过可编程的执行核心来共享对主机处理器的存取。所述渲染目标程序运行以根据几何数据来产生图像的片段数据并将所述片段数据写入至图像缓冲器。所述后处理目标程序运行以根据一个或多个填满的图像缓冲器中的所述片段数据来产生像素数据帧，并将所述像素数据写入至帧缓冲器中。与主机处理器的操作并行地，扫描输出引擎从帧缓冲器中读取先前所产生帧的像素数据，并将所述像素数据提供至显示装置。所述扫描输出引擎周期性地触发所述主机处理器来操作所述后处理目标程序产生下一个帧。较佳对所述扫描输出引擎与所述后处理目标程序之间的定时进行控制，以在所述扫描输出引擎结束对当前帧的读取时，要显示的下一个帧已在帧缓冲器中准备就绪。

[0010] 根据本发明的一个方面，一种图形处理器包括可编程执行核心、仲裁单元、及扫描输出引擎。所述可编程执行核心配置成可在若干上下文之间进行切换，并进一步经配置以执行与被切换到的上下文相关联的执行程序指令。所述仲裁单元耦接至所述执行核心，且经配置以控制所述执行核心在不同上下文之间的切换。所述扫描输出引擎经配置以将所产生的像素数据帧等时地传输至显示端口，并经耦接以周期性地将触发信号传输至所述仲裁单元。所述上下文包括：渲染上下文，其具有用于产生图像数据的第一程序指令序列与其相关联；及后处理上下文，其具有用于根据所述图像数据来产生像素数据帧的程序指令与其相关联。所述仲裁单元进一步经配置以响应于所述触发信号而将所述执行核心切换至所述后处理上下文。在某些实施例中，所述扫描输出引擎进一步配置成使所述触发信号相对于帧事件的结束具有实质固定的时间关系。

[0011] 在某些实施例中，所述仲裁单元进一步经配置以使所述执行核心保持于所述后处理上下文中，直至在所述第二程序指令流中检测到帧终止事件为止，并在此后将所述执行核心切换至所述渲染上下文。所述帧结束事件可例如对应于在所述第二程序指令流中出现一指向所述第二程序指令流的起点的无条件跳转指令。

[0012] 根据本发明的另一方面，提供一种用于产生图像的方法。在处理器的共享执行核心中，操作渲染目标程序；所述渲染目标程序响应于第一程序指令序列而产生图像的片段数据。与操作所述渲染目标程序并行地，操作扫描输出引擎以将像素数据帧等时地递送至显示装置；所述扫描输出引擎周期性地产生触发信号。响应于所述触发信号，在所述处理器的共享执行核心中操作后处理目标程序；所述后处理目标程序响应于第二程序指令序列而

根据一个或多个图像的片段数据来产生新的像素数据帧，且可将所述新的像素数据帧提供至所述扫描输出引擎。在某些实施例中，所述第二程序指令序列可例如包括：用于对图像的所述片段数据进行下降过滤的指令；用于对图像的所述片段数据进行升高过滤的指令；用于为所述帧中每一像素计算一 LCD 过激励值的指令；及 / 或用于使用两个或更多个不同图像的片段数据来形成合成图像的指令。

[0013] 根据本发明的再一方面，一种图形处理系统包括图像缓冲器、帧缓冲器及多处理器。每一图像缓冲器经配置以存储图像的片段数据，且每一帧缓冲器经配置以存储帧的像素数据。所述多处理器包括可编程执行核心、仲裁单元、及扫描输出引擎。所述可编程执行核心配置成可在若干上下文之间进行切换，以便所述执行核心执行与被切换到的上下文相关联的程序指令。所述仲裁单元耦接至所述执行核心，且经配置以控制所述执行核心在不同上下文之间的切换。所述扫描输出引擎经配置以将像素数据帧等时地从所述帧缓冲器传输至显示端口，并经耦接以周期性地将触发信号传输至所述多处理器的仲裁单元。所述上下文包括：渲染上下文，其具有用于产生图像的片段数据及将每一图像的所述片段数据写入至所述图像缓冲器之一的第一程序指令序列与其相关联；及后处理上下文，其具有用于根据所述图像缓冲器中的帧数据来产生像素数据帧并将所述帧的像素数据写入至所述帧缓冲器之一的第二程序指令序列与其相关联。所述仲裁单元进一步经配置以响应于所述触发信号而将所述执行核心切换至所述后处理上下文。

[0014] 下文具体实施方式部分与附图一起将使人们能够更好地了解本发明的性质及优点。

附图说明

[0015] 图 1 是根据本发明一实施例的计算机系统的高阶方块图；

[0016] 图 2 是根据本发明一实施例的图形处理子系统的数据流图；

[0017] 图 3A 图解说明根据本发明一实施例的渲染命令流；

[0018] 图 3B 图解说明根据本发明一实施例的后处理程序；

[0019] 图 4 是一方块图，其显示根据本发明一实施例的多处理器的各组件；

[0020] 图 5 是根据本发明一实施例的仲裁逻辑过程的流程图；

[0021] 图 6 是一方块图，其显示根据本发明另一实施例的多处理器的各组件；

[0022] 图 7 图解说明根据本发明一实施例的渲染命令流，其具有用于写入后处理程序的命令；

[0023] 图 8 是根据本发明一实施例的支持两个扫描输出引擎的多处理器的方块图；

[0024] 图 9 图解说明一合成图像；

[0025] 图 10 是根据本发明一实施例的支持合成图像的多处理器的方块图；

[0026] 图 11 图解说明根据本发明一实施例用于形成合成图像的后处理程序；

[0027] 图 12 是根据本发明一实施例用于产生 LCD 过激励值的过程的流程图；及

[0028] 图 13 是根据本发明一实施例的用于旋转图像的过程的流程图。

具体实施方式

[0029] 概述及术语

[0030] 本发明的实施例提供图形处理器，在所述图形处理器中，渲染目标程序与后处理目标程序通过可编程的执行核心来共享对主机处理器的存取。以渲染步调运行的所述渲染目标程序根据几何数据来产生图像的片段数据，并将所述片段数据写入至图像缓冲器。以扫描输出（显示）步调运行的后处理目标程序根据在一个或多个填满的图像缓冲器中的片段数据来产生像素数据，并将所述像素数据写入至帧缓冲器中。并行地，扫描输出引擎从不同的帧缓冲器中读取先前所产生的帧的像素数据，并将所述像素提供至显示装置。与所述扫描输出作业同步地，所述扫描输出引擎触发所述主机处理器以扫描输出步调执行所述后处理目标程序，以便当所述扫描输出引擎结束对当前帧的读取时，下一个帧将准备就绪。

[0031] 在本文中使用以下术语：

[0032] 术语“图像”是指构成某个图片的表示形式的“片段”阵列，其中每一片段均具有颜色并可具有其他属性，例如深度、透明度等等。片段颜色可表示为任意方便的格式，包括红色 / 绿色 / 蓝色 (RGB)、亮度 - 色度 (YCrCb)、色调 / 光亮度 / 饱和 (HLS)、单色强度值、或其他格式，并具有任何所需程度的分辨率。所述片段并不需要在数量或布置上相对于显示装置具有任何特定关系，且所述片段阵列在各片段之间既可具有也可不具有均匀的间距。

[0033] “渲染”大体是指根据景物数据（其通常包括图元及相关几何数据）产生图像或片段数据的过程。渲染过程可包括许多级，例如顶点处理（变换至屏幕空间）、设置（在屏幕空间中选择所关心的图元及属性）、光栅化（对片段位置阵列进行取样，以确定哪一（哪些）图元覆盖每一片段）、着色（根据哪一（哪些）图元覆盖一片段而产生该片段的属性）、及着色后光栅操作（例如通过降低过滤来变换片段阵列）。一般以平均速度（“渲染步调”）对图像进行渲染，在某些实施例中，所述平均速度是适用于动画的速度，例如每秒约 30 个图像。渲染过程的细节对于本发明而言并不重要，且渲染速度可在图像之间出现波动。

[0034] 经渲染的图像显示于一使用“像素”阵列或光栅的显示装置上，其中响应于作为像素值的函数的驱动信号，每一像素均显示一颜色。如上文所述，显示装置的像素既可对应于也可不对应于经渲染图像的单个片段。本文中所述的“帧”大体是指像素光栅的完整的一组像素值。新的帧是以规定的屏幕刷新速率或“显示步调”（例如 80Hz）进行显示，所述规定的屏幕刷新速率或“显示步调”可以是用户可选的参数。各连续的帧可显示同一图像或不同图像。

[0035] “后处理”大体是指处理一个或多个图像的片段数据，以产生像素数据帧。后处理较佳以显示步调（屏幕刷新速率）进行，显示步调一般高于动画速度。根据本发明可实施许多后处理操作。下文是许多实例；应了解，本发明并非仅限于后处理操作的任何特定组合或序列。

[0036] 一个后处理操作是过滤，其涉及到对来自一个图像的不同片段的值进行混合，以获得单个像素值。过滤可使像素数量不同于片段数量。在一个实施例中，可对片段进行下降过滤，以产生少于片段数量的像素数量。举例而言，每一像素值可为邻近片段值的加权平均值。下降过滤可用于减小图像大小、构建图形保真作为过取样（例如多重取样）操作的最后一级等等。另一选择为，可对各片段进行上升过滤，以产生大于片段数量的像素数量。例如，可使用内插在输入片段值之间添加额外像素值或者添加额外像素值来取代输入片段值。可使用上升过滤例如来放大要显示的图像。

[0037] 另一后处理操作是合成，其大体是指将来自两个或更多个图像的片段组合成最终

图像。例如，可将光标图像覆盖于屏幕图像的一部分上，或者可将视频图像（例如电影）覆盖于桌面上的窗口中。在某些实施例中，所组合的图像可按不同的速率进行更新，从而使得除以显示步调外，难以正确地组合这些图像。在某些实施例中，合成也可包括对图像进行混合以产生例如淡入、淡出或渐隐等效果。为得到这些效果，可使用随时间变化的相应权重来混合旧的与新的图像。

[0038] 再一后处理操作是表面映射，其大体是指将图像的片段数据映射至 2-D 或 3-D 表面上并将该表面投影至像素阵列上。在一个实施例中，可将片段数据映射至梯形表面上，以为作为投影仪的显示装置提供“梯形失真”校正；所述梯形表面的形状及尺寸可由用户进行调整，以补偿当来自投影仪的光束轴线不垂直于屏幕或其他显示表面时所出现的众所周知的失真。在其他实施例中，也可将片段数据映射至任意形状的表面上，可对所述任意形状的表面进行界定以便产生所需的视觉效果或者以便支持以减小的失真投影至非平整表面（例如圆柱体或穹顶）上。

[0039] 另一后处理操作是表面旋转，其是指将图像的片段数据映射至已围绕垂直于图像的轴线旋转过某一角度（例如 90° 的倍数）的 2-D 表面上。例如，顺时针旋转 90° 将使图像的左上角中的片段出现于像素阵列的右上角处，使图像中的水平线变成像素阵列中的垂直线，等等。可将表面旋转视为上文所述表面映射操作的一特例。在一个实施例中，对以可旋转方式安装的显示装置使用表面旋转，对于图形输入板 PC 显示器及某些平板监视器即为如此。

[0040] 再一后处理操作是发光度补偿。例如，当将图像投影至屏幕上时，距投影仪较远的像素往往比较近的像素更暗。或者，如果屏幕周围的环境光不均匀，则屏幕的一部分可能看起来比其余部分更亮。发光度补偿可通过随屏幕位置补偿像素亮度来修正这些效应，且用户能够根据需要调整各种发光度补偿参数。

[0041] 其他后处理操作对特定显示装置的性质进行补偿。例如，常常使用 γ 修正使像素值以指数方式按比例换算，以便补偿由模拟信号所驱动的显示装置中的非线性电压响应。作为另一实例，在 LCD 监视器中，可通过为每一像素提供一驱动值来改善响应时间，所述驱动值部分地取决于所需亮度及部分地取决于所述亮度相对于前一帧的改变量。在后处理中也可实施专用于其他类型显示装置的其他补偿操作。

[0042] 在某些实施例中，后处理操作是为得到正确结果而应以显示步调进行的操作，例如 LCD 过激励或合成。当渲染与扫描输出不同步时，常常很难或不可能在渲染时确定渲染步调将如何与显示步调相配合，且因此如果要在渲染过程中进行所述操作，也将很难或不可能确保结果正确。

[0043] “扫描输出”大体是指将帧（在后处理之后）的像素数据传送至一连接至显示装置的输出路径。像后处理一样，扫描输出是以显示步调进行。在某些实施例中，通过后处理所产生的像素数据早已呈适用于驱动显示装置的格式。在其他实施例中，扫描输出可包括将像素数据变换成适当格式，并可包括重新按比例缩放、数字 - 模拟转换及其他信号处理技术。

[0044] 系统概述

[0045] 图 1 是根据本发明一实施例的计算机系统 100 的方块图。计算机系统 100 包括通过总线 106 进行通信的中央处理器 (CPU)102 及系统存储器 104。从耦接至总线 106 的一个

或多个用户输入装置 108（例如键盘、鼠标）接收用户输入。在基于像素的显示装置 110（例如传统的基于 CRT 或 LCD 的监视器）上提供视觉输出，所述显示装置 110 在耦接至系统总线 106 的图形处理子系统 112 的控制下运行。系统盘 107 及其他组件 - 例如一个或多个可拆卸存储装置 129（例如软盘驱动器、光盘 (CD) 驱动器、及 / 或 DVD 驱动器）- 也可耦接至系统总线 106。系统总线 106 可使用各种总线协议中的一种或多种来构建，这些总线协议包括 PCI（外围组件互连）、AGP（高级图形处理）及 / 或 PCI-Express(PCI-E)；可提供例如北桥及南桥（未显示）等恰当的“桥式”芯片来互连各种组件及 / 或总线。

[0046] 图形处理子系统 112 包括图形处理单元 (GPU)114 及图形存储器 116- 其可例如使用一个或多个集成电路装置（例如可编程处理器、应用专用集成电路 (ASIC) 及存储装置）来构建。GPU 114 包括多处理器 120、存储器接口模块 122、及扫描输出模块 124。

[0047] 多处理器 120 可经配置以为图形处理子系统 120 实施渲染及后处理任务。多处理器 120 包括可编程执行核心（在图 1 中未明确显示），并能够同时执行两个或更多个过程。一个过程是“渲染目标程序”132，其较佳经配置以根据由在 CPU 102 上执行的各种程序所提供的 2-D 或 3-D 景物数据来产生图像（片段）数据。另一过程是“后处理目标程序”134，其经配置以将片段数据变换成准备在显示装置 110 上显示的像素数据。在一个实施例中，在任一既定时刻，在多处理器 120 的这些过程中只有一个过程现用；可切换入及切换出不同的过程，以提供同时处理的效果。在某些实施例中，渲染目标程序 132 与后处理目标程序 134 可构建于同一实体处理引擎中，所述实体处理引擎较佳能够进行上下文切换以同时支持这两个目标程序；在其他实施例中，则将渲染目标程序 132 与后处理目标程序 134 分别构建于单独的处理引擎中，且不需要在其间进行上下文切换。在下文中将进一步说明多处理器 120 的特定实施例。

[0048] 与多处理器 120 及扫描输出引擎 124 进行通信的存储器接口模块 122 管理与图形存储器 116 进行的所有交互作用。存储器接口模块 122 还可包括用于将从系统总线 106 接收的片段或像素数据写入至图形存储器 116 中的恰当位置而不由多处理器 120 处理的路径。可根据需要改变存储器接口模块 122 的特定配置，且由于其对于理解本发明而言无关紧要，因而不再对其加以赘述。

[0049] 图形处理器 116 可使用一个或多个一般传统设计的集成电路存储装置来构建，其可包含各种物理或逻辑分区，例如渲染缓冲器 126、显示缓冲器 127、渲染命令缓冲器 128 及后处理 (PP) 命令缓冲器 129。渲染缓冲器 126 存储由渲染目标程序 132 或由在 CPU 102 上执行的各种过程所产生的一个或多个图像的片段数据。渲染缓冲器 126 较佳缓冲多个不同的图像，包括由同一过程所产生的图像序列，以便当正对第一图像的片段数据进行后处理时，可将后一图像的片段数据写入至渲染缓冲器 126 中的不同区域。

[0050] 显示缓冲器 127 存储由后处理目标程序 134 根据渲染缓冲器 126 中的片段数据所产生的像素数据。显示缓冲器 127 较佳存储至少两个完整的像素数据帧，以便在正将下一个帧写入至“后面”缓冲器时，可从“前面”缓冲器中扫描输出一个帧的像素数据。

[0051] 渲染命令缓冲器 128 及 PP 命令缓冲器 129 用于对经由系统总线 106 接收到的命令进行排队，以供由多处理器 120 执行。渲染命令缓冲器 128 中的命令将由渲染目标程序 132 执行，而 PP 命令缓冲器 129 中的命令将由后处理目标程序 134 执行，如在下文中所述。

[0052] 图形存储器 116 的其他部分可用于存储 GPU 114 所需的数据（例如纹理数据、颜

色查找表等等）、GPU 114 的可执行程序码等等。

[0053] 可与多处理器 120 集成于单个芯片上或构建于单独芯片上的扫描输出引擎 124 从像素缓冲器 127 中读取像素数据并将所述数据传送至显示装置 110 进行显示。在一个实施例中，扫描输出引擎 124 等时地运行，无论在 GPU 114 中或系统 100 中的其他位置中可进行的任何其他活动如何，均以规定的刷新速率（例如 80Hz）来扫描输出像素数据帧。在某些实施例中，所述规定的刷新速率可以是用户可选择的参数，且可根据显示格式来改变扫描输出次序（例如交错扫描或渐进扫描）。扫描输出引擎 124 可包括用于数据格式化、数字－模拟转换的电路、及用于将像素数据转换成适用于显示装置的格式的其他信号处理电路。可根据需要改变扫描输出引擎 124 的特定配置。

[0054] 在系统 100 的工作过程中，CPU 102 执行各种程序，例如操作系统程序、应用程序、及图形处理子系统 112 的一个或多个驱动程序。所述驱动程序可构建传统的应用程序界面 (API)，例如 OpenGL、Microsoft DirectX 或 D3D，所述应用程序界面 (API) 使应用程序及操作系统程序能够调用图形处理子系统 112 的各种功能，此在所属领域中众所周知。

[0055] 图形处理子系统 112 的操作较佳与系统 100 的其他操作不同步。例如，在某些实施例中，渲染命令缓冲器 128 与 PP 命令缓冲器 129 对经由系统总线 106 所接收的命令进行排队，以供由 GPU 114 执行。更具体而言，在 CPU 102 上执行的图形驱动器可将渲染命令流或程序写入至渲染命令缓冲器 128 并将后处理命令流或程序写入至 PP 命令缓冲器 129。所述命令流保持于其各自的命令缓冲器 128、129 中，直至 GPU 114 准备对其进行处理为止。

[0056] 渲染命令缓冲器 128 较佳构建为先进先出缓冲器 (FIFO)，其由 GPU（更具体而言，由在 GPU 上执行的图形驱动器）写入并由 GPU 114（更具体而言，由多处理器 120 上的渲染目标程序 132) 读取。读取与写入可不同步地进行。在一个实施例中，图形驱动器周期性地将新命令及数据写入至渲染命令缓冲器 128 中由“放置”指针所确定的位置上，在每一次写入之后，图形驱动器均递增所述“放置”指针。不同步地，渲染目标程序 132 可依序读取及处理先前存储于渲染命令缓冲器 128 中的所述命令以及数据。渲染目标程序 132 保持一“得到”指针，以识别命令缓冲器 128 中的当前读取位置，且该得到指针在每一次读取之后递增。在图形驱动器保持充分超前于渲染目标程序 132 的条件下，GPU 114 能够在不引起等待 CPU 102 的空闲时间的情况下渲染图像。在某些实施例中，视渲染命令缓冲器 128 及景物的复杂度而定，图形驱动器可正在为超前于渲染目标程序 132 当前所正渲染的图像的几个图像写入命令及数据。渲染命令缓冲器 128 可为固定尺寸（例如 5 兆字节），并可以环绕方式进行写入及读取（例如在写入至最末位置之后，图形驱动器可将“放置”指针复位至第一位置；类似地，在从最末位置读取之后，渲染目标程序 132 可将“得到”指针复位至第一位置）。

[0057] PP 命令缓冲器 129 较佳构建为圆形队列。图形驱动程序并非为每一个帧均写入新的后处理命令流，而是可向 PP 命令缓冲器 129 写入一次后处理流或程序并仅当要改变后处理时写入新的流或程序。多处理器 120 的后处理目标程序如下文所述对每一个帧执行该流，且所执行的命令可保留于 PP 命令缓冲器 129 中以供此后再执行。PP 命令缓冲器 129 可为固定尺寸（例如 1 兆字节），且可通过一无条件“跳转”命令来结束所述命令流，所述无条件“跳转”命令将控制转交回至所述命令流的开头以便处理下一个帧。

[0058] 在某些实施例中，多处理器 120 进行的渲染目标程序 132 的操作与任何其他操作均不同步，而后处理目标程序 134 的操作则与扫描输出引擎 124 的操作同步，以便以显示步

调进行后处理。例如，多处理器 120 可周期性地（例如每一个扫描输出帧一次）暂停渲染目标程序 132 并将后处理目标程序 134 操作一足够长的时间，以使后处理目标程序 134 能够产生屏幕像素数据帧并将该数据写入至显示缓冲器 127。下文将说明能提供适当操作的控制机制的实例。

[0059] 应了解，本文所述的系统只是例示性的，且也可具有各种变化形式及修改形式。GPU 可使用任何适当的技术来构建，例如构建为一个或多个集成电路装置。GPU 可安装于可包含一个或多个这种处理器的扩展卡上、直接安装于系统母板上、或集成至系统芯片组组件中（例如集成至一个常用 PC 系统架构的北桥芯片中）。所述图形处理子系统可包括任意量的专用图形存储器（某些实施方案可不具有专用图形存储器），并可使用系统存储器与专用图形存储器的任意组合形式。扫描输出电路可与 GPU 相集成或者提供于单独的芯片上，并可例如使用一个或多个 ASIC、可编程处理器元件、其他集成电路技术、或其任一组合来构建。此外，可将实施本发明的 GPU 并入各种装置中，包括通用计算机系统、视频游戏控制台及其他专用计算机系统、DVD 播放器、例如移动电话或个人数字助理等手持式装置、等等。

[0060] 数据流综览

[0061] 图 2 是图形处理子系统 112 的数据流图，其进一步图解说明多处理器 120、渲染目标程序 132、后处理目标程序 134 及扫描输出引擎 124 之间的关系。数据传播是由实线箭头表示，命令传播是由空心箭头表示，且控制信号传播是由虚线箭头表示。

[0062] 在图 2 中，将渲染缓冲器 126 显示为包括三个图像缓冲器 226（在本文中分别称作 A、B 及 C）及一旗语存储区域 206。图像缓冲器 226 分别大至足以存储一个图像的所有片段数据，且较佳操作渲染目标程序 132 来将一个图像的所有片段数据写入至缓冲器 A、将下一图像的所有片段数据写入至缓冲器 B，依此类推。应了解，可提供任意数量的图像缓冲器 226。旗语区域 206 用于如下文所述控制对图像缓冲器 226 的存取。

[0063] 图中显示显示缓冲器 127 包括两个帧缓冲器 227（在本文中分别称作 X 及 Y）以及一缓冲器选择 (BSEL) 存储区域 208。帧缓冲器 227 分别大至足以存储一完整帧的所有像素数据，且可操作后处理目标程序 134 以将一个帧写入至缓冲器 X、将下一个帧写入至缓冲器 Y、依此类推。BSEL 存储区域 208 大至足以存储一唯一地标识扫描输出引擎 124 将读取哪一帧缓冲器 227 的值。在一个实施例中，BSEL 存储区域 208 存储用于标识帧缓冲器 X（如果该位为 0）或帧缓冲器 Y（如果该位为 1）的单个位。在另一实施例中，BSEL 存储区域 208 存储图形存储器 116 中帧缓冲器 X 或帧缓冲器 Y 中一者的起始地址的标识符（例如偏移量值）。也可使用其他标识符。

[0064] 在该实施例中，渲染目标程序 132 例如经由系统总线 106 接收几何数据。可将所接收的几何数据存储于图形存储器 116 中、多处理器 120 的单芯片数据高速缓冲存储器（未显示）中、或其他地方。渲染目标程序 132 根据由渲染命令缓冲器 128 所提供的命令来处理几何数据，以产生图像的片段数据。所述片段数据写入至其中一个帧缓冲器 226 中，例如写入至缓冲器 A。

[0065] 后处理目标程序 134 根据从 PP 命令缓冲器 129 所接收的命令来处理图像缓冲器 226 中的片段数据，以产生帧的像素数据。所述像素数据写入至其中一个帧缓冲器 227，例如写入至缓冲器 X。在该实施例中，后处理目标程序 134 可从除渲染目标程序 132 当前所正

写入的一个图像缓冲器 226 以外的任何图像缓冲器 226 获得片段数据，例如如果渲染目标程序 132 正向缓冲器 A 进行写入，则从缓冲器 B 及 / 或 C 获得片段数据。

[0066] 扫描输出引擎 124 从其中一个帧缓冲器 227 读取像素数据，并将其提供至显示装置。在该实施例中，扫描输出引擎 124 从在 BSEL 存储区域 208 中所标识的帧缓冲器 227 中进行读取，所述帧缓冲器 227 较佳不与后处理目标程序 134 所正写入的帧缓冲器相同；例如，如果后处理目标程序 134 正向帧缓冲器 X 进行写入，则扫描输出引擎 124 将从缓冲器 Y 进行读取。扫描输出引擎 124 可使像素数据不加修改地通过，或者其可应用数字 - 模拟转换及 / 或其他信号处理算法将所述数据正确地格式化，以供递送至显示装置。在某些实施例中，在将像素数据写入至帧缓冲器 227 之前，由后处理目标程序 134 实施除数字 - 模拟转换以外的所有逐帧处理；在其他实施例中，扫描输出引擎 124 可实施某种处理。

[0067] 扫描输出引擎 124 较佳根据显示装置的要求将每一个帧的像素的递送同步化，并能够检测某些事件，例如一行屏幕像素的结尾（在所属领域中称作水平回扫）及屏幕的结尾（在所属领域中称作垂直回扫）。可使用传统技术来构建扫描输出引擎 124。在一替代实施例中，扫描输出引擎 124 在交替的帧上读取缓冲器 X 及 Y，且后处理目标程序 134 在交替的帧上对缓冲器 Y 及 X 进行写入。在该实施例中，可使用 BSEL 存储区域 208 来存储一旗语，所述旗语仅在初始化时用于使扫描输出引擎 124 与后处理目标程序 134 的操作同步。例如，可将要被扫描输出的第一缓冲器（例如缓冲器 X）的旗语存储于 BSEL 存储区域 208 中。在系统设置时，将该旗语（其可例如为单个位）设定为锁定（或被获取）状态，以阻止扫描输出引擎 124 读取缓冲器 X。在后处理目标程序 134 将第一个帧写入至缓冲器 X 之后，其便释放所述旗语，从而启用由扫描输出引擎 124 从缓冲器 X 中进行读取操作。此后，所述旗语可保持于其释放状态，并将保持正确的同步。

[0068] 同步技术

[0069] 较佳逐个帧地对扫描输出引擎 124 及后处理目标程序 134 的操作进行同步，此意味着对于扫描输出引擎 124 从显示缓冲器 127 中读取的每一个帧，后处理目标程序 134 均将一个新的帧写入至显示缓冲器 127。例如，较佳在扫描输出引擎 124 从帧缓冲器 Y 中读取一个帧的同时，后处理目标程序 134 将一个帧写入至帧缓冲器 X。当扫描输出引擎 124 达到帧缓冲器 Y 的末尾时，其便切换至读取帧缓冲器 X，同时后处理目标程序 134 将一个帧写入至帧缓冲器 Y。

[0070] 在一个实施例中，使用一触发器信号来处理该同步，所述触发器信号由扫描输出引擎 124 在“触发器”路径 246 上确定。所述触发器信号对于每一个帧产生一次，较佳与帧结尾事件呈固定的时间关系。所述帧结尾事件可为任何会表明扫描输出引擎 124 处于或接近帧结尾处的事件，例如垂直回扫（或 vsync）事件，从当前帧缓冲器中读取最末（或距最末像素还有 N 个）像素，或类似事件。响应于触发器信号被确定，多处理器 120 激活后处理目标程序 134 来产生一个新的像素数据帧。在产生新的帧之后，多处理器 120 便停用后处理目标程序 134，直至触发器信号下一次被确定为止。

[0071] 较佳提供缓冲器同步来防止在正对缓冲器进行扫描输出的同时，后处理目标程序 134 写入至帧缓冲器 227。在一个实施例中，每当后处理目标程序 134 结束一新的帧的产生时，其便将所述新的帧的帧缓冲器 227 的标识符（例如 X 或 Y）写入至 BSEL 存储区域 208 内，如由“BSEL_ 写入”路径 242 所指示。在每一个帧开始时，扫描输出引擎 124 均通过存

取 BSEL 存储区域 208 来确定要读取哪一帧缓冲器 227，如由“BSEL 读取”路径 244 所指示。只要每一个帧对 BSEL 存储区域 208 仅读取一次并每一个帧对 BSEL 存储区域 208 仅写入一次，便可对这些操作的定时进行协调以避免出现竞争状态。例如，扫描输出引擎 124 可恰好在产生触发器信号之前、在后处理目标程序 134 已结束产生新的帧并更新 BSEL 值之后，存取 BSEL 存储区域 208。

[0072] 渲染目标程序 132 与后处理目标程序 134 的操作较佳不相互同步，这是因为后处理目标程序 134 以显示步调（即屏幕刷新速率）产生帧，渲染目标程序 132 则以渲染步调产生图像，而渲染步调可不同于（例如慢于）显示步调。在某些实施例中，使用缓冲器同步技术来避免出现其中后处理目标程序 134 使用一个图像（或图像群组）来产生一帧的一部分、并使用一不同的图像（或图像群组）来产生同一帧的另一部分的情形，以及防止出现其中在后处理引擎 132 正从图像缓冲器 226 进行读取的同时渲染目标程序 132 向该缓冲器 226 进行写入的情形。具体而言，可控制对图像缓冲器 226 的存取，以使后处理目标程序 134 不从渲染目标程序 132 当前正在写入的图像缓冲器 226 进行读取，并使渲染目标程序 132 不会覆写仍由后处理目标程序 134 所需要的图像缓冲器 226。缓冲器同步机制可进一步使得在各后处理帧之间出现对后处理目标程序 134 所要读取的图像缓冲器的选择的任何改动；此有助于避免在所显示帧的不同部分中出现撕裂或其他视觉不一致现象。

[0073] 图 2 通过控制信号路径 232、234、236 图解说明一种基于不对称旗语及缓冲器索引的用于图像缓冲器 226 的缓冲器同步技术。渲染缓冲器 126 的旗语区域 206 存储旗语 207(SA，SB，SC)，以用于控制由渲染目标程序 132 对每一图像缓冲器 226 的存取，以便不将新的片段数据写入至图像缓冲器 226，直至后处理目标程序 134 已结束对于可存在于该缓冲器 226 中的任何旧片段数据的处理为止。每一旗语 207 较佳均具有“被获取”及“释放”状态，这些状态用于如下文所述进行存取控制。一般而言，传统的旗语数据结构及格式可适于构建旗语 207。

[0074] 在渲染目标程序 132 开始写入新图像的片段数据之前，其力图获取对应于该片段数据所要写入的图像缓冲器 226 的旗语 207，如由“获取”路径 232 所指示。只有在后处理目标程序 134 已将一图像缓冲器的旗语释放之后，渲染目标程序 132 才可获取该图像缓冲器的旗语；如下文所进一步说明，旗语释放是在后处理目标程序 134 不再需要相关联的图像缓冲器之后进行。如果不能立即获取旗语 207，则渲染目标程序 132 暂停其命令处理，并等待至例如可获得旗语等时间为止，然后继续进行渲染。当后处理目标程序 134 以显示步调（例如每一扫描输出帧一次）释放缓冲器时，渲染目标程序 132 一般将不需要等待。在其他实施例中，只要不需要以快于显示步调的速率渲染新的图像，渲染目标程序 132 的执行中的任何延迟均是可接受的。

[0075] 当渲染目标程序 132 结束将一图像的片段数据写入至其中一个图像缓冲器 226 时，其便释放该缓冲器的旗语 207，如由“释放”路径 234 所指示。释放旗语 207（例如旗语 SA）会防止渲染目标程序 132 存取相关联的缓冲器 226（例如缓冲器 A），直至例如后处理目标程序 134 释放旗语 207 为止，如下文所述。在释放旗语 207 之后（或与其同时），渲染目标程序 132 将新的索引值写入 PP 命令缓冲器 129 内，如由“索引写入”路径 236 所指示。所述索引值表示刚刚完成的图像缓冲器 226，并向后处理目标程序 134 发出信号来指示该图像缓冲器 226 现在可供用于处理。

[0076] 在每一个帧开始时，后处理目标程序 134 均从 PP 命令缓冲器 129 读取索引值，如由“索引读取”路径 238 所指示。在一个实施例中，将索引作为 PP 命令缓冲器 129 中其中一个命令的自变数进行读取。根据该索引值，后处理目标程序 134 确定其进行后处理需要哪一图像缓冲器 226，并释放所不需要的任何图像缓冲器 226 的旗语 207，如由“释放”路径 240 所指示。其旗语 207 被后处理目标程序 134 释放的图像缓冲器 226 变得可供渲染目标程序 132 使用。

[0077] 应注意，如果索引值上一个帧以来尚未出现变化，后处理目标程序 134 较佳不进行等待或拖延。在某些实施例中，无论索引值是否已出现变化，后处理目标程序 134 均实施相同的后处理；如果索引值尚未出现变化，则后处理目标程序 134 所产生的新的帧可与前一帧完全相同。在其他实施例中，可使后处理目标程序 134 的某些或所有活动以索引值是否已出现变化为条件。在此等实施例中，后处理目标程序 134 较佳存储前一帧所用的值，并将其与 PP 命令缓冲器 129 中的索引值进行比较，以判定索引是否已出现变化。如果索引尚未出现变化，则后处理目标程序 134 可只是不进行任何操作，或者其可执行不同的一组指令。

[0078] 就以下意义而言，本文中的同步机制是“不对称的”：旗语 207 可拖延渲染目标程序 132 的操作，但不拖延后处理目标程序 134 的操作。此种方案使得无论渲染目标程序 132 产生新图像的速率如何，均能够以显示步调产生新的帧。为避免出现死锁，较佳将图像缓冲器 226 的数量选择得足够大，以便始终存在至少一个未被后处理目标程序 134 使用的图像缓冲器 226。例如，如果后处理目标程序 134 需要使用 M 个最近的图像来进行其所要实施的某一（某些）特定操作，则将提供至少 M+1 个图像缓冲器 226。如果需要，可提供更大数量的图像缓冲器 226。提供额外的图像缓冲器使得渲染引擎的操作能够远远超前于扫描输出引擎。如果需要，也可使用额外的图像缓冲器在渲染目标程序 132 中执行经三重缓冲的渲染。

[0079] 应了解，本文所述的特定同步技术只是例示性的，且也可使用其他技术。某些实施例可完全省却缓冲器同步，从而允许在所显示的帧中出现撕裂。

[0080] 命令流

[0081] 在某些实施例中，以不同的方式管理渲染命令流与后处理命令流。更具体而言，如上文所述，渲染命令缓冲器 128 较佳构建成使渲染目标程序 132 将本文所述的渲染命令执行一次；针对所要渲染的每一图像，均将一组新的命令写入至渲染命令缓冲器 128 中。相比之下，所需的一组后处理操作通常在各帧之间保持一致，且针对每一个帧来写入相同的一组后处理命令将比较低效。当 PP 命令缓冲器 129 以无限循环形式操作时，向 PP 命令缓冲器 129 中写入新的后处理命令会更为复杂。后处理目标程序 134 的执行与图形驱动器不同步，且所述驱动器一般并不知晓后处理目标程序 134 在任一既定时刻是否正在执行或者正在执行所述循环中的哪一命令。

[0082] 相应地，PP 命令缓冲器 129 较佳构建成使图形驱动器可将整个后处理命令程序作为一命令流写入至 PP 命令缓冲器 129。在执行之后，将该程序保留于 PP 命令缓冲器 129 中，以便可针对每一个帧来重新执行同一程序。因此，仅当要改变后处理时，才需要将新的后处理命令写入至 PP 命令缓冲器 129 中。

[0083] 现在将对用于渲染命令缓冲器 128 及 PP 命令缓冲器 129 的实例性命令流结构进

行说明。

[0084] 图 3A 示意性地图解说明根据本发明一实施例的一渲染命令流 300 的一部分，所述渲染命令流 300 可在渲染命令缓冲器 128 中进行缓冲。对于每一新的图像，所述命令序列均以“ACQ”命令 306、308 开始，所述“ACQ”命令 306、308 指令渲染目标程序 132 在继续进行之前获取所要写入的下一图像缓冲器 226 的旗语 207（例如在为 ACQ 命令 306 的情况下是缓冲器 A 的旗语 SA，在为 ACQ 命令 308 的情况下是缓冲器 B 的旗语 SB）。如上文所述，如果无法获得旗语 207，则渲染目标程序 132 将等待至例如可获得旗语 207 等时刻。

[0085] “RCMD”命令 310、312 分别代表完整的一组命令及用于渲染图像的数据。在实际中，可针对每一图像包含任意数量的渲染命令。例如，RCMD 命令 310、312 可包含顶点（或几何形状）处理命令、光栅化命令、片段着色命令等等的任一组合。RCMD 命令 310、312 也可包含可与各种渲染命令相关联的适当数据（例如变换矩阵，将顶点分组成图元等等）。在某些实施例中，可将渲染命令作为适于由多处理器 120 执行的微指令来递送；在其他实施例中，可能要求在执行所述命令之前由主机 120 对所述命令进行解码。在某些实施例中，渲染命令序列也可包含例如循环、如果 - 则语句等流程控制命令。一般而言，RCMD 命令 310、312 可为传统性质，且由于对于理解本发明而言并不重要，因而不再对其进行详细说明。

[0086] 在每一图像的结尾处是一“翻转 (FLIP)”命令 314，316，其指示对当前图像的渲染完成。响应于所述 FLIP 命令，渲染目标 132 较佳将其渲染目标从当前渲染缓冲器切换至下一渲染缓冲器（例如在为翻转命令 314 的情况下从缓冲器 A 切换至缓冲器 B），并还释放刚刚完成的渲染缓冲器的旗语（例如在为翻转命令 314 的情况下为旗语 SA）。在释放旗语 207 之后，渲染目标程序 132 以新的值来更新 PP 命令缓冲器 129 中的索引值，所述新的值标识刚刚完成的渲染缓冲器（例如在为翻转命令 314 的情况下为缓冲器 A）。此外，对翻转命令的处理也可包括复位计数器或可由渲染目标程序 132 维护的其他变量。也可响应于翻转命令而执行其他传统的图像结束处理。

[0087] 图 3B 示意性地图解说明根据本发明一实施例可存储于 PP 命令缓冲器 129 中的后处理程序（或流）320。在该实施例中，程序 320 执行一无限循环，其包括：索引命令 322，其自变数是如上文所述由渲染目标程序 132 写入及覆写的索引；一个或多个后处理 (“PCMD”) 命令 326、328；一个或多个位置（“LOC”）命令 330、332；一 BSEL 更新（“BUPD”）命令 334；以及一最终无条件跳转（“JMPB”）命令 336，其用于返回至该循环的开头。

[0088] 索引命令 322 可首先执行，其将当前索引值提供至后处理目标程序 134。在某些实施例中，后处理目标程序 134 将当前索引值与最近使用的索引值相比较，并可根据比较结果来控制进一步的处理，所述最近使用的索引值可存储于可由后处理目标程序 134 进行存取的状态寄存器（在图 2 或图 3B 中未显示）中。例如，在某些其中索引值尚未出现变化的情形中，下一显示帧将与当前显示帧完全相同。在此种情形中，后处理目标程序 134 可不产生新的帧，且其可在位置 208（图 2）上不加修改地留下 BSEL 值，从而使扫描输出引擎 124 只是将同一帧缓冲器 227 接连地读取多次，直至例如索引值确实出现变化等时刻为止。在其他情形中，即使索引值尚未出现变化，也可能希望使后处理目标程序 134 采取某种措施，例如调节 LCD 过激励值或者前进经过例如淡入、淡出或者渐隐等过渡效果。在其他实施例中，无论索引值是否发生变化，后处理目标程序 134 均执行相同的命令。在初始化时，可将索引值设定为零值，此指示尚未将任何图像提供至图像缓冲器 226；后处理目标程序 134 较佳检

测所述零值并采取适当措施（例如将空白屏幕值写入至帧缓冲器 227 中的每一像素）。

[0089] PCMD 命令 326、328 代表用于所要执行的各种后处理操作的指令。在实际中，可在内容 320 中包含任意数量的 PCMD 命令。如同上文所述的 RCMD（渲染）命令一样，可将 PCMD 命令以适于由多处理器 120 直接执行的指令形式递送；在其他实施例中，可能要求在执行所述命令之前由主机对所述命令进行解码。同样，如同 RCMD 命令一样，PCMD 命令可根据需要包含流程控制命令（例如，某些命令可以索引条码 324 中的索引值是否出现变化为条件）。

[0090] 如下文所进一步说明，在某些实施例中，使用与渲染目标程序 132 相同的处理引擎来执行后处理目标程序 134；相应地，作为 RCMD 命令 310、312（图 3A）给出的任何命令序列也可作为 PCMD 命令 326、328 给出。此外，也可定义后处理操作所特有的特殊 PCMD 命令。原则上，可通过 PCMD 命令 326、328 的适当序列来构建后处理操作的任意组合（包括上文所述实例中的任何一者或多者）。在某些实施例中，PCMD 命令 326、328 仅用于应逐帧地进行的处理，例如 LCD 过激励或光标覆盖。然而，应了解，可通过 PCMD 命令 326、328 对各种各样的后处理操作进行编程。

[0091] LOC 命令 330、332 与索引条目 324 相结合地将一个（或多个）图像缓冲器 226 标识为特定的 PCMD 命令或特定的 PCMD 命令群组的数据源。索引条目 324 是由渲染目标程序 132 响应于上文所述的翻转命令而写入，其标识最近填满的图像缓冲器 226。LOC 命令 330、332 可根据需要指代最近填满的图像缓冲器 226 或者先前填满的各图像缓冲器 226。在一个实施例中，LOC 命令 330、332 相对于最近填满的图像缓冲器来标识图像缓冲器 226。例如，假定存在 N 个分配有索引值 0，1，...，N-1 的 N 个图像缓冲器 226，且索引条目 324 中的值由 R 标记。LOC 命令可为 LOC ＝ R 的形式，以标识最近填满的图像缓冲器，或者 LOC ＝ (R-1)，以标识次最近填满的缓冲器，依此类推。当使用相对引用方式时，在索引条目 324 更新时不需要修改 LOC 命令 330、332。在某些实施例中，可存在可适用于所有 PCMD 命令的一个 LOC 命令或一组 LOC 命令。

[0092] BUPD 命令 334 及 JMPB 命令 336 使后处理循环结束。BUPD 命令 334 指令后处理目标程序 134 将新近写入的帧缓冲器 227 的标识符写入至 BSEL 存储位置 208，以便使扫描输出引擎 124 将读取其作为下一个帧。JMPB 命令 336 将控制返回至程序 320 的开头，如由虚线箭头 338 所示。如下文所述，JMPB 命令 336 也可用作后处理程序的完成信号（在本文中也称作“帧完成事件”）；换句话说，一旦后处理目标程序 134 遇到 JMPB 命令 334，其便进入空闲状态以等待下一触发器事件。例如，如下文所述，在到达 JMPB 命令 336 时，后处理目标程序 134 可被上下文切换出。

[0093] 在某些实施例中，某些 PCMD 命令 326、328 可具有相关联的状态参数。例如，可使放大参数与执行扩大或缩小的 PCMD 命令相关联，或者可使覆盖尺寸或位置参数与执行合成的 PCMD 命令相关联。这些参数可包含于后处理命令流中，或者其可写入至可由后处理目标程序 134 存取的不同位置（例如 PP 命令缓冲器内的参数段，单独的缓冲器，或者执行核心中的参数寄存器）。可通过覆写后处理命令流、选择性地修改后处理命令流、或者修改缓冲器或寄存器的内容来处理参数更新。下文将进一步说明对命令及参数的修改。

[0094] 应了解，本文所述的程序流只是例示性的，且也可存在变化及修改形式。可使用任意适当的命令集合来构建 RCMD 及 PCMD 命令。可使用其他条件作为帧完成事件取代 JMPB

来指示后处理程序的结束。例如，如果使用“放置”指针及“得到”指针（类似于尚未参照图1中的渲染命令缓冲器128所述的指针）来构建PP命令缓冲器，则“放置”指针将引用所述程序中最末命令的位置。当“得到”指针与“放置”指针引用相同的位置时，便已到达最末命令。也可使用出现该事件作为帧完成事件来指示后处理目标程序已结束一帧。在其他实施例中，可使用任何唯一地与完成对帧的后处理相关联的事件或状态来作为帧完成事件。

[0095] 多处理器

[0096] 图4是根据本发明一实施例的多处理器120的简化方块图。多处理器120包括仲裁单元402、及包含上下文管理器406的执行核心404。

[0097] 执行核心404可为一般的传统设计，并可构建任何所需的微架构及/或指令集合。在某些实施例中，执行核心404包括若干个用于实施各种操作（加法、乘法、向量算术、数学函数、过滤算法、存储器存取等等）的功能单元405、以及相关联的控制逻辑407，所述相关联的控制逻辑407用于将输入命令解码为所述功能单元的可执行的指令、识别及收集每一指令所需的操作数、向所述功能单元发出指令、以及将结果转接至其他功能单元或寄存器文件（未明确显示）。执行核心404也可包括用于维护硬件状态的各个方面的各种其他寄存器。执行核心404还经配置以存取渲染缓冲器126及显示缓冲器127（其可如同1所示处于图像缓冲器116中）。在某些实施例中，可提供多个平行的执行核心404，其中每一核心404对一景物、图像或帧的一不同部分进行处理。执行核心404的具体架构对于本发明并不重要，且特定元件

[0098] 在一个实施例中，执行核心404支持两个独立的处理信道：构建后处理目标程序134（图1）的高优先权信道(HPC)，及构建渲染目标程序132的正常优先权信道(NPC)。上下文管理器406为每一信道维持单独的架构状态（上下文）。更具体而言，NPC上下文块408包含为NPC所存储的架构状态，且HPC上下文块410包含为HPC所存储的架构状态。可将架构状态存储于寄存器中或其他适合的硬体中。

[0099] 上下文管理器406使功能单元405及控制逻辑407能够存取适用于当前现用信道的架构状态，从而使一在其处理期间在某一点处被切换出的信道可在被重新切入时在同一点处重新开始。在某些实施例中，执行核心404具有一现用寄存器集合（在图4中未显示），且上下文管理器406通过如下方式执行信道切换（例如从HPC至NPC）：首先将“旧”信道（例如HPC）的架构状态从现用寄存器集合复制至适当的上下文块（例如HPC上下文块410），随后将“新”信道（例如NPC）的架构状态从适当的上下文块（例如NPC上下文块408）复制至现用寄存器集合。在其他实施例中，执行核心404可有选择地存取NPC上下文块408或HPc上下文块410中的对应寄存器，且上下文管理器406以与上下文相关的方式将执行核心404指引至适当的寄存器。上下文切换的具体实施方案对于本发明而言并不重要；在某些实施例中，可使用任何能提供足够快（例如300微秒或更快）的上下文切换的实施方案。

[0100] 仲裁单元402确定在任一既定时刻哪一信道将现用。在一实施例中，仲裁单元402具有用于从渲染命令缓冲器128接收备选命令的NPC输入端412及用于从后处理命令缓冲器129接收备选命令的HPC输入端414。仲裁单元402还具有用于从扫描输出引擎124接收触发器信号（上文中参照图2所述）的触发器输入端416、及用于从执行核心404接收事件信号的事件输入端418。根据这些输入，在每一循环中，仲裁单元402均选择HPC或NPC

作为现用信道，向执行核心 404 发送一标识所选信道的上下文信号（在路径 420 上）及用于所选信道的备选命令（在路径 422 上）。未被选定的备选命令在下一循环中重新提供于输入端上，此时，其既可能被选定也可能不被选定。可使用适当的定时及控制电路（在图 4 中未显示）来确保在执行核心 404 具有所选命令的正确上下文时执行所选命令；该电路可为一般的传统设计。

[0101] 图 5 是可在仲裁单元 402 中执行的选择逻辑过程 500 的流程图。在该实施例中，扫描输出引擎 124 在路径 424 上与每一垂直回扫（或其他帧结尾）事件大致重合地产生触发器信号。后处理命令缓冲器 129 包含用于具有无条件 JMPB 命令的无限循环的命令，如在图 3B 中所示。

[0102] 在初始化（步骤 501）之后，仲裁单元 402 切换入 NPC（步骤 502）。在每一循环中，仲裁单元 402 均检查触发器信号是否被确定（步骤 504）。只要触发器信号未被确定，便继续选择来自渲染命令缓冲器 128 的命令（步骤 506）。根据渲染命令缓冲器 128 中的命令，渲染图像并将片段数据写入至渲染缓冲器 126；换句话说，渲染目标程序 132 运行。HPC 在该时间期间不现用。

[0103] 当在步骤 504 中检测到触发器信号时，仲裁单元 402 将上下文切换至 HPC（步骤 508）并开始从 PP 命令缓冲器 129（步骤 510）中选择命令。仲裁单元 402 继续从 PP 命令缓冲器 129 中选择命令，直至在步骤 512 中检测到无条件 JMPB 命令为止。根据 PP 命令缓冲器 129 中的命令，对渲染缓冲器 126 中的片段数据实施后处理，将所得到的屏幕像素数据写入至显示缓冲器 127 中；换句话说，后处理目标程序 134 运行。帧的后处理的完成是由 JMPB 命令来指示，如上文所述。一旦检测到该命令（步骤 512），并将其转接至核心 404（步骤 514），此后仲裁单元 402 将切换入 NPC 上下文，从而返回至步骤 502。在某些实施例中，JMPB 命令仅影响仲裁单元 402，且不转接至核心 404。此时，渲染目标程序 132 重新开始。

[0104] 过程 500 以对应于显示帧速率（例如 80Hz）的间隔中止渲染，从而可以显示步调进行后处理。一般而言，如此周期性地中止渲染而进行后处理不会不利地影响图形子系统的总体性能，除非后处理所耗用（且因此不能用于渲染）的时间长到足以使得无法以所需速度（例如 30Hz）渲染新图像为止。在现代执行核心中可具有的处理能力及典型的后处理操作序列情况下，可保持可接受的性能。

[0105] 应了解，本文所述的多处理器及仲裁逻辑只是例示性的，且也可存在变化及修改形式。在本文中，“上下文切换”作为一般性术语用于描述可在第一指令流中的任意一点从所述第一指令流切换至另一指令流、然后在实质上同一点处返回第一指令流的处理器的操作；该术语限定于任意特定处理器架构。可采用任何多任务架构，包括传统的多线程架构，且可使用任何机制来保持一临时空闲的过程或上下文的状态。此外，不需要赋予 HPC 优先于 NPC 的绝对优先权，只要 HPC 可在由显示步调所确定的时间表内可靠地结束新帧的产生既可。应注意，本文所述的实例性仲裁逻辑并不利用事件输入 418；不同的仲裁逻辑可使用该输入。例如，后处理目标程序 134 可在输入路径 418 上产生事件信号来通知仲裁单元 402 其已结束一帧。

[0106] 在另一实施例中，可对各单独的缓冲器读取路径使用独立的处理引擎来构建渲染目标程序及后处理目标程序。在该实施例中，渲染目标程序及后处理目标程序可并行地执行，且不要求在这些目标程序之间进行上下文切换。

[0107] 修改后处理程序

[0108] 在上文所述的实施例中，PP 命令缓冲器 129 一般并不逐帧地更新。可在后处理目标程序 134 空闲时的任意时刻，例如通过在图 1 中的 CPU 102 上运行的图形驱动器来更新 PP 命令缓冲器 128，但这些更新并不与任何特定图像的显示同步。相反，对 PP 命令缓冲器 129 的更新通常会在更新之后对下一个帧生效。如果进行多次更新，则不保证所有更新均将对同一帧生效。

[0109] 在其他实施例中，可使后处理操作的改变与特定图像同步化。该可选特征还使所要控制的多次更新能够对同一个帧生效。可例如通过构建多个 PP 命令缓冲器 129 来提供此种同步，其中渲染目标程序 132 控制 PP 命令缓冲器 129 的内容以及对于在任一既定时刻应读取哪一 PP 命令缓冲器 129 的选择。

[0110] 图 6 是根据本发明一实施例的多处理器 600 的简化方块图，多处理器 600 构建多个 PP 命令缓冲器 602(1)-602(3)。多处理器 600 包括仲裁单元 604 及具有上下文管理器 608 的执行核心 606。这些组件可大体上类似于上文所述的仲裁单元 402、执行核心 404、及上下文管理器 406；因此，主机 600 提供一执行后处理目标程序 134 的 HPC 及一执行渲染目标程序 132 的 NPC。多处理器 600 还包括用于控制在各 PPC 命令缓冲器 602(1)-602(3) 之间进行选择的附加逻辑。

[0111] 更具体而言，每一 PP 命令缓冲器 602(1)-602(3) 均向选择电路（例如多路复用器）610 提供备选后处理命令。选择电路 610 响应于存储于主机 600 的 PP 索引寄存器 612 中的“PP 索引”值而选择其中一个备选项。所选备选项作为 HPC 备选指令提供至仲裁单元 604。仲裁单元 604 可如上文所述选择 HPC 与 NPC 备选指令。

[0112] NPC 是响应于特殊命令而写入 PP 索引寄存器 612，所述特殊命令可包含于通过渲染命令缓冲器 614 递送至主机 600 的渲染命令流中。图 7 图解说明包含适当命令的渲染命令流 700 的一部分。在流 700 中，ACQ 命令 702、RCMD 命令 704 及翻转命令 710 大体类似于上文参照图 3A 所述的命令。PP 命令缓冲器选择（“SELPP”）命令 706 及 PP 命令缓冲器写入（“WRPP”）命令 708 执行与特定图像同步的后处理操作中的变化。

[0113] 更具体而言，SELPP 命令 706 指令渲染目标程序 132（构建于 NPC 中）选择所要写入的 PP 命令缓冲器 602(1)-602(3) 中可用的一个。可使用类似于上文所述旗语 207（图 2）的旗语机制或其他控制机制来防止渲染目标选择仍由后处理目标程序 134 使用的 PP 命令缓冲器 602。

[0114] WRPP 命令 708 较佳伴随有数据（在图 7 中未明确显示），所述数据呈要写入所选 PP 命令缓冲器 602 的后处理程序（或其一部分）的形式。所述程序可根据需要具有图 3B 中所示的形式或者不同的形式。渲染目标程序 132 通过经由图 6 中所示的“PPC 写入”路径 616 将相关联的程序写入至所选 PP 命令缓冲器 602 中而执行 WRPP 命令 708。路径 616 可包含适用于有选择地将程序信息指引至仅其中一个 PP 命令缓冲器 602(1)-603(3) 的电路（未显示）；该电路可为传统设计。在某些实施例中，在执行 WRPP 命令 708 期间，还将一标识渲染目标程序 132 刚刚完成的图像缓冲器的索引条目写入至所选 PP 命令缓冲器 602 中。

[0115] 此后，渲染目标程序 132 执行翻转命令 710 并经由“PPI- 写入”路径 618 将所选 PP 命令缓冲器 602 的标识符写入至 PP 索引寄存器 612 中。可沿路径 618 提供适当的死锁（例如由仲裁单元 604 控制的锁存器或传输门），以便在 HPC 现用时，对 PP 索引寄存器 612

进行的更新不会生效。举例而言，在 PP 索引寄存器 612 正由后处理目标程序 134 使用（锁定）时，可停止渲染目标程序 132 执行 PP 索引写入操作。应注意，在该实施例中，WRPP 命令可包括将适当的索引值写入至所选 PP 命令缓冲器 602；并不要求进行单独的索引更新操作。

[0116] 后处理目标程序 134（构建于 HPC 中）跟踪在其最后一次迭代中是读取了哪一 PP 命令缓冲器 602，且还能经由“PPI- 读取”路径 620 对 PP 索引寄存器 612 进行读取存取。相应地，后处理目标程序 134 可判定当前 PP 命令缓冲器 602（由 PP 索引寄存器 612 规定）与最后所读取的 PP 命令缓冲器 602 是否相同；如果不相同，则后处理目标程序 134 将最后所读取的 PP 命令缓冲器 602 的旗语解锁或者以其他方式释放最后所读取的 PP 命令缓冲器 602 以便进行覆写。

[0117] 在某些实施例中，SELPP 命令 706 及 WRPP 命令 708 包含于仅用于其中要改变后处理程序的图像的渲染命令流 700 中。对于其他帧，可省略这些命令，在此种情形中，翻转命令 710 较佳不改变存储于 PP 索引寄存器 612 中的值。相反，可执行对当前 PP 命令缓冲器 602 的索引更新。因此，可对任意数量的连续帧使用同一 PP 命令缓冲器 602，其中仅在实际改变后处理程序时使用新的 PP 命令缓冲器 602。

[0118] 在另一实施例中，由 WRPP 命令递增地更新而非完全覆写 PP 命令缓冲器 602 中的后处理程序。举例而言，在初始化时，可对每一 PP 命令缓冲器 602 加载以相同的“缺省”后处理程序。渲染目标程序 132 根据需要修改所述程序的某些部分（例如各种操作的参数），并更新索引值。上文所述的 SELPP 及 WRPP 命令也可适用于该实施例，例如使 WRPP 命令规定所要覆写的 PP 命令缓冲器 602 中的条目以及新的命令（或仅参数值）。

[0119] 所属领域的一般技术人员将知，也可具有其他同步方案。例如，可根据需要将关于接下来应读取哪一 PP 命令缓冲器的信息存储于仲裁单元中、HPC 上下文中的存储器中、或其他位置处。

[0120] 多个显示头

[0121] 如在所属领域中所知，某些 GPU 设计有多个显示头以用于驱动多个显示装置。通常为每一显示头分别提供单独的扫描输出引擎，这是因为不同的显示装置可能以不同的像素或帧率运行，或者可能具有不同的像素格式要求。

[0122] 图 8 是一简化方块图，其图解说明根据本发明一实施例的多处理器 804，多处理器 804 支持一用于驱动显示装置“0”（未明确显示）的第一扫描输出引擎 802(1) 及一用于驱动显示装置“1”（也未明确显示）的第二扫描输出引擎 802(2)。多处理器 804 包括仲裁单元 806 及包含上下文管理器 810 的执行核心。这些组件可大体上类似于图 4 中的对应组件，只是多处理器 804 支持两个 HPC（在本文中标记为 HPC0 及 HPC1）及一个 NPC。每一 HPC 均执行不同的后处理目标程序；对于显示装置“0”为 HPC0，而对于显示装置“1”为 HPC1；所述 NPC 包含用于提供由这两个后处理目标程序所耗用的图像的实例化渲染目标程序。在某些实施例中，可存在多个渲染目标，其中这些渲染目标程序所产生的图像由一个或多个后处理目标程序以任意所需方式耗用。相应地，执行核心 808 支持至少三个上下文，且仲裁单元 806 如下文所述实施三路（或更多路）仲裁。

[0123] 每一扫描输出引擎 802(1)、802(2) 均以适用于其各自显示装置的参数（例如像素率、帧率）来运行。由于不同显示装置的后处理操作可有所不同，因而构建于 HPC0 中的后

处理目标程序较佳将帧写入至显示缓冲器 820(1) 中，而构建于 HPC1 中的后处理目标程序将帧写入至在物理或逻辑上分离的显示缓冲器 820(2) 中。显示缓冲器 820(1) 由扫描输出引擎 802(1) 读取，而显示缓冲器 820(2) 由扫描输出引擎 802(2) 读取。HPC0 及 HPC1 二者均可从同一渲染缓冲器 822 读取，并可读取存储于其中的同一图像或不同图像，或者其可从不同的渲染缓冲器 822 读取。

[0124] 仲裁单元 806 经由渲染命令缓冲器 812 接收 NPC 的备选命令流，经由 PP 命令缓冲器 814(1) 接收 HPC0 的备选命令流，且经由 PP 命令缓冲器 814(2) 接收 HPC1 的备选命令流。一般而言，由于不同显示装置所需的后处理操作可能不同，因而 PP 命令缓冲器 814(1)、814(2) 可提供不同的后处理程序。

[0125] 仲裁单元 806 分别从每一扫描输出引擎 802 接收触发器信号 :“触发器 -0”路径 824(1) 从扫描输出引擎 802(1) 提供触发器信号，而“触发器 -1”路径 824(2) 从扫描输出引擎 802(2) 提供触发器信号。响应于触发器 -0 信号，仲裁单元 806 切换入 HPC0- 其运行至帧完成（如在上文所述的其他实施例中一样），且响应于触发器 -1 信号，仲裁单元 806 切换入 HPC1- 其也运行至帧完成。

[0126] 在某些情形中，可存在交叠的触发器事件。例如，仲裁单元 806 可响应于触发器 -0 信号而切换入 HPC0，然后在 HPC0 已结束其帧之前接收触发器 -1 信号。在一个实施例中，仲裁单元 806 允许 HPC0 结束其帧，然后立即切换入 HPC1 中。也可使用类似逻辑来处理其中触发器 -1 信号首先到达的另一交叠情景。只要为使 HPC0 与 HPC1 分别产生帧所需的总时间明显小于帧时间（如果这两个显示装置具有不同的帧率，则以更快的帧率），且只要为 NPC 留下足够的处理时间来以可接受的速率渲染新图像，该算法便有效。

[0127] 在其他实施例中，可将各种目标程序中的某些或全部构建于单独的处理引擎中，从而提高并行处理容量。例如，对于三个引擎，可将 NPC 及两个 HPC 分别构建于单独的引擎中。

[0128] 在另一实施例中，可通过提供两个扫描输出引擎及单个 HPC 来支持多个显示头，所述扫描输出引擎中的一者独自使用 HPC。另一显示头的扫描输出引擎可在需要时使用传统的专用电路来构建有限的后处理能力。在再一实施例中，可由来自两个（或更多个）扫描输出引擎之一的触发器信号来触发单个 HPC。

[0129] 后处理的实例

[0130] 如上文所述，后处理可包括各种各样的操作。现在将说明后处理操作的某些实例，此并不限制本发明的范畴。

[0131] 一个后处理操作是合成，其中可将来自不同缓冲器的图像相互叠加而产生帧。例如，图 9 图解说明自三个不同图像缓冲器产生的合成帧 900。背景图像缓冲器 902 提供用于在背景区域 904 中产生像素的片段数据，电影图像缓冲器 906 提供用于在电影区域 908 中产生像素的片段数据，且光标图像缓冲器 910 提供用于在光标区域 912 中产生像素的片段数据。

[0132] 在该实例中，任何图像的内容及 / 或位置均可分别因帧而异，且不同图像的更新率可有所不同。例如，区域 904 中的背景图像可为很少发生变化的桌面壁纸。区域 908 中的电影图像可以 30Hz 或更高的速率变化，且光标图像缓冲器 910 的内容也可实质上实时地变化，例如以指示当前的系统活动。光标区域 912 的位置也可实质上实时地变化，以反映用

户所操作的指点装置的运动。

[0133] 根据本发明的一实施例，可通过提供多个独立的图像缓冲器（或图像缓冲器群组）作为后处理目标程序的片段数据源来产生合成图像。图 10 是一方块图，其显示经配置以分别产生多个图像的多处理器 1000。在该实施例中，多处理器 1000 支持一个构建后处理目标程序的 HPC 及两个 NPC：NPCF 构建用于全屏图像的渲染目标程序，而 NPCC 构建用于光标图像的渲染目标程序。应了解，尽管在图 10 中图解说明两个 NPC，然而可提供任意数量的 NPC 来产生可能需要组合的任意数量的图像源。此外，可使用任意数量的不同处理引擎来构建目标程序；例如，一个引擎可构建所有三个目标程序，可为每一目标程序提供单独的引擎，或者可每一目标程序提供多个引擎。

[0134] 图形存储器 1002 可大体上类似于上文所述的图形存储器 116，其包括两个渲染缓冲器 1004、1006，其中渲染缓冲器 1004、1006 中的每一者均如上文所述存储多个图像。渲染缓冲器 1004 存储由 NPCF 所渲染的全屏图像的片段数据，而渲染缓冲器 1006 存储由 NPCC 渲染的光标图像的片段数据。应注意，光标图像可包含比全屏图像少得多的片段；相应地，渲染缓冲器 1006 可明显小于渲染缓冲器 1004。图形存储器 1002 还包括显示缓冲器 1008，用于存储由 HPC 所产生的帧的像素数据。

[0135] 主机 1000 包括仲裁单元 1010 及具有上下文管理器 1014 的执行核心 1012，上下文管理器 1014 可大体上类似于上文所述的对应组件。在此种情形中，上下文管理器 1014 存储三个不同的上下文（NPCC 上下文 1011，NPCF 上下文 1013 及 HPC 上下文 1015），且仲裁单元 1010 在三个不同上下文之间进行选择。用于全屏图像及光标图像的单独的渲染命令流经由各自的渲染命令缓冲器 1016（用于全屏图像）及 1018（用于光标图像）提供。这些渲染流可大体上类似于图 3A 所示的渲染流 300。

[0136] 仲裁单元 1010 响应于来自扫描输出引擎 1030（如上文所述）的触发器信号而选择 HPC，并在 HPC 空闲时在 NPCC 与 NPCF 之间进行选择。可使用循环选择、事件驱动的选择（例如根据经由“evt”路径 1017 从执行核心 1012 接收的事件信号）、或其他选择算法。

[0137] PP 命令缓冲器 1020 提供后处理程序，其一实例图解说明于图 11 中。后处理程序 1100 大体上类似于图 3B 所示的后处理程序 320，只是提供多个索引项 1104、1106。索引项 1104 在新的光标图像完成之后存储由 NPCC 所更新的光标索引值 (INDC)，而索引项 1106 则在新的全屏图像完成之后存储由 NPCF 更新的全屏索引值 (ENDF)。应注意，可相互独立地并以不同的速率来更新 INDC 与 INDF 值。

[0138] 在该实施例中，后处理目标程序较佳每当得到触发时便产生新的帧，而无论两个索引项中的一个是否已发生变化。此会确保如常常所需要的那样以显示步调更新屏幕上的光标位置。

[0139] 在一个实施例中，PCMD 命令 1110、1112 包括用于如下的命令：从适当的寄存器（其可为传统设计）或自存储器位置读取当前光标位置；确定当前光标大小（其可例如通过 PP 命令缓冲器 1020 中的命令进行设定）；确定哪些像素处于光标区域内；及根据哪些像素处于光标区域内而选择来自当前光标图像（由索引值 INDC 标识）或当前全屏图像（由索引值 INDF 标识）的每一像素的一个或多个片段。所述合成算法自身可类似于现用的基于硬件的算法，但其较佳构建于可在执行核心 912 中执行的程序名中，而非构建于专用硬件中。例如透明度或光标与下面全屏图像部分之间的边缘混合等特征也可经由适当的程序

码来构建。

[0140] 重新参见图 10，扫描输出引擎 1030 可与上文所述的各种扫描输出引擎相同。在该实施例中，所有合成均由 HPC 中的后处理目标程序在主机 1000 中进行，且扫描输出引擎 1030 的操作与如何产生显示缓冲器 1008 中的像素无关。

[0141] 另一后处理操作是 LCD 过激励（在所属领域中也称作“LCD 前馈”或“响应时间补偿”(RTC)）。如在所属领域中所知，如果部分地根据所需的新强度且部分地根据所需新强度与前一强度之间的差别在各个帧之间调整用于驱动像素的信号，则可使 LCD 屏幕更快地作出响应。根据本发明的一实施例，可将 LCD 过激励构建于后处理目标程序中，所述后处理目标程序执行一包含适当命令的后处理程序。

[0142] 例如，考虑图像的每一片段均对应于帧中的一个像素的情形。（本发明并不仅限于此种情形；其在此处用于图解说明目的。）图 12 图解说明用于产生帧的过程 1200，其可通过适当的后处理程序（例如如在图 3B 中所示）构建于后处理目标程序 134 中（例如参见图 2）。过程 1200 较佳地构建于一如下实施例中：其中将最新完成的图像存储于具有索引值 R 的图像缓冲器 226（图 2）中，并将紧接的前一图像存储于索引值为 (R-1) 模 N 的不同图像缓冲器 226 中；这两个图像缓冲器 226 均由后处理目标程序 134 保持锁定。

[0143] 在步骤 1202 中，后处理目标程序 134 检查 PP 命令缓冲器 129 中的索引条目。在步骤 1204 中，判定索引值是否已发生变化。如果未发生变化，则在步骤 1206 中，将帧中所有像素的增量值设定为 0，从而反映所需像素值无一发生变化。如果像素值已发生变化，则在步骤 1208 中，根据缓冲器 R 中的片段值及缓冲器 (R-1) 模 N 中的片段值来计算每一像素的增量值。在一个实施例中，所述增量值简单地为这两个缓冲器中对应片段的值之差。

[0144] 在步骤 1210 中，根据缓冲器 R 中的片段值来确定所需的像素强度。在一个实施例中，所述片段值是所需的像素强度；在其他实施例中，所需像素强度是所述片段值的函数（例如包含 γ 修正）。

[0145] 在步骤 1212 中，根据所需强度及增量值来确定像素的过激励值。在一个实施例中，可规定一用于计算过激励信号的函数。在另一实施例中，可根据一使用所需强度及增量值（或新的及旧的强度值）来加索引的查找表来确定过激励信号；执行核心 404 可包括经配置以实施表查找的功能单元。

[0146] 应了解，过程 1200 只是例示性的，且也可具有各种变化形式及修改形式。各步骤可对每一片段依序执行或者对片段群组（或对所有片段）并行执行，可改变步骤次序，并可修改或组合各步骤。可替换对增量值的不同定义，并还可引入对片段值的其他修改。此外，可将过程 1200 与用于从片段数据产生像素数据的其他步骤（例如下降过滤、上升过滤、合成等等）相组合。

[0147] 应了解，过程 1200 为像素产生的过激励信号并非一定是将存储于帧缓冲器 227 中的最终像素值。可使用额外的后处理命令来执行随后的操纵。

[0148] 后处理操作的第三实例是表面旋转，在表面旋转中，将图像的片段数据映射至已围绕垂直于图像的轴线旋转过某一角度（例如 90° 的倍数）的 2-D 表面上。所述轴线与图像平面的交点及 / 或所述旋转角度可以是可配置的参数。在一个实施例中，所述交点固定于图像中心处（或附近），且所述角度是可配置的 90° 的倍数（例如 0°、90°、180°、270°）。根据所述角度及旋转角度，可将图像缓冲器中的每一片段位置映射至帧缓冲器中

的对应像素位置。相应地，可通过如下方式在后处理目标程序中执行表面旋转：在片段位置与像素位置之间定义一映射，并使用该映射根据从中读取片段的地址来确定每一像素的写入地址。所述映射可例如作为可按图像缓冲器的位置偏移量来存取的查找表的形式或者作为用于根据图像缓冲器的位置偏移量来计算像素缓冲器的位置偏移量的公式形式提供。在一个实施例中，可使用可按图像缓冲器位置偏移量存取的查找表及当前旋转角度来为若干个所允许的旋转角度提供映射。

[0149] 图 13 是一用于产生旋转帧的过程 1300 的流程图，其可通过适当的后处理程序（例如，如在图 3B 中所示）而构建于后处理目标程序 134（例如参见图 2）中。在步骤 1302 中，根据所需的旋转，在图像缓冲器中的每一片段位置与像素缓冲器中的对应像素位置之间定义一映射。所述映射可例如在渲染目标程序的初始化期间定义一次，且仅当旋转参数发生变化时才进行更新。在一个实施例中，使用后处理命令来提供映射；在另一实施例中，后处理命令提供可由后处理目标程序用以计算或选择适当映射的参数。

[0150] 在步骤 1304 中，从图像缓冲器位置（在本文中称为源位置）读取片段数据，并在步骤 1306 中处理片段数据以产生像素值。可在步骤 1306 中包含任意类型的后处理操作，包括例如合成及 / 或 LCD 过激励操作。一旦完成处理，便在步骤 1308 中使用源位置及在步骤 1302 中所定义的映射为像素值确定帧缓冲器中的目的地位置。在步骤 1310 中，将像素值写入至目的地位置。在步骤 1312 中，判定是否还有更多片段要处理。如果有，则过程 1300 返回至步骤 1304 来处理下一片段；当当前图像的所有片段均已得到处理时，过程 1300 结束（步骤 1314）。

[0151] 应了解，过程 1300 只是例示性的，且也可具有变化形式及修改形式。被描述为按顺序的各步骤也可并行执行，可改变步骤次序，并可修改或组合各步骤。举例而言，可并行处理多个片段。在某些实施例中，源位置与目的地位置之间的映射可并非一对一映射；例如，也可减小或增大旋转图像来适合于像素阵列的尺寸。减小或增大可涉及到混合数个片段值以产生一个像素值。过程 1300 可适用于该实例，只要可使用一个片段的位置作为源位置来用于定义向像素的映射即可。在其他实施例中，可将所述映射定义成对要用于每一像素位置的一个或多个源位置进行标识。

[0152] 在本文中使用合成操作及 LCD 过激励操作作为可在后处理目标程序中执行的操作的实例。也可除这些实例之外或者取代这些实例而执行其他操作；上文在“概述及术语”一节中阐述了某些实例，但本发明并不仅限于特定的后处理操作。所属领域的一般技术人员将知，后处理目标程序可有利地用于执行任何为获得正确行为而希望或需要以显示步调执行的操作。可改变包含于后处理目标程序中的操作的数量及复杂度，只要可以显示步调产生新的帧即可。

[0153] 用于后处理程序的源

[0154] 如上文所述，通过图形驱动器程序将后处理程序写入至 PP 命令缓冲器。所述驱动器可直接（例如，通过将命令经由系统总线 106 传送至图 1 所示实施例中 PP 命令缓冲器 129 的存储器位置内）或者间接地（例如，通过如上文参照图 7 所述将 WRPP 命令写入至渲染命令缓冲器 128 中）写入命令。

[0155] 现在将对用于后处理程序的源进行说明。在某些实施例中，图形驱动器对来自由执行各种后处理操作的预定程序段形成的库的后处理程序进行组合。图形驱动器可提供适

当的应用程序界面 (API)，从而允许应用程序开发者规定其希望使用的后处理操作及任何相关参数。API 自身可大体上类似于用于控制扫描输出时间处理的传统 API，但在图形驱动器内的实施方案是不同的。具体而言，响应于 API 指令，图形驱动器从所述库中选择适当的程序段，将所述段组合成完整的程序（例如，增加例如上文所述的索引检查命令及跳转命令等控制命令），并将所述程序写入至 PP 命令缓冲器中或将适当的 WRPP 命令插入渲染命令流内。

[0156] 在其他实施例中，应用程序开发者可定义“定制”后处理程序。例如，图形驱动器可提供一从应用程序或操作系统程序接受任意后处理程序或程序段的 API。也可提供具有用于写入定制后处理程序的指令及 / 或建议的开发者套件。在再一些实施例中，可支持库程序段及定制程序段的组合。

[0157] 在上文所述的实施例中，将图形驱动器标识为向 PP 命令缓冲器中写入后处理程序。在某些实施例中，在抽象层上构建图形驱动器程序，所述抽象层次隐藏了在上面执行驱动器程序的硬件及 / 或操作系统的细节。例如，可提供资源管理器程序来支持图形驱动器程序与系统硬件组件的交互作用。所述资源管理器构建低层硬件及针对具体操作的功能度，这些低层硬件及针对具体操作的功能度由图形驱动器程序响应于来自操作系统程序及 / 或应用程序的处理请求而被调用。该额外的抽象层通过提供资源管理器的不同实施方案而使同一驱动器代码能够适用于不同的硬件配置。当构建资源管理器时，可由所述资源管理器来处理后处理程序向 PP 命令缓冲器的实际写入，从而使其对驱动器透明。

[0158] 在又一实施例中，可由驱动器发出传统的扫描输出控制命令。这些命令由 GPU 内的适当命令接口组件接收并转换成用于后处理目标程序的控制结构。例如，在接收到扫描输出命令之后，所述命令接口组件可在 PP 命令缓冲器中写入或覆写命令或参数，或者，其可将适当的 SELPP 及 / 或 WRPP 命令插入渲染命令流内。因此，可使 PP 命令缓冲器及后处理目标程序对图形驱动器（包括资源管理器）透明，且本发明可与未经修改的传统图形驱动器一起使用。应了解，可使用由 GPU 中的适当命令接口组件操作的除 PP 命令缓冲器以外的其他控制结构来控制后处理目标程序的行为。

[0159] 其他实施例

[0160] 如上文所述，本发明的实施例使 GPU 的执行核心中可用的大量处理能力能够用于以显示步调进行像素处理。此外，本文中所述的后处理目标程序的可编程性在可实施的显示后处理的量及类型方面提供比传统专用电路明显更大的灵活性。此外，本发明的某些实施例能够明显减少或消除传统 GPU 中专用后处理电路的量，从而在芯片面积及功耗方面具有优点。

[0161] 尽管上文参照具体实施例来说明本发明，然而所属领域中的技术人员将知，也可存在诸多修改形式。例如，本发明并不仅限于任何特定执行核心或上下文管理架构。此外，尽管上文是说明后处理操作的具体实例，然而应了解，可在本发明的范畴内构建任何操作或操作组合，包括在本文中未明确提及的操作。

[0162] 本文的说明提及了处理目标程序，例如渲染目标程序及后处理目标程序。在该上下文中，“目标程序”应理解为是指具有状态及行为方面的实体。所属领域的一般技术人员将知，可通过各种方式在多处理器中构建多个目标程序。例如，可构建具有上下文切换特征的单个处理引擎，以便能够根据需要切入及切出不同的目标程序。另一选择为，本文所述的

任何或所有目标程序均可使用能够实现该目标程序的状态及行为方面的单独的处理引擎来构建。

[0163] 此外，尽管上文参照具体硬件及软件组件来说明本发明，然而所属领域的技术人员将知，也可使用硬件及／或软件组件的不同组合，且被描述为在硬件中执行的特定操作也可在软件中执行，反之亦然。

[0164] 可将包含本发明各种特征的计算机程序编码于各种计算机可读媒体上进行存储及／或传输；适合的媒体包括磁盘或磁带、例如 CD（光盘）或 DVD（数字通用光盘）等光学存储媒体、快闪存储器、以及适于通过符合各种协议的有线网络、光学网络及／或无线网络（包括因特网）进行传输的载波信号。以程序码进行编码的计算机可读媒体可使用兼容的装置一起进行包装或与其他装置分开提供（例如通过因特网下载）。

[0165] 因此，尽管上文参照具体实施例来说明本发明，然而应了解，本发明旨在涵盖归属于上文权利要求书范围内的所有修改及等价形式。

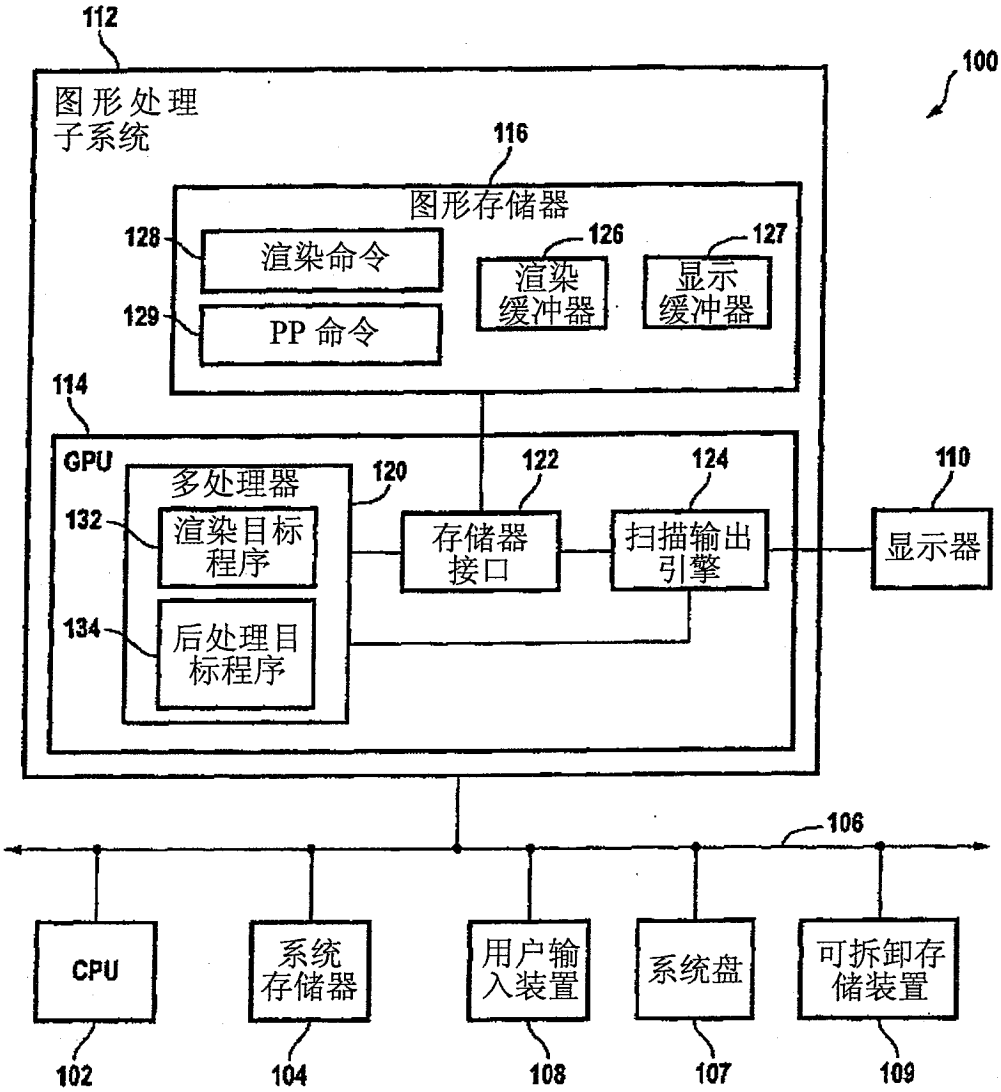

112
100
图形处理
子系统
116
图形存储器
128
渲染命令
126
渲染
缓冲器
127
显示
缓冲器
129
PP 命令
114
GPU
多处理器
120
122
124
110
132
渲染目标
程序
存储器
接口
扫描输出
引擎
显示器
134
后处理目
标程序
106
CPU
系统
存储器
用户输
入装置
系统盘
可拆卸存
储装置
102
104
108
107
109

图 1

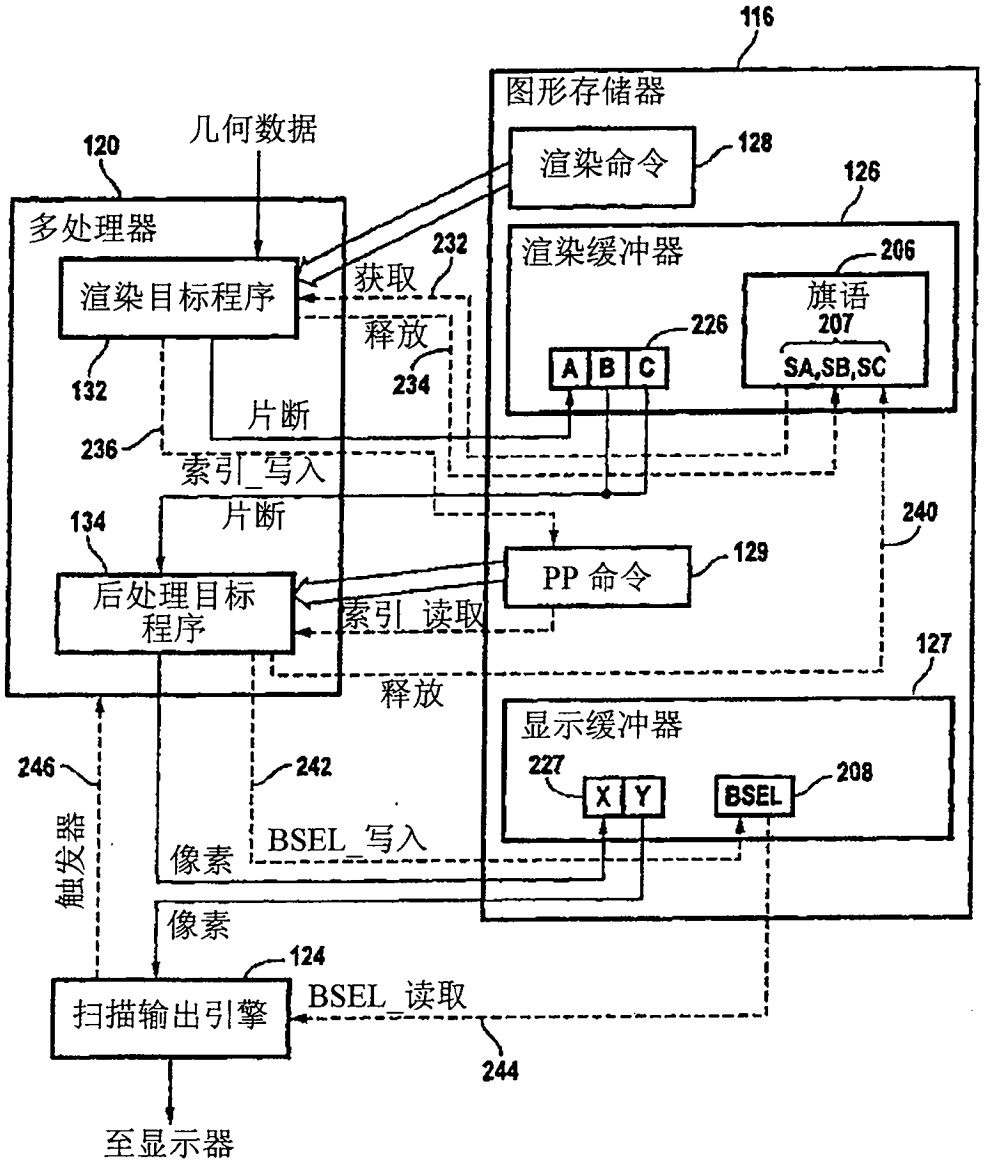

116
图形存储器
几何数据
120
多处理器
渲染命令
128
126
渲染缓冲器
206
旗语
207
SA,SB,SC
226
A B C
渲染目标程序
232
获取
释放
234
132
片断
236
索引_写入
片断
134
240
后处理目标程序
PP 命令
129
索引_读取
127
释放
显示缓冲器
227
X Y
BSEL
208
246
242
触发器
BSEL_写入
像素
像素
124
扫描输出引擎
BSEL_读取
244
至显示器

图 2

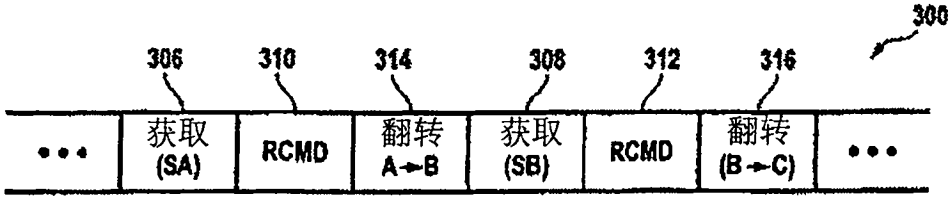

300
306
310
314
308
312
316
• • •
获取 (SA)
RCMD
翻转 A→B
获取 (SB)
RCMD
翻转 (B→C)
• • •

图 3A

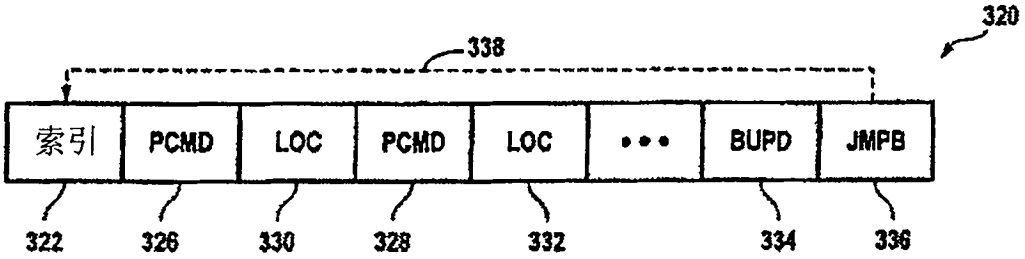

320
338
索引
PCMD
LOC
PCMD
LOC
• • •
BUPD
JMPB
322
326
330
328
332
334
336

图 3B

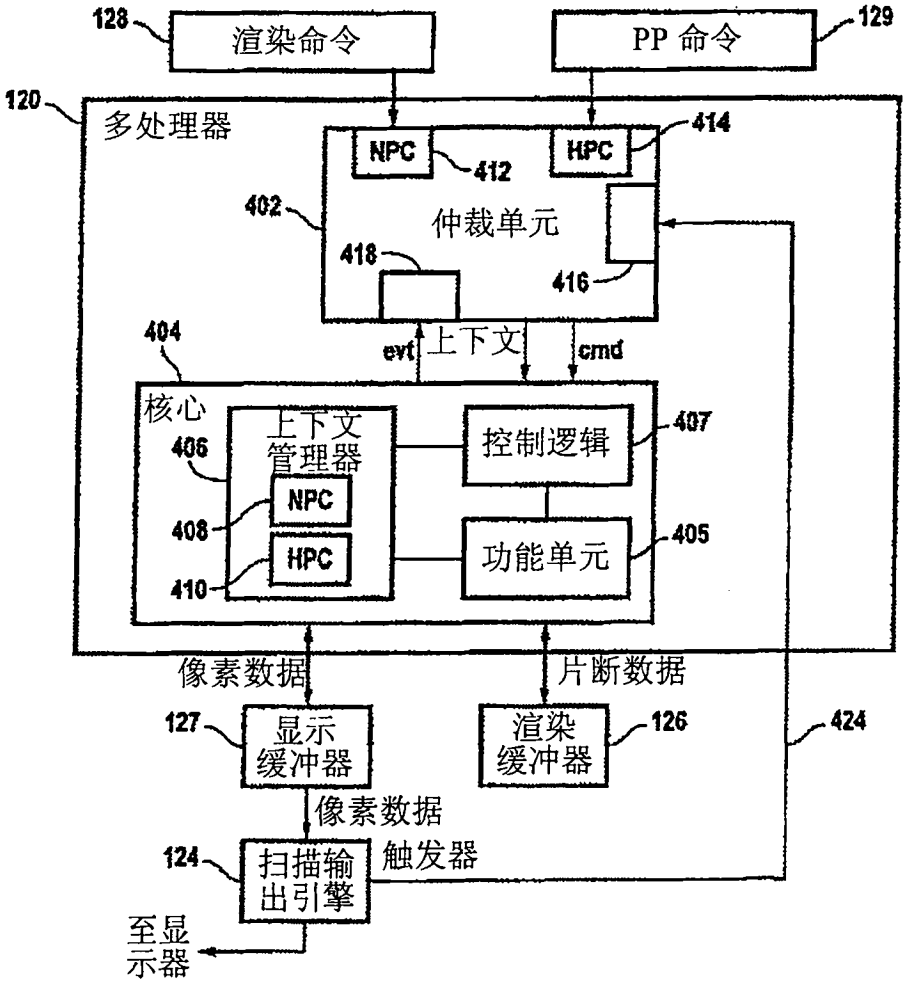

128
渲染命令
PP 命令
129
120
多处理器
NPC
412
HPC
414
402
仲裁单元
418
416
404
evt
上下文
cmd
核心
上下文管理器
406
NPC
408
HPC
410
控制逻辑
407
功能单元
405
像素数据
片断数据
127
显示缓冲器
渲染缓冲器
126
424
像素数据
124
扫描输出引擎
触发器
至显示器

图 4

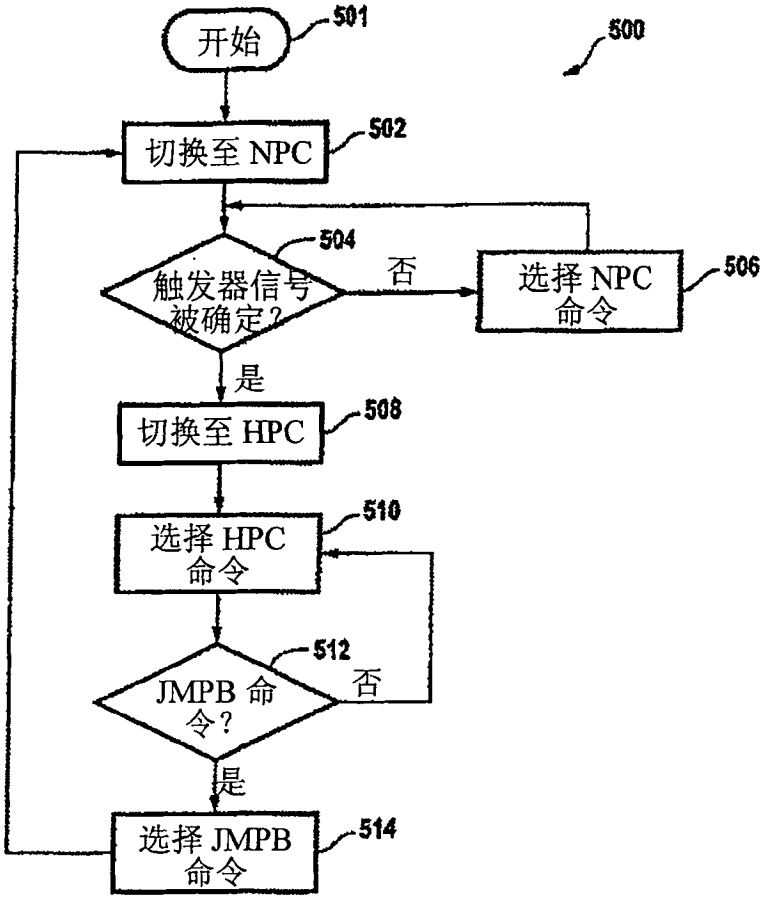

开始
501
500
切换至 NPC
502
触发器信号被确定?
504
否
选择 NPC 命令
506
是
切换至 HPC
508
选择 HPC 命令
510
JMPB 命令?
512
否
是
选择 JMPB 命令
514

图 5

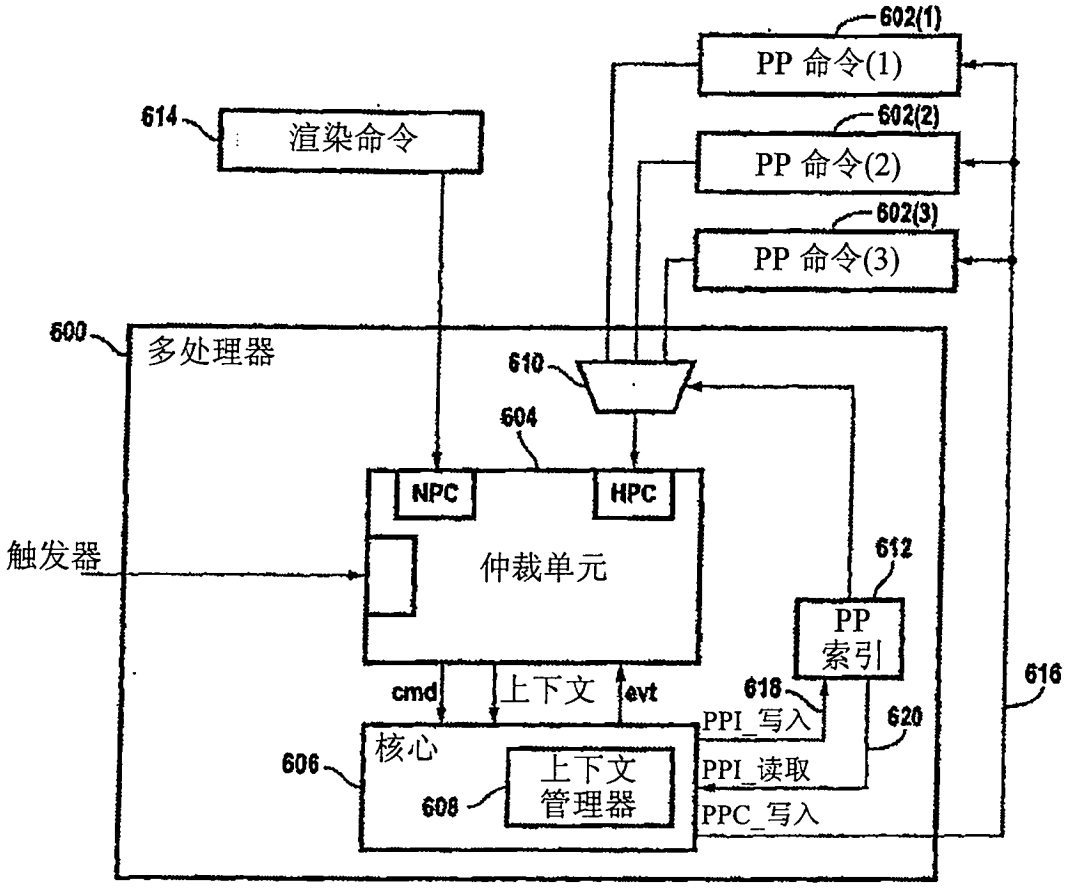

602(1)
PP 命令(1)
614
渲染命令
602(2)
PP 命令(2)
602(3)
PP 命令(3)
600
多处理器
610
604
NPC
HPC
触发器
仲裁单元
612
PP
索引
616
cmd
上下文
evt
618
PPI_写入
620
核心
606
上下文
管理器
608
PPI_读取
PPC_写入

图 6

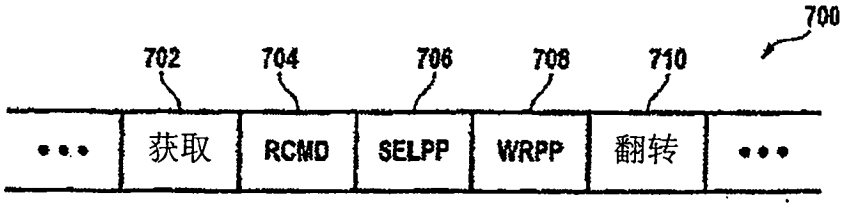

700
702
704
706
708
710
• • •
获取
RCMD
SELPP
WRPP
翻转
• • •

图 7

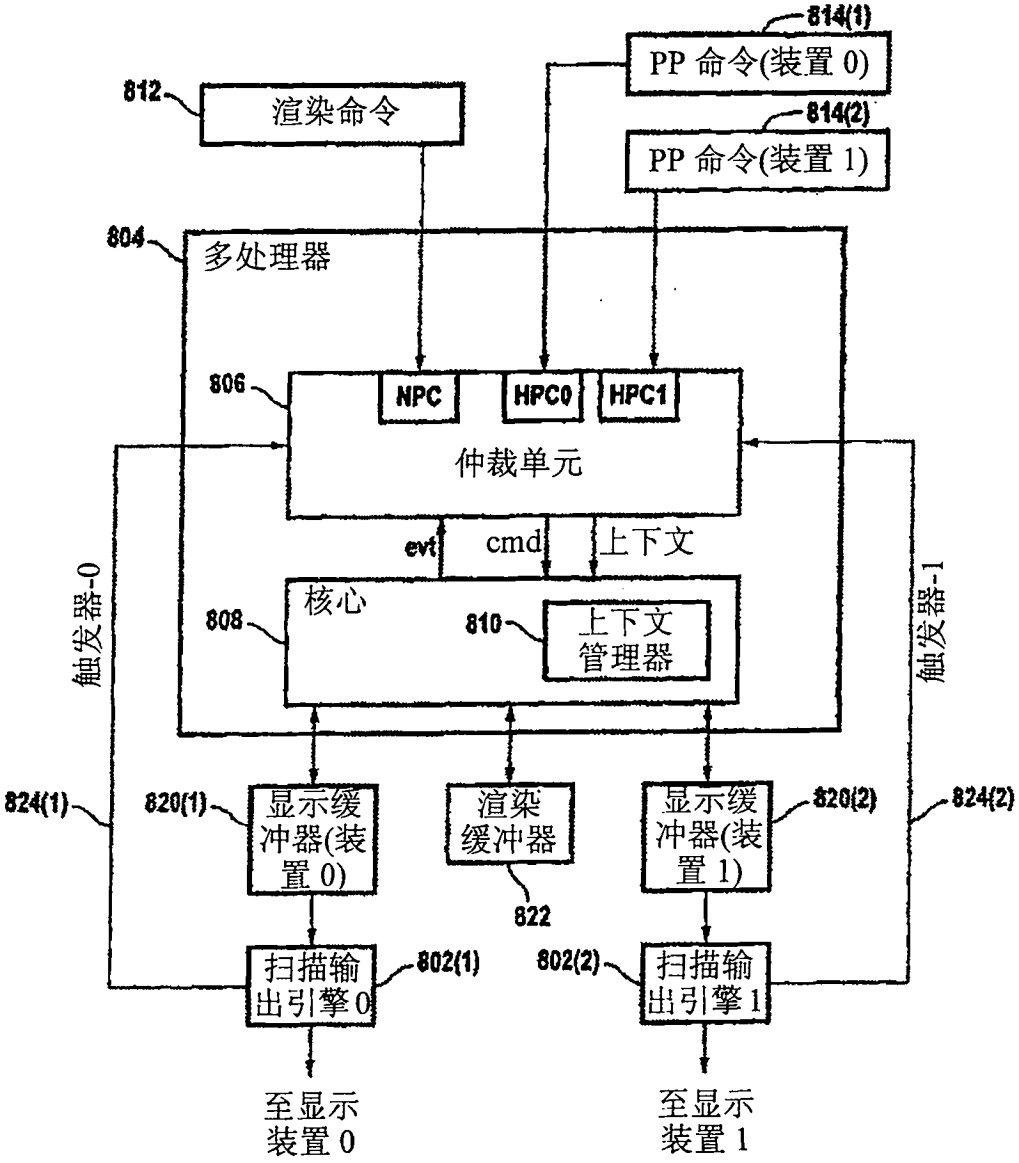

814(1)
PP 命令(装置 0)
812
渲染命令
814(2)
PP 命令(装置 1)
804
多处理器
806
NPC
HPC0
HPC1
仲裁单元
evt
cmd
上下文
触发器-0
808
核心
810
上下文管理器
触发器-1
824(1)
820(1)
显示缓冲器(装置 0)
渲染缓冲器
822
显示缓冲器(装置 1)
820(2)
824(2)
扫描输出引擎 0
802(1)
802(2)
扫描输出引擎 1
至显示装置 0
至显示装置 1

图 8

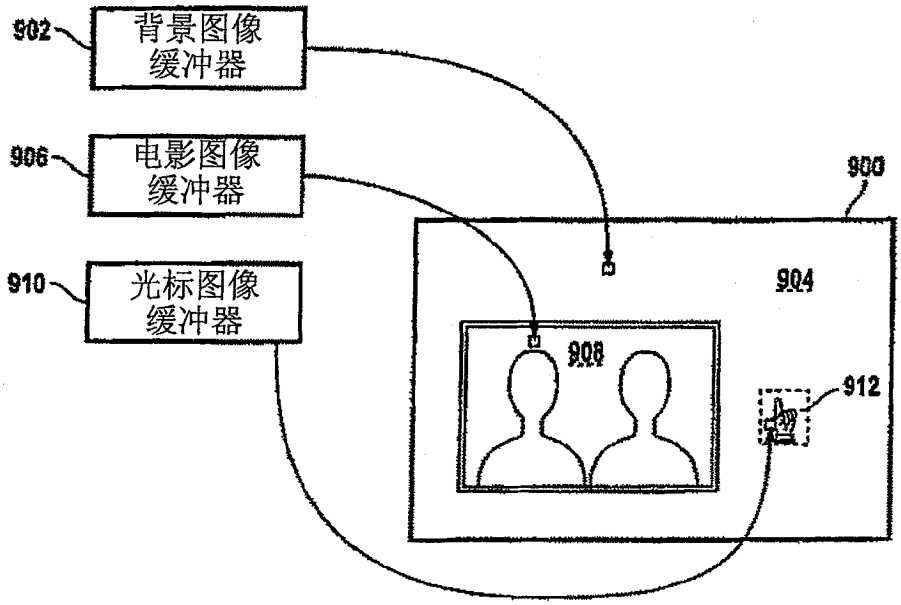

902
背景图像
缓冲器
906
电影图像
缓冲器
910
光标图像
缓冲器
900
904
908
912

图 9

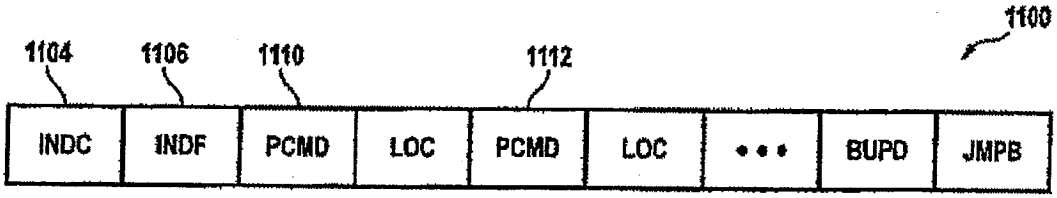

1100
1104
1106
1110
1112
INDC
INDF
PCMD
LOC
PCMD
LOC
• • •
BUPD
JMPB

图 11

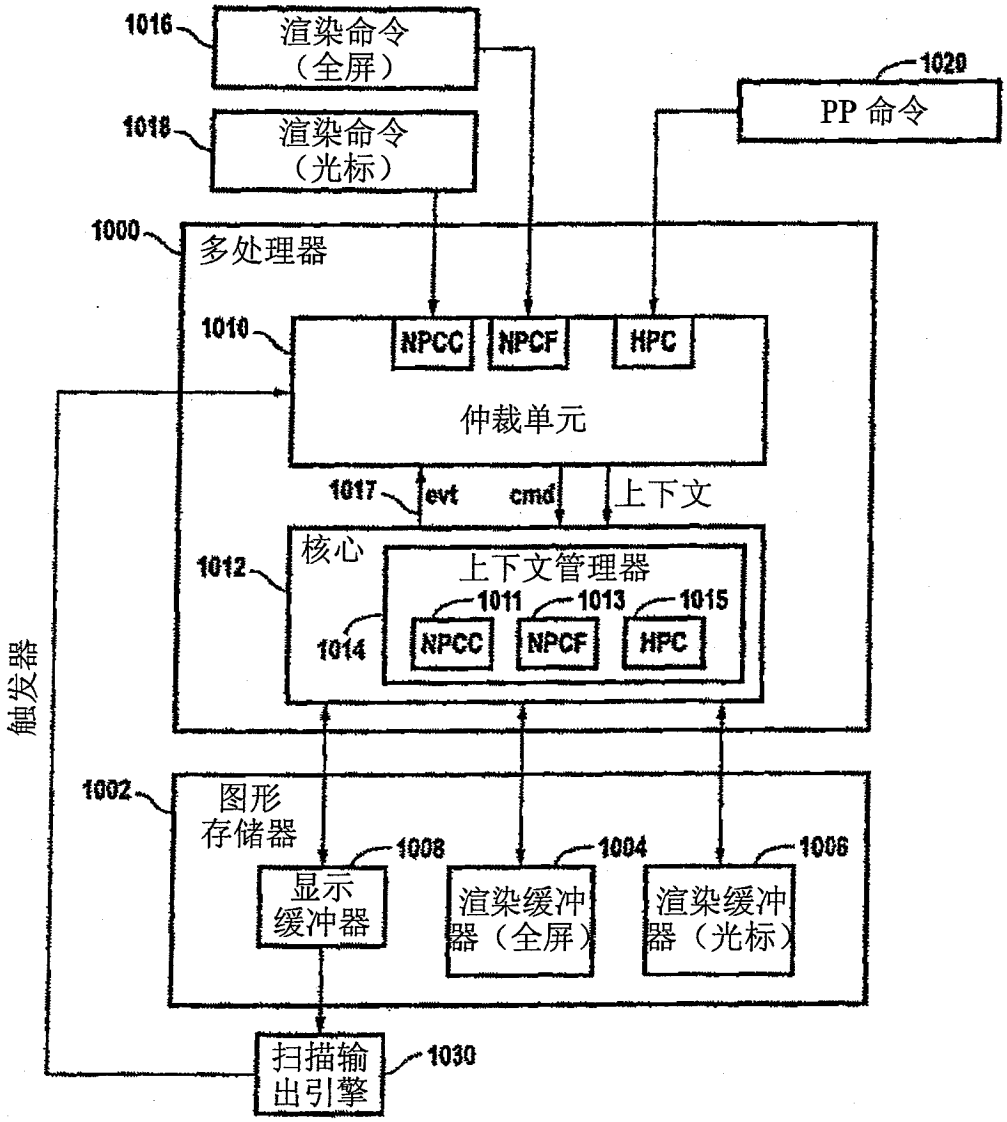

1016
渲染命令（全屏）
1018
渲染命令（光标）
1020
PP 命令
1000
多处理器
1010
NPCC
NPCF
HPC
仲裁单元
1017
evt
cmd
上下文
核心
1012
上下文管理器
1011
1013
1015
1014
NPCC
NPCF
HPC
触发器
1002
图形存储器
1008
1004
1006
显示缓冲器
渲染缓冲器（全屏）
渲染缓冲器（光标）
扫描输出引擎
1030

图 10

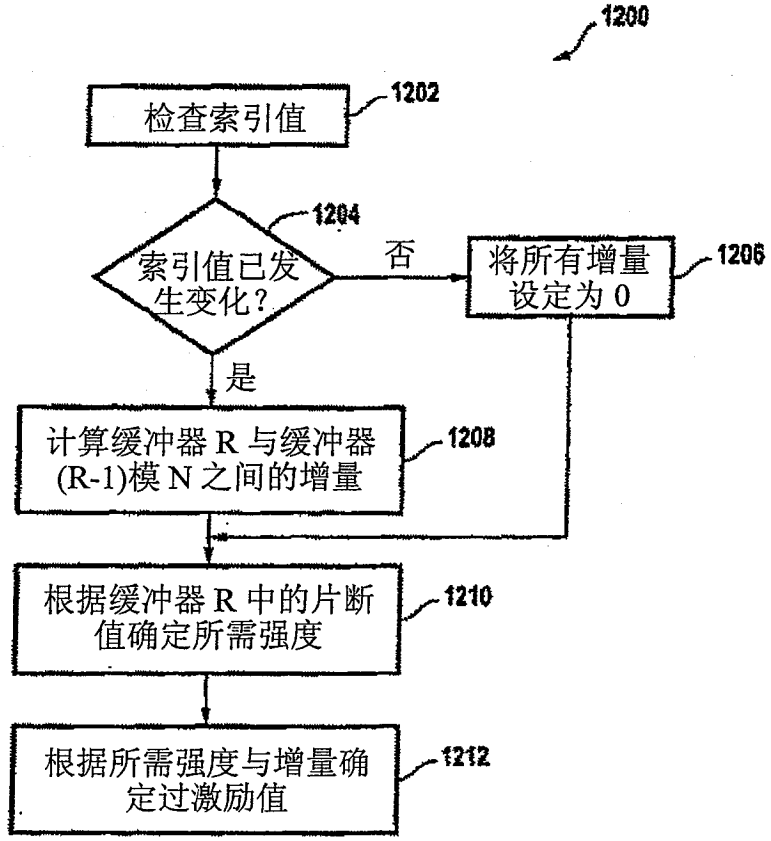

1200
检查索引值
1202
索引值已发生变化?
1204
否
将所有增量设定为 0
1206
是
计算缓冲器 R 与缓冲器 (R-1)模 N 之间的增量
1208
根据缓冲器 R 中的片断值确定所需强度
1210
根据所需强度与增量确定过激励值
1212

图 12

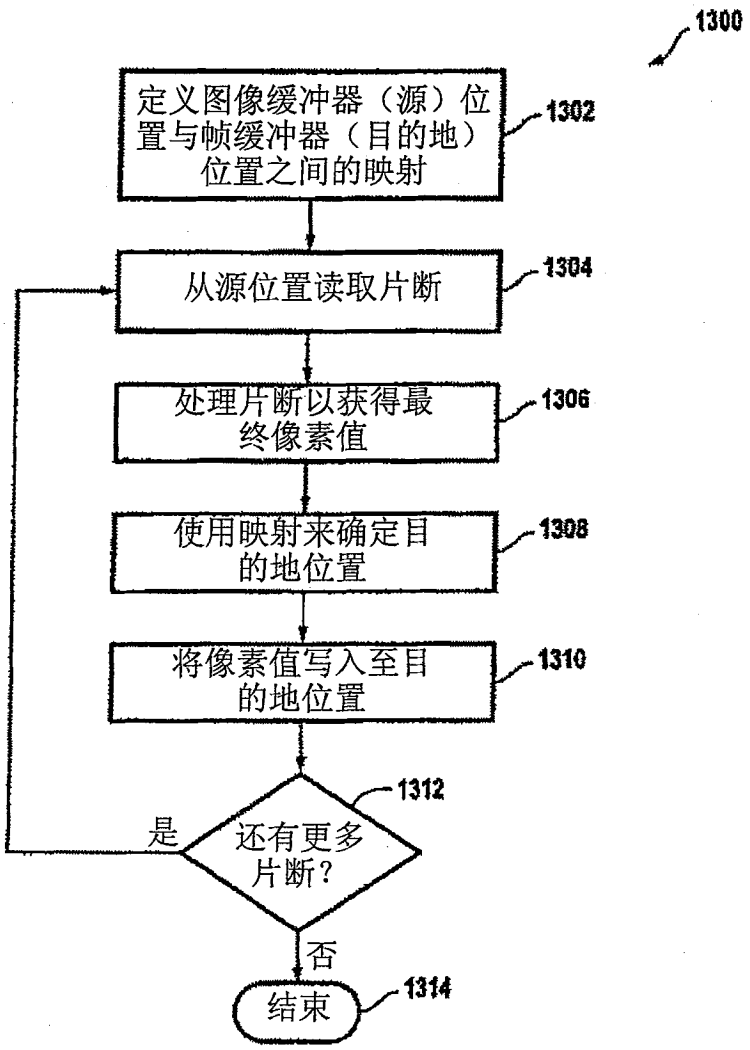

1300
定义图像缓冲器（源）位置与帧缓冲器（目的地）位置之间的映射
1302
从源位置读取片断
1304
处理片断以获得最终像素值
1306
使用映射来确定目的地位置
1308
将像素值写入至目的地位置
1310
还有更多片断？
1312
是
否
结束
1314

图 13