(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 111080505 A
(43)申请公布日 2020.04.28

(21)申请号 201911383330.2

(22)申请日 2019.12.27

(71)申请人 西安芯瞳半导体技术有限公司
地址 710065 陕西省西安市高新区丈八街办丈八一路3号旺都1幢2单元11层21101号

(72)发明人 樊良辉 陈成 张竞丹 李洋

(74)专利代理机构 西安维英格知识产权代理事务所(普通合伙) 61253
代理人 李斌栋 沈寒酉

(51)Int.Cl.
G06T 1/00(2006.01)
G06T 1/20(2006.01)

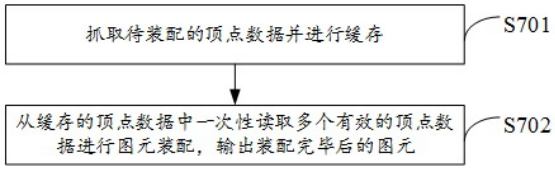
权利要求书2页 说明书13页 附图6页

(54)发明名称

一种提高图元装配效率的方法、装置及计算机存储介质

(57)摘要

本发明实施例公开了一种提高图元装配效率的方法、装置及计算机存储介质;该方法可以包括:抓取待装配的顶点数据并进行缓存;从缓存的顶点数据中一次性读取多个有效的顶点数据进行图元装配,输出装配完毕后的图元。



CN 111080505 A

1.一种提高图元装配效率的方法,其特征在于,所述方法包括:

抓取待装配的顶点数据并进行缓存;

从缓存的顶点数据中一次性读取多个有效的顶点数据进行图元装配,输出装配完毕后的图元。

2.根据权利要求1所述的方法,其特征在于,所述抓取待装配的顶点数据并进行缓存,包括:

通过顶点抓取模块根据命令处理器所下发的调度指令从显存中读取待装配的顶点数据,并将所述待装配的顶点数据存放于顶点数据缓存。

3.根据权利要求1所述的方法,其特征在于,所述从缓存的顶点数据中一次性读取多个有效的顶点数据进行图元装配,输出装配完毕后的图元,包括:

通过所述顶点发送模块对所述顶点数据缓存中的顶点数据进行读取,向图元装配模块一次性发送多个有效的顶点数据;

所述图元装配模块一次性接收所述顶点发送模块所发送的多个有效的顶点数据,并根据图元类型对接收到的所述多个有效的顶点数据进行图元装配,输出所述装配完毕后的图元。

4.根据权利要求3所述的方法,其特征在于,所述通过所述顶点发送模块对所述顶点数据缓存中的顶点数据进行读取,向图元装配模块一次性发送多个有效的顶点数据,包括:

所述顶点发送模块根据图元类型设置向所述图元装配模块发送顶点的发送窗口位置和移动步长;

所述顶点发送模块根据所述发送窗口位置和移动步长从所述顶点数据缓存中读取多个有效的顶点数据,并向所述图元装配模块发送所述多个读取的有效的顶点数据。

5.根据权利要求3所述的方法,其特征在于,所述方法还包括:

通过所述顶点发送模块根据顶点数据对应的顶点信息和/或图元配置状态信息对所述顶点数据缓存中的顶点数据进行筛选,以舍弃不参与图元装配的顶点数据。

6.根据权利要求3所述的方法,其特征在于,所述图元装配模块为一多路选择器;相应地,所述图元装配模块一次性接收所述顶点发送模块所发送的多个有效的顶点数据,并根据图元类型对接收到的所述多个有效的顶点数据进行图元装配,输出所述装配完毕后的图元,包括:

所述图元装配模块一次性地接收3个顶点数据作为输入进行图元装配,并将接收到的3个顶点数据根据图元类型并传递至相应的输出端口。

7.根据权利要求1至6任一项所述的方法,其特征在于,所述方法还包括:

通过所述图元装配模块将输出的装配完毕后的图元缓存至图元数据缓存;

通过包围盒生成模块根据图元类型,从所述图元数据缓存中读取图元数据并对装配完毕的图元数据生成包围盒。

8.一种图元装配装置,其特征在于,所述装置包括:顶点抓取模块、顶点数据缓存,顶点发送模块和图元装配模块;

其中,所述顶点抓取模块,经配置以针对待装配的顶点数据进行抓取,并将已抓取的顶点数据存放于所述顶点数据缓存;

所述顶点发送模块,经配置以在对所述顶点数据缓存中的顶点数据进行读取后,向所

述图元装配模块一次性发送多个有效的顶点数据；

所述图元装配模块，经配置以一次性接收所述顶点发送模块所发送的多个有效的顶点数据，并根据图元类型对接收到的多个有效的顶点数据进行图元装配，输出装配完毕后的图元。

9.根据权利要求8所述的装置，其特征在于，所述顶点发送模块，经配置以：

根据图元类型设置向所述图元装配模块发送顶点的发送窗口位置和移动步长；

根据所述发送窗口位置和移动步长从所述顶点数据缓存中读取多个有效的顶点数据，并向所述图元装配模块发送所述多个读取的有效的顶点数据。

10.根据权利要求8所述的装置，其特征在于，所述图元装配模块为一多路选择器；相应地，所述图元装配模块经配置以：

一次性地接收3个顶点数据作为输入进行图元装配，并将接收到的3个顶点数据根据图元类型并传递至相应的输出端口。

11.根据权利要求8至10任一项所述的装置，其特征在于，所述装置还包括：图元数据缓存和包围盒生成模块；其中，

所述图元数据缓存，经配置以缓存所述图元装配模块输出的装配完毕后的图元；

所述包围盒生成模块，经配置以根据图元类型，从所述图元数据缓存中读取图元数据并对装配完毕的图元数据生成包围盒。

12.一种GPU，其特征在于，所述GPU包括权利要求8至11任一项所述的图元装配装置。

13.一种计算机存储介质，所述计算机存储介质存储有提高图元装配效率的程序，所述提高图元装配效率的的程序被至少一个处理器执行时实现权利要求1至7任一项所述的提高图元装配效率的方法的步骤。

一种提高图元装配效率的方法、装置及计算机存储介质

技术领域

[0001] 本发明实施例涉及图形处理器(GPU,Graphics Processing Unit)技术领域,尤其涉及一种提高图元装配效率的方法、装置及计算机存储介质。

背景技术

[0002] 常规的图形渲染管线中通常包括图元装配单元,其被配置为将顶点着色器所输出的所有顶点作为输入,根据顶点的原始连接关系还原出图形的网格结构,从而转换为图形的图元并输出。

[0003] 而当前图元装配单元在进行图元装配的过程中,其实现方式为每次仅输入一个顶点,但在满足预定条件后才输出装配完成的图元。由此导致输入顶点的数据量与输出图元的数据量不匹配,随着目前GPU计算能力的不断提升,上述不匹配成为了限制图形渲染管线性能提升的瓶颈之一。

发明内容

[0004] 有鉴于此,本发明实施例期望提供一种提高图元装配效率的方法、装置及计算机存储介质;能够使得图元装配单元在进行图元装配过程中实现输入顶点的数据量与输出图元的数据量之间的匹配,减少图元装配单元再转配前后的数据延时,提高图形渲染管线的性能。

[0005] 本发明实施例的技术方案是这样实现的:

第一方面,本发明实施例提供了一种提高图元装配效率的方法,所述方法包括:

抓取待装配的顶点数据并进行缓存;

从缓存的顶点数据中一次性读取多个有效的顶点数据进行图元装配,输出装配完毕后的图元。

[0006] 第二方面,本发明实施例提供了一种图元装配装置,所述装置包括:顶点抓取模块、顶点数据缓存,顶点发送模块和图元装配模块;

其中,所述顶点抓取模块,经配置以针对待装配的顶点数据进行抓取,并将已抓取的顶点数据存放于所述顶点数据缓存;

所述顶点发送模块,经配置以在对所述顶点数据缓存中的顶点数据进行读取后,向所述图元装配模块一次性发送多个有效的顶点数据;

所述图元装配模块,经配置以一次性接收所述顶点发送模块所发送的多个有效的顶点数据,并根据图元类型对接收到的多个有效的顶点数据进行图元装配,输出装配完毕后的图元。

[0007] 第三方面,本发明实施例提供了一种GPU,所述GPU包括第二方面所述的图元装配装置。

[0008] 第四方面,本发明实施例提供了一种计算机存储介质,所述计算机存储介质存储有提高图元装配效率的的程序,所述提高图元装配效率的的程序被至少一个处理器执行时

实现第一方面所述的提高图元装配效率的方法的步骤。

[0009] 本发明实施例提供了一种提高图元装配效率的方法、装置及计算机存储介质;从抓取并缓存的顶点数据中一次性读取多个有效的顶点数据进行图元装配,并输出装配完毕后的图元。相比于传统的类似状态机的形式进行图元装配,极大地提高了图元装配的效率。

附图说明

[0010] 图1为可实施本发明实施例一个或多个方面的计算装置的框图。

[0011] 图2为说明图1中处理器、GPU和系统存储器的实例实施方案的框图。

[0012] 图3为更加详细地说明由图2的GPU结构所形成的图形处理管线的框图。

[0013] 图4为图形处理管线中图元装配级的组成框图。

[0014] 图5为本发明实施例提供的FIFO结构的顶点数据缓存结构示意图。

[0015] 图6为本发明实施例提供的状态机示意图。

[0016] 图7为本发明实施例提供的一种提高图元装配效率的方法流程示意图。

具体实施方式

[0017] 下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行清楚、完整地描述。

[0018] 一般来说,GPU在对图形或图像进行渲染的过程中,均需要经过图元装配过程,即在获得顶点之后,将顶点按照原始连接关系还原出图形的网格结构,即图元,但目前在图元装配的过程中,每次仅能够输入一个顶点,在满足预定条件后才输出装配完成的图元,从而导致输入顶点的数据量与输出图元的数据量不匹配。而本发明实施例期望描述一种用于进行高效图元装配的技术,举例来说,在图元装配过程中,通过提升输入顶点数目的方式来提高图元装配效率的技术。

[0019] 图1为本发明实施例提供的一种能够实施用于进行高效图元装配技术的计算装置1,该计算装置1的实例包括但不限于:无线装置、移动或蜂窝电话(包含所谓的智能电话)、个人数字助理(PDA)、视频游戏控制台(包含视频显示器、移动视频游戏装置、移动视频会议单元)、膝上型计算机、桌上型计算机、电视机顶盒、平板计算装置、电子书阅读器、固定或移动媒体播放器,等。在图1的实例中,该计算装置2可以包括:处理器6、系统存储器10和GPU 12。计算装置2还可包含显示处理器14、收发器模块3、用户接口4和显示器8。收发器模块3和显示处理器14两者可为与处理器6和/或GPU 12相同的集成电路(IC)的部分,两者可在包含处理器6和/或GPU 12的一或多个IC的外部,或可形成于在包含处理器6和/或GPU 12的IC外部的IC中。

[0020] 为清楚起见,计算装置2可包含图1中未图示的额外模块或单元。举例来说,计算装置2可在其中计算装置2为移动无线电话或的实例中包含扬声器和麦克风(两者均未在图1中示出)来实现电话通信,或在计算装置2为媒体播放器的情况下包含扬声器。计算装置2还可包含摄像机。此外,计算装置2中所示的各种模块和单元可能不是在计算装置2的每个实例中都是必需的。举例来说,在计算装置2为桌上型计算机或经装备以与外部用户接口或显示器介接的其它装置的实例中,用户接口4和显示器8可在计算装置2外部。

[0021] 用户接口4的实例包含(但不限于)轨迹球、鼠标、键盘和其它类型的输入装置。用

户接口4还可为触摸屏，并且可作为显示器8的部分并入。收发器模块3可包含电路以允许计算装置2与另一装置或网络之间的无线或有线通信。收发器模块3可包含调制器、解调器、放大器和用于有线或无线通信的其它此类电路。

[0022] 处理器6可为微处理器，例如中央处理单元(CPU)，其经配置以处理供执行的计算机程序的指令。处理器6可包括控制计算装置2的运算的通用或专用处理器。用户可将输入提供到计算装置2，以致使处理器6执行一或多个软件应用程序。在处理器6上执行的软件应用程序可包含(例如)操作系统、文字处理器应用程序、电子邮件应用程序、电子表格应用程序、媒体播放器应用程序、视频游戏应用程序、图形用户接口应用程序或另一程序。另外，处理器6可执行用于控制GPU 12的运算的GPU驱动程序22。用户可经由一或多个输入装置(未图示)(例如，键盘、鼠标、麦克风、触摸垫或经由用户输入接口4耦合到计算装置2的另一输入装置)将输入提供到计算装置2。

[0023] 在处理器6上执行的软件应用程序可包含一或多个图形渲染指令，其指令处理器6来致使将图形数据渲染到显示器8。在一些实例中，所述软件指令可符合图形应用程序编程接口(API)，例如开放式图形库API、开放式图形库嵌入系统(OpenGL ES)API、Direct3D API、X3D API、RenderMan API、WebGL API、开放式计算语言(OpenCLT M)、RenderScript或任何其它异构计算API，或任何其它公用或专有标准图形或计算API。所述软件指令还可为针对无渲染算法(例如计算摄影、卷积神经网络、视频处理、科学应用程序等)的指令。为了处理图形渲染指令，处理器6可向GPU 12发出一或多个图形渲染命令发(例如，通过GPU驱动程序22)，以致使GPU 12执行图形数据的渲染中的一些或全部。在一些实例中，待渲染的图形数据可包含例如点、线、三角形、四边形、三角形带等图形图元的列表。

[0024] GPU 12可经配置以执行图形运算，从而将一或多个图形图元渲染到显示器8。因此，当在处理器6上执行的软件应用中的一者需要图形处理时，处理器6可将图形命令和图形数据提供到GPU 12以用于渲染到显示器8。图形数据可包含(例如)绘制命令、状态信息、图元信息、纹理信息等。在一些情况下，GPU 12可内置有高度并行结构，其提供比处理器6高效的对复杂图形相关运算的处理。举例来说，GPU 12可包含经配置来以并行方式对多个顶点或像素进行运算的多个处理元件，例如着色器单元。在一些情况下，GPU 12的高度并行性质允许GPU 12比使用处理器6直接将场景绘制到显示器8更快速地将图形图像(例如，GUI和二维(2D)和/或三维(3D)图形场景)绘制到显示器8上。

[0025] 在一些情况下，可将GPU 12集成到计算装置2的母板中。在其它情况下，GPU 12可存在于图形卡上，所述图形卡安装在计算装置2的母板中的端口中，或可以其它方式并入在经配置以与计算装置2互操作的外围装置内。GPU 12可包含一或多个处理器，例如一或多个微处理器、专用集成电路(ASIC)、现场可编程门阵列(FPGA)、数字信号处理器(DSP)或其它等效的集成或离散逻辑电路。GPU 12还可包含一或多个处理器核心，使得GPU 12可被称作多核处理器。

[0026] 图形存储器40可为GPU 12的一部分。因此，GPU 12可在不使用总线的情况下从图形存储器40读取数据且将数据写入到图形存储器40。换句话说，GPU 12可使用本地存储装置而不是芯片外存储器在本地处理数据。此类图形存储器40可被称作芯片上存储器。这允许GPU 12通过消除GPU 12经由总线读取和写入数据的需要来以更高效的方式操作，其中经由总线操作可经历繁重的总线业务。然而，在一些情况下，GPU 12可不包含单独的存储器，

而是经由总线利用系统存储器10。图形存储器40可包含一或多个易失性或非易失性存储器或存储装置，例如，随机存取存储器(RAM)、静态RAM(SRAM)、动态RAM(DRAM)、可擦除可编程ROM(EPROM)、电可擦除可编程ROM(EEPROM)、快闪存储器、磁性数据媒体或光学存储媒体。

[0027] 在一些实例中，GPU 12可将完全形成的图像存储在系统存储器10中。显示处理器14可从系统存储器10检索图像，且输出致使显示器8的像素照亮以显示所述图像的值。显示器8可为计算装置2的显示器，其显示由GPU 12产生的图像内容。显示器8可为液晶显示器(LCD)、有机发光二极管显示器(OLED)、阴极射线管(CRT)显示器、等离子显示器或另一类型的显示装置。

[0028] 图2是进一步详细说明图1中处理器6、GPU 12和系统存储器10的实例实施方案的框图。如图2所示，处理器6可执行至少一个软件应用程序18、图形API 20和GPU 驱动程序22，其中的每一者可为一或多个软件应用程序或服务。在一些实例中，图形API 20和GPU驱动程序22可实施为CPU 6的硬件单元。

[0029] 可供处理器6和GPU 12使用的存储器可包含系统存储器10和输出缓冲器16。输出缓冲器16可为系统存储器10的部分或可与系统存储器10分离。输出缓冲器16可存储经渲染图像数据，例如像素数据，以及任何其它数据。输出缓冲器16还可被称为帧缓冲器或显存。

[0030] 图形存储器40可包含片上存储装置或存储器，其物理上集成到GPU12的集成电路芯片中。如果图形存储器40是在芯片上，那么与经由系统总线从系统存储器10读取值或将值写入到系统存储器10相比，GPU12能够更加快速地从图形存储器40读取值或将值写入到图形存储器40。

[0031] 输出缓冲器16存储GPU 12的目的地像素。每个目的地像素可与唯一屏幕像素位置相关联。在一些实例中，输出缓冲器16可存储每个目的地像素的色彩分量和目的地α值。举例来说，输出缓冲器16可存储每个像素的红色、绿色、蓝色、α(RGBA)分量，其中“RGB”分量对应于色彩值，并且“A”分量对应于目的地α值(例如，用于图像合成的不透明度值)。尽管将输出缓冲器16和系统存储器10说明为单独的存储器单元，但在其它实例中，输出缓冲器16可以是系统存储器10的一部分。此外，输出缓冲器16还可能够存储除像素之外的任何合适的数据。

[0032] 软件应用程序18可为利用GPU 12的功能性的任何应用程序。举例来说，软件应用程序18可为GUI应用程序、操作系统、便携式制图应用程序、用于工程或艺术应用的计算机辅助设计程序、视频游戏应用程序或使用2D或3D图形的另一类型的软件应用程序。

[0033] 软件应用程序18可包含指令GPU 12渲染图形用户接口(GUI)和/或图形场景的一或多个绘制指令。举例来说，绘制指令可包含界定将由GPU 12渲染的一组一或多个图形图元的指令。在一些实例中，绘制指令可共同地界定用于GUI中的多个开窗表面的全部或部分。在额外实例中，所述绘制指令可共同地定义图形场景的全部或部分，所述图形场景包含在由应用程序定义的模型空间或世界空间内的一或多个图形对象。

[0034] 软件应用程序18可经由图形API 20调用GPU驱动程序22，以向GPU 12发出一或多个命令，以用于将一或多个图形图元渲染到可显示的图形图像中。举例来说，软件应用程序18可调用GPU驱动程序22，以向GPU 12提供图元定义。在一些情况下，图元定义可以例如三角形、矩形、三角形扇、三角形带等的绘制图元的列表的形式被提供到GPU 12。图元定义可包含指定与待呈现的图元相关联的一或多个顶点的顶点规格。所述顶点规格可包含每个顶

点的位置坐标，且在一些情况下包含与顶点相关联的其它属性，例如色彩属性、法向量和纹理坐标。图元定义还可包含图元类型信息(例如，三角形、矩形、三角形扇、三角形带等)、缩放信息、旋转信息及类似者。

[0035] 基于由软件应用程序18向GPU驱动程序22发出的指令，GPU驱动程序22可调配指定供GPU 12执行的一或多个运算以便渲染图元的一或多个命令。当GPU 12接收到来自CPU 6的命令时，GPU 12可使用处理器集群46执行图形处理管线，以便对命令进行解码，并对图形处理管线进行配置以执行命令中所制定的操作。

[0036] 处理器集群46可包含一或多个可编程处理单元24和/或一或多个固定功能处理单元26。对于上述两种处理单元，可编程处理单元24可包含例如被配置成执行从CPU 6下载到GPU 12上的一或多个着色器程序的可编程着色器单元。在一些实例中，可编程着色器单元可被称为“着色器处理器”或“统一着色器”，且可被配置为能够至少执行顶点和片元着色操作以呈现图形；可选地，可编程着色器单元还可以被配置执行几何或其它着色操作以呈现图形。因此，处理器集群46中的可编程着色器单元可至少包含顶点着色器单元、片元着色器单元，此外，还可以包含几何着色器单元、外壳着色器单元、域着色器单元、计算着色器单元和/或统一着色器单元。在具体实施过程中，可编程着色器单元可各自包含用于提取和解码操作的一或多个组件、用于进行算术计算的一或多个ALU、一或多个存储器、高速缓存和寄存器。

[0037] 并且，固定功能处理单元26可包含经硬连线以执行某些功能的硬件。尽管固定功能硬件可经由例如一或多个控制信号而配置以执行不同功能，但所述固定功能硬件通常并不包含能够接收用户编译程序的程序存储器。在一些实例中，处理器集群46中的固定功能处理单元26可包含例如执行图元装配的处理单元、执行光栅操作的处理单元，所述光栅操作例如深度测试、剪刀测试、α掺合等。对于执行图元装配的处理单元来说，其能够将通过顶点着色器单元已完成着色的顶点按照原始连接关系还原出图形的网格结构，即图元，从而供后续片元着色器单元进行处理。

[0038] 一般来说，GPU 12从CPU 6所接收到的命令，其示例为，处理器6执行GPU驱动程序，以使得GPU驱动程序22可基于由软件应用程序18向GPU驱动程序22发出的指令产生定义用于由GPU 12执行的操作集合的命令流。所述命令流能够控制处理器集群46中可编程处理单元24和固定功能处理单元26的操作。举例来说，命令流通常控制GPU 12 的哪些组件执行命令流中界定的操作。

[0039] 如上所述，GPU 12可包含可从GPU驱动程序22接收命令流的命令处理器30。命令处理器30可以是被配置成接收并处理一或多个命令流的硬件与软件的任意组合。由此，命令处理器30可在本地控制GPU 资源而无需处理器6 的干预。举例来说，GPU 12的命令处理器30可从处理器6接收一个或一个以上“任务”。命令处理器30可独立地调度所述任务由GPU 12的资源(比如一或多个可编程处理单元24和一或多个固定功能处理单元26)执行。在一个实例中，命令处理器30可以是硬件处理器。在图2中所示出的实例中，命令处理器30可包含于GPU 12中。在其它实例中，命令处理器30可以是与CPU 6和GPU 12分离的单元。命令处理器30还可被称为流处理器、命令/流处理器及类似者，以指示其可以是被配置成接收命令和/或操作的流的任何处理器。

[0040] 命令处理器30可处理一或多个命令流，其包含调度操作，所述调度操作包含于由

GPU 12执行的一或多个命令流中。具体地说，命令处理器30可处理一或多个命令流，且调度所述一或多个命令流中的操作，以由处理器集群46执行。在操作中，GPU驱动程序22可向命令处理器30发送包括待由GPU 12执行的一系列操作的命令流。命令处理器30可接收包括命令流的操作流且可基于命令流中的操作次序依序地处理命令流的操作，且可调度命令流中的操作可以由处理器集群46中的一或多个处理单元执行。

[0041] 图3是进一步以图2中所示的GPU结构中由处理器集群46所形成的图形处理管线80示例，需要说明的是，图形处理管线80是利用处理器集群46中所包含的可编程处理单元24和固定功能处理单元26通过级联形成的逻辑结构，相应来说，图形处理管线80通常包含可编程级(如图3中圆角框示意)和固定功能级(如图3中方框示意)，举例来说，可编程级可由处理器集群46中的可编程处理单元24来执行，固定功能级可由处理器集群46中的固定功能处理单元26实现。如图3所示，图形处理管线80所包括的各级依次为：

输入汇编器级82，在图3的实例中展示为固定功能级且通常负责将图形数据(三角形、线和点)供应到图形处理管线80。举例来说，输入汇编器级82可收集高阶表面、图元等的顶点数据，且将顶点数据和属性输出到顶点着色器级84。

[0042] 顶点着色器级84，在图3中展示为可编程级，且负责处理所接收顶点数据和属性，并通过每次针对每个顶点实施一组操作来处理顶点数据。。

[0043] 图元装配级86，在图3中展示为固定功能级，负责收集顶点着色器级84输出的顶点并将所述顶点组成几何图元，以由几何着色器级88处理。例如，图元装配级86可以配置为将每三个连续的顶点组成为几何图元(即三角形)，以发送到几何着色器级88。在一些实施例中，特定的顶点可以被重复用于连续的几何图元(例如，三角形带中的两个连续的三角形可以共享两个顶点)。图元装配级86发送几何图元(即相关顶点的集合)到几何着色器级88。

[0044] 几何着色器级88，在图3中展示为可编程级，负责通过对几何图元实施一组操作(即几何着色程序或程序)来处理几何图元。细分曲面操作可以由每个几何图元生成一个或更多几何图元。换句话说，几何着色器级88可以将每个几何图元再分为更细网眼的两个或多个几何图元，用于被剩余的图形处理管线600处理。几何着色器级88发送几何图元到光栅器级90。

[0045] 光栅器级90通常为负责修剪和准备片元着色器级92的图元的固定功能级。举例来说，光栅器级90可产生若干碎片以供片元着色器级92进行阴影处理。

[0046] 片元着色器级92，在图3中展示为可编程级从光栅器级90接收碎片且产生例如颜色等每像素数据。片元着色器级92还可执行例如纹理混合和照明模型计算等每像素处理。

[0047] 输出合并器级94，在图3中展示为固定功能级，通常负责对像素数据实施多种操作，例如实施透明测试(alpha test)、模板测试(stencil test)、以及将像素数据与对应于与该像素相关的其他片段的其他像素数据混合。当输出合并器级94已经完成处理像素数据(即输出数据)时，可以将处理完成的像素数据写入到渲染目标，例如输出缓冲器16以产生最终结果。

[0048] 而本发明实施例期望能够通过提升输入顶点数目的方式来提高图元装配效率，基于此，参见图4，其为图3中所示的图元装配级86更详细的说明框图，在本发明实施例中，图元装配级86针对顶点数据进行装配过程不再是采用状态机的方式每次仅传入一个顶点，在满足设定条件后输出装配完成的图元；而是采用类似多路选择器的方式，一次传入多个(例

如三个)顶点数据,并针对该一次性传入的多个顶点数据执行图元装配。

[0049] 如图4所示,图元装配级86中可以包括顶点抓取模块8601和顶点数据缓存8602,顶点抓取模块8601可针对待装配的顶点数据进行抓取,并将已抓取的顶点数据存放于顶点数据缓存8602;

在一些示例中,顶点抓取模块8601可经配置为从显存中抓取待装配的顶点数据,需要说明的是,需要装配的顶点数据可以是经过图形处理管线80中的顶点着色器级84完成顶点着色之后的顶点数据;具体来说,顶点抓取模块8601可以根据命令处理器30所下发的调度指令从显存中读取待装配的顶点数据,举例来说,调度指令中可以包括待装配的顶点数据的顶点信息,至少包括以下一项或多项:顶点绘制模式、待装配的顶点数据在显存中的存储地址或索引、待装配的顶点数据格式、待装配的顶点数据数目等信息。

[0050] 在另一些示例中,顶点数据缓存8602可以优选为符合先入先出(FIFO)结构,并且在实施过程中,顶点数据缓存8602可以是高速缓冲存储器以作为图元装配级86的一部分,也可以为了节省片上面积和资源从图形存储器40中开辟一存储空间。由于目前总线位宽一般为64位,突发传输机制允许连续传输8次,即一次能够获得64×8的数据量;而一个顶点的数据量为128位,如此顶点抓取模块8602每次最多可以获得4个顶点数据,结合顶点抓取模块8602的耗时,顶点数据缓存8602可缓存的数据容量可设置为8。可以理解地,该数据容量与实际的硬件线宽和突发传输设计有关,当然可以根据实际的设计需要及相关技术的发展进行调整。

[0051] 图元装配级86还可包括顶点发送模块8603和图元装配模块8604;顶点发送模块8603可以在对顶点数据缓存8602中的顶点数据进行读取后,向图元装配模块8604一次性发送多个有效的顶点数据;而图元装配模块8604可以一次性接收顶点发送模块8603所发送的多个有效的顶点数据,并根据图元类型对接收到的多个有效的顶点数据进行图元装配,输出装配完毕后的图元。

[0052] 在一些示例中,顶点发送模块8603可经配置以根据图元类型设置向图元装配模块8604发送顶点的发送窗口位置和移动步长;以及通过根据发送窗口位置和移动步长从顶点数据缓存8602中读取多个有效的顶点数据,并向图元装配模块8604发送所述多个读取的有效的顶点数据。通常来说,由于目前对图元进行一次装配最多需要的顶点数为3个,所以为了达到最大的图元装配效率,发送窗口长度优选设置为三个顶点宽度,也就是说,顶点发送模块8603一次性向图元装配模块8604发送三个有效的顶点数据,而图元装配模块8604也就相应地一次性接收3个有效的顶点数据并进行图元装配。

[0053] 在本示例中,图元类型通常可以包括:GL_POINTS、GL_LINES、GL_LINE_STRIP、GL_LINE_LOOP、GL_TRIANGLE、GL_TRIANGLE_STRIP以及GL_TRIANGLE_FAN或GL_POLYGON;并且设定FIFO结构的顶点数据缓存8602中顶点数据如图5所示,其下标依次为0、1、2、3、4、5、6、7。那么针对上述七种图元类型,顶点发送模块8603设置发送窗口位置和移动步长,以及通过根据发送窗口位置和移动步长从顶点数据缓存8602中读取多个顶点数据,并向图元装配模块8604发送所述多个读取的顶点数据,具体可以实现为:

若图元类型为GL_POINTS,发送窗口的起始位置为顶点数据缓存8602中的前三个顶点数据;移动步长设置为3,即每次向后移动三个顶点。当设置完成之后,顶点发送模块8603根据顶点图元的信息判断三个顶点是否都为需要装配的顶点,即若发送窗口中的其中一个顶

点为图元重启顶点，则删除该顶点并按序取顶点数据缓存8602中的下一个顶点，并做相同的判断。若发送窗口移动到顶点数据缓存8602的最后并且包含无效顶点，则标记无效顶点位置，以供图元装配模块8604使用。以图5所示的顶点数据来说，发送窗口的起始顶点数据下标为0，宽度为3，则顶点发送模块8603第一次向图元装配模块8604发送的顶点数据的下标为0、1、2。发送完毕之后发送窗口的起始顶点移动至下标3，相类似的，则顶点发送模块8603第二次向图元装配模块8604发送的顶点数据下标为3、4、5。若顶点数据缓存8602中的顶点数据不再补充，则顶点发送模块8603最后一次向图元装配模块8604发送的顶点数据下标为6、7并且在第三个位置上标记非顶点数据，完成发送。若顶点数据缓存8602中补充新的顶点数据，则顶点数据下标变更为6、7、0、1、2、3；则发送窗口从而能够在顶点数据缓存8602的数据中循环遍历，可以理解地，上述循环遍历状态同样可以应用于以下所有情况，后续不再赘述。

[0054] 若图元类型为GL_LINES，发送窗口的起始位置仍旧为顶点数据缓存8602中的前三个顶点数据；移动步长设置为2，即每次向后移动两个顶点。当设置完成之后，顶点发送模块8603根据顶点图元信息判断前两个顶点是否都为需要装配的顶点：若发送窗口中的其中一个顶点为图元重启顶点，则将窗口起始位置移动到图元重启顶点之后第一个顶点位置，并做相同的判断；若发送窗口移动到顶点数据缓存8602的最后并且有效顶点只有一个，则删除发送窗口中顶点，发送完毕。以图5所示的顶点数据来说，发送窗口的起始顶点数据下标为0，宽度为3，顶点发送模块8603第一次向图元装配模块8604发送的顶点数据的下标为0、1、2，其中2顶点图元装配模块8604不处理。依次类推，窗口移动2个顶点，顶点发送模块8603第二次向图元装配模块8604发送的顶点数据的下标为2，3、4；若2为重启图元的顶点，则发送窗口起始位置改为3，向图元装配模块8604发送下标为3、4，5的顶点数据，此时，顶点数据缓存8602中还剩余5、6、7。顶点发送模块8603再向图元装配模块8604发送一次顶点数据时，顶点数据缓存8602中仅剩余下标为7的一个顶点数据，则顶点发送模块8603删除该下标为7的顶点数据不再发送，发送完毕。

[0055] 若图元类型为GL_LINE_STRIP，发送窗口的起始位置为顶点数据缓存8602中的前三个顶点数据；移动步长设置为1，即每次向后移动一个顶点。设置完成之后，顶点发送模块8603可以根据顶点图元信息判断发送窗口内的前两个顶点是否都为需要装配的顶点：若其中一个顶点为重启图元的顶点，则将发送窗口的起始位置移动到该重启图元的顶点之后第一个顶点位置，并做相同的判断；若发送窗口移动到顶点数据缓存8602的最后并且有效顶点只有一个，则删除发送窗口中顶点数据，发送完毕。以图5所示的顶点数据来说，发送窗口的起始顶点数据下标为0，宽度为3，顶点发送模块8603第一次向图元装配模块8604发送的顶点数据的下标为0、1、2，顶点2不做处理。发送窗口移动一个顶点，顶点发送模块8603第二次向图元装配模块8604发送的顶点数据的下标为1、2、3，以此类推。针对重启顶点和删除顶点的操作与前述GL_LINES类型类似，在此不再赘述。

[0056] 若图元类型为GL_LINE_LOOP，发送窗口的起始位置仍旧为顶点数据缓存8602中的前三个顶点数据，并在发送窗口的第三个位置保存顶点数据缓存8602中的第一个顶点；移动步长设置为1，即每次向后移动一个顶点。设置完成之后，顶点发送模块8603根据顶点图元信息判断前两个顶点是否都为需要装配的顶点：若其中一个顶点为重启顶点，则将发送窗口的起始位置移动到图元重启顶点之后第一个顶点位置，并更新发送窗口的第三个位置

的顶点为此时顶点数据缓存8602中的第一个顶点，并做相同的判断；若发送窗口移动到顶点数据缓存8602最后并且有效顶点只有一个，则将发送窗口的第三个顶点位置的顶点数据赋值给发送窗口的第二个顶点位置的顶点数据值。以图5所示的顶点数据来说，发送窗口的起始顶点下标及宽度如前述各类型的描述所示，顶点发送模块8603第一次向图元装配模块8604发送顶点数据的下标为0、1、0，发送窗口中第三个顶点下标保存为起始顶点的下标。依次类似，顶点发送模块8603第二次向图元装配模块8604发送顶点数据的下标为1、2、0。若2顶点为重启顶点，则发送窗口的起始位置变为3，发送窗口第三个位置保存3顶点的下标，此时发送窗口状态为3、4、3。即发送下标为3、4、3的顶点数据。

[0057] 若图元类型为GL_TRIANGLE，发送窗口的起始位置为顶点数据缓存8602中的前三个顶点；移动步长设置为3，即每次向后移动三个顶点。设置完成之后，顶点发送模块8603根据顶点图元的信息判断三个顶点是否都为需要装配的顶点：若发送窗口中的其中一个顶点为图元重启顶点，则将发送窗口的起始位置移动到该图元重启顶点之后的第一个顶点位置，并做相同的判断；若发送窗口移动到顶点数据缓存8602最后并且包含无效顶点，则删除发送窗口中所有顶点，发送完毕。以图5所示的顶点数据来说，结合上述描述，GL_TRIANGLE类型与GL_POINTS类型的的顶点数据发送情况类似，顶点发送模块8603第一次向图元装配模块8604发送的顶点数据下标为0、1、2。顶点发送模块8603第二次向图元装配模块8604发送的顶点数据的下标为3、4、5，以此类推。针对重启顶点和删除顶点的操作与前述GL_LINES类型类似，在此不再赘述。

[0058] 若图元类型为GL_TRIANGLE_STRIP，发送窗口的起始位置为顶点数据缓存8602中的前三个顶点；移动步长设置为1，即每次向后移动一个顶点。设置完成之后，顶点发送模块8603根据顶点图元的信息判断三个顶点是否都为需要装配的顶点：若发送窗口中的其中一个顶点为图元重启顶点，则将发送窗口起始位置移动到该图元重启顶点之后的第一个顶点位置，并做相同的判断；若发送窗口移动到顶点数据缓存8602的最后并且包含无效顶点，则删除发送窗口中所有顶点，发送完毕。以图5所示的顶点数据来说，结合上述描述，GL_TRIANGLE_STRIP类型与GL_LINE_STRIP类型的顶点数据发送情况类似，在此不再赘述。

[0059] 若图元类型为GL_TRIANGLE_FAN或GL_POLYGON，发送窗口的起始位置为顶点数据缓存8602中的前三个顶点；移动步长设置为1，即每次向后移动一个顶点，且在移动过程中，保持发送窗口的第一个顶点值不变。设置完成之后，顶点发送模块8603根据顶点图元的信息判断三个顶点是否都为需要装配的顶点：若发送窗口中的其中一个顶点为图元重启顶点，则将发送窗口的起始位置移动到该图元重启顶点之后的第一个顶点位置，并更新发送窗口中第一个位置的顶点值为此时顶点数据缓存8602的第一个顶点，之后做相同的判断；若发送窗口移动到顶点数据缓存8602最后并且包含无效顶点，则删除发送窗口中所有顶点，发送完毕。以图5所示的顶点数据来说，发送窗口的起始顶点下标及宽度如前述各类型的描述所示，顶点发送模块8603第一次向图元装配模块8604发送顶点数据的下标为0、1、2。发送窗口移动1个顶点并且保持第一个顶点不变，此时发送窗口中的顶点数据的下标为0、2、3。即顶点发送模块8603发向图元装配模块8604发送顶点数据的下标为0、2、3；后续以此类推。若顶点2为重启顶点，则移动发送窗口位置，此时发送窗口中的顶点数据下标为3、4、5，即顶点发送模块8603向图元装配模块8604发送顶点数据的下标为3、4、5并且保持3顶点不变。那么在下次顶点发送模块8603向图元装配模块8604发送时，发送窗口中的顶点数据

下标为3、5、6。针对删除顶点的操作与前述类似，在此不再赘述。。

[0060] 此外，在一些示例中，顶点发送模块8603还可经配置以根据顶点数据对应的顶点信息和/或图元配置状态信息对顶点数据缓存8602中的顶点数据进行筛选，以舍弃不参与图元装配的顶点数据。举例来说，不需要装配的顶点包括无效顶点，顶点发送模块8603会将该顶点从顶点数据缓存8602中丢弃，重新读取接下来的顶点数据并做相同的判断。

[0061] 在另一些示例中，相应于每次顶点发送模块8603均一次性地向图元装配模块8604发送三个顶点数据，那么图元装配模块8604也就能够一次性地接收3个顶点数据作为输入，针对图元类型进行快速装配，此时图元装配模块8604可以被看成为一个多路选择器，将接收到的3个顶点数据根据图元类型传递至相应的输出端口；具体来说，图元装配模块8604可以根据图元配置信息例如图元颜色、大小、图元顶点顺序以及图元样式等对顶点进行调整等操作，以设置图元的颜色、大小等。通过该示例描述，图元装配模块8604的顶点输入数据量和图元输出数据量是匹配的，也就是说输入图元装配模块8604的数据量与图元装配模块8604输出的数据量之间相适应，或者在同一时钟内两者相等。

[0062] 需要说明的是，传统的图元装配过程，是以类似状态机的形式进行的，即一次仅接收一个顶点，输出为三个顶点。结合图5所示的FIFO结构以及GL_TRIANGLE类型为例，参见图6，设定现有0、1、2、3、4、5、6七个顶点依次输入。起始状态为state1，首先接收0顶点并判断是否为图元重启顶点，不为重启顶点时保存顶点，将图元装配的状态置为state2，并等待接收第二个顶点。当1顶点输入时，同样判断其是否为图元重启顶点，若不为重启顶点时保存顶点，并将图元状态置为state3，并等待接收第三个顶点。当2顶点输入时，且其不为重启顶点时，将顶点根据图元类型进行排序、设置图元的属性(包括图元大小、图元颜色等)后，输出图元，并重新设置状态为state1且开始新的三角形装配。若在上面中间过程遇到图元重启顶点则删除该保存的顶点，并更新图元装配的状态为state1。例如当1顶点为重启图元，删除0顶点，将状态置为state1，等待下一个顶点输入。若最后顶点不够三个时，删除保存的顶点，将状态置为state1。由上述可知，当以类似状态机的形式进行图元装配时，完成一个图元装配至少需要3个时钟周期。

[0063] 而在本示例中，图元装配模块8604能够一次性地接收3个顶点数据作为输入，进行图元装配，并根据图元类型将图元传递至相应的输出端口；在具体实施过程中，可以优选通过多路选择器的形式进行实现。同样以GL_TRIANGLE类型为例：仍然设定现有0、1、2、3、4、5、6七个顶点依次输入。由于顶点发送模块8603发送顶点的过程中已经对顶点数据进行了筛选、舍弃或删除，所以输入到图元装配模块8604的顶点数据均为有效顶点，只需按照图元类型对图元设置属性即可。比如，当0、1、2三个顶点输入时，根据图元的设置将三个顶点进行排序、设置图元属性，就可以直接输出图元到流水线的后续模块，并接收新的三个顶点。由于存在七种不同的图元类型，不同的图元类型设置的属性和排序不一致，因此需要最多七路不同的处理方式即可。从此可以看出本发明实施例所采用的技术方案完成一个图元装配仅需要一个时钟周期，相比于传统的图元装配，极大地提高了图元装配的效率。

[0064] 参见图4，位于图元装配模块8604之后，图元装配级86还可以包括图元数据缓存8605以及包围盒生成模块8606。图元装配模块8604将输出的装配完毕后的图元缓存至图元数据缓存8605，包围盒生成模块8606经配置以根据图元类型，从图元数据缓存8605中读取图元数据并对装配完毕的图元数据生成包围盒。

[0065] 在一些示例中，由于图元装配模块8604的图元装配效率的提升，为了平衡图元装配模块8604与包围盒生成模块8606的处理速率，防止在流水线中避免由于速率不匹配造成流水线停滞，本发明实施例还可以在图元装配模块8605与包围盒生成模块8606之间设置一图元数据缓存8605，图元数据缓存8605同样可以优选为符合先入先出(FIFO)结构，并且在实施过程中，其可以是高速缓冲存储器以作为图元装配级86的一部分，也可以为了节省片上面积和资源从图形存储器40中开辟一存储空间。由于顶点数据缓存8602存放的是所有目前需要处理的顶点数据，图元数据缓存8605实际上存放的是排序组合后的顶点数据。所以图元数据缓存8605的容量大小可以与顶点数据缓存8602设置一致；而图元属性信息则可以通过另外的存储空间进行保存，这里暂不做赘述。以FIFO结构为例，图元数据缓存8605的读取规则可以是只要图元数据缓存8605不为空，包围盒生成模块8606就根据图元类型从中顺序读取一个图元，直至图元数据缓存8605为空，此时说明所有数据处理完毕。。

[0066] 在一些示例中，包围盒生成模块8606不再直接从图元装配模块8604接收图元数据来产生包围盒，而是通过图元数据缓存8605来获取图元数据，在获取完毕后，仍然可以示例性地将读取的图元数据经历视见体剔除、背面剔除、小三角形的包围盒处理以及裁剪处理，生成包围盒。

[0067] 图7为本发明实施例提供的一种提高图元装配效率的方法，该方法可以应用于图3或图4所示的设置于GPU 12内图形处理管线80中的图元装配级86部分，如图7所示该方法可以包括：

S701：抓取待装配的顶点数据并进行缓存；

S702：从缓存的顶点数据中一次性读取多个有效的顶点数据进行图元装配，输出装配完毕后的图元。

[0068] 对于图7来说，在一些示例中，抓取待装配的顶点数据并进行缓存，包括：

通过顶点抓取模块8601根据命令处理器30所下发的调度指令从显存中读取待装配的顶点数据，并将待装配的顶点数据存放于顶点数据缓存8602。

[0069] 对于图7来说，在一些示例中，从缓存的顶点数据中一次性读取多个有效的顶点数据进行图元装配，输出装配完毕后的图元，包括：通过顶点发送模块8603对顶点数据缓存8602中的顶点数据进行读取，向图元装配模块8604一次性发送多个有效的顶点数据；图元装配模块8604一次性接收顶点发送模块8603所发送的多个有效的顶点数据，并根据图元类型对接收到的多个有效的顶点数据进行图元装配，输出装配完毕后的图元。

[0070] 对于图7来说，在一些示例中，通过顶点发送模块8603对顶点数据缓存8602中的顶点数据进行读取，向图元装配模块8604一次性发送多个有效的顶点数据，包括：

顶点发送模块8603根据图元类型设置向图元装配模块8604发送顶点的发送窗口位置和移动步长；

顶点发送模块8603根据发送窗口位置和移动步长从顶点数据缓存8602中读取多个有效的顶点数据，并向图元装配模块8604发送所述多个读取的有效的顶点数据。

[0071] 对于图7来说，在一些示例中，所述方法还包括：通过顶点发送模块8603根据顶点数据对应的顶点信息和/或图元配置状态信息对顶点数据缓存8602中的顶点数据进行筛选，以舍弃不参与图元装配的顶点数据。

[0072] 对于图7来说，在一些示例中，所述图元装配模块8604为一多路选择器，相应地，图

元装配模块8604一次性接收顶点发送模块8603所发送的多个有效的顶点数据，并根据图元类型对接收到的多个有效的顶点数据进行图元装配，输出装配完毕后的图元，包括：图元装配模块8604能够一次性地接收3个顶点数据作为输入进行图元装配，并将接收到的3个顶点数据根据图元类型并传递至相应的输出端口。

[0073] 对于图7来说，所述方法还包括：通过图元装配模块8604将输出的装配完毕后的图元缓存至图元数据缓存8605；通过包围盒生成模块8606根据图元类型，从图元数据缓存8605中读取图元数据并对装配完毕的图元数据生成包围盒。

[0074] 在上述一或多个实例或示例中，所描述的功能可实施于，所描述功能可实施于硬件、软件、固件或其任何组合中。如果实施于软件中，那么可将功能作为一或多个指令或代码存储在计算机可读媒体上或经由计算机可读媒体传输。计算机可读媒体可包含计算机数据存储媒体或通信媒体，通信媒体包含促进将计算机程序从一处传递到另一处的任何媒体。数据存储媒体可为可由一或多个计算机或一或多个处理器存取以检索用于实施本发明中描述的技术的指令、代码和/或数据结构的任何可用媒体。举例来说且非限制，此类计算机可读媒体可包括U盘、移动硬盘、RAM、ROM、EEPROM、CD-ROM或其它光盘存储装置、磁盘存储装置或其它磁性存储装置，或可用于运载或存储呈指令或数据结构的形式的所要程序代码且可由计算机存取的任何其它媒体。并且，任何连接被恰当地称作计算机可读媒体。举例来说，如果使用同轴电缆、光纤电缆、双绞线、数字订户线(DSL)或例如红外线、无线电和微波等无线技术从网站、服务器或其它远程源传输软件，那么同轴电缆、光纤电缆、双绞线、DSL或例如红外线、无线电和微波等无线技术包含于媒体的定义中。如本文中所使用，磁盘和光盘包含压缩光盘(CD)、激光光盘、光学光盘、数字多功能光盘(DVD)、软性磁盘和蓝光光盘，其中磁盘通常以磁性方式再现数据，而光盘利用激光以光学方式再现数据。以上各项的组合也应包含在计算机可读媒体的范围内。

[0075] 代码可由一或多个处理器执行，所述一或多个处理器例如是一或多个数字信号处理器(DSP)、通用微处理器、专用集成电路(ASIC)、现场可编程逻辑阵列(FPGA)或其它等效的可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件。。因此，如本文中所使用的术语“处理器”和“处理单元”可指前述结构或适于实施本文中所描述的技术的任何其它结构中的任一者。另外，在一些方面中，本文中所描述的功能性可在经配置用于编码和解码的专用硬件和/或软件模块内提供，或者并入在组合式编解码器中。而且，所述技术可完全实施于一或多个电路或逻辑元件中。

[0076] 本发明实施例的技术可实施于各种各样的装置或设备中，所述装置或设备包含无线手持机、集成电路(IC)或一组IC(即，芯片组)。本发明中描述各种组件、模块或单元是为了强调经配置以执行所公开的技术的装置的功能方面，但未必需要由不同硬件单元实现。实际上，如上文所描述，各种单元可结合合适的软件和/或固件组合在编码解码器硬件单元中，或者通过互操作硬件单元的集合来提供，所述硬件单元包含如上文所描述的一或多个处理器。

[0077] 已描述了本发明的各种方面。这些和其它实施例在所附权利要求书的范围内。需要说明的是：本发明实施例所记载的技术方案之间，在不冲突的情况下，可以任意组合。

[0078] 以上所述，仅为本发明的具体实施方式，但本发明的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本发明揭露的技术范围内，可轻易想到变化或替换，都应涵

盖在本发明的保护范围之内。因此，本发明的保护范围应以所述权利要求的保护范围为准。

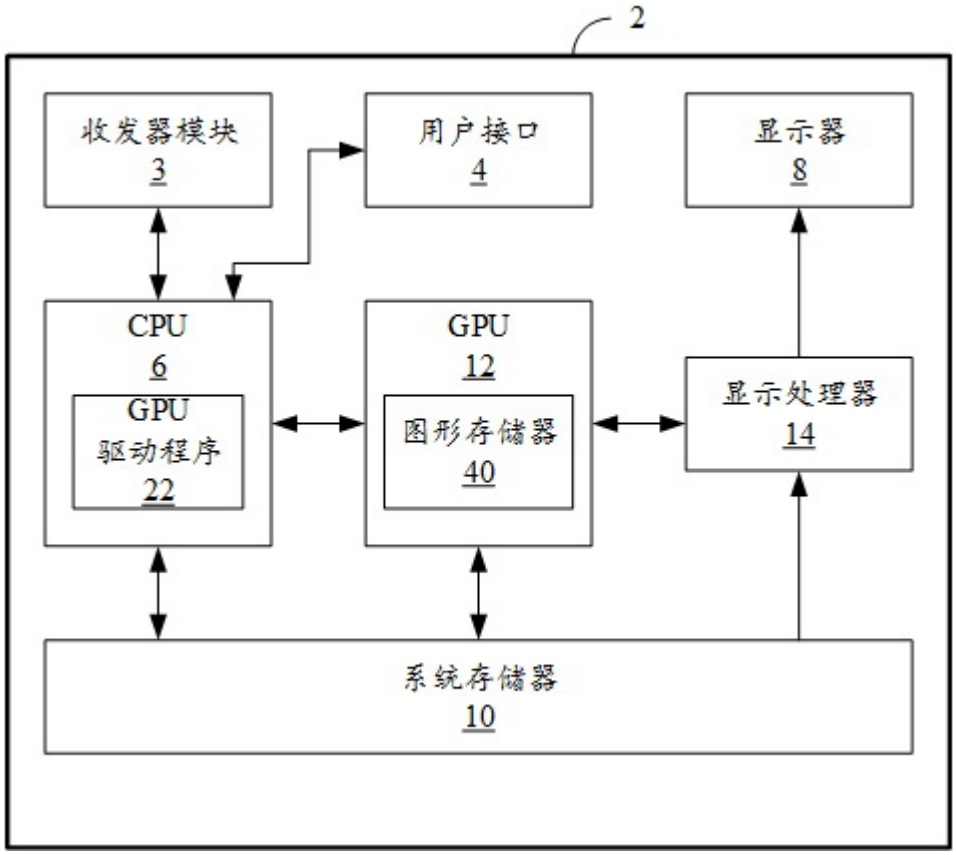

2
收发器模块
3
用户接口
4
显示器
8
CPU
6
GPU
驱动程序
22
GPU
12
图形存储器
40
显示处理器
14
系统存储器
10

图 1

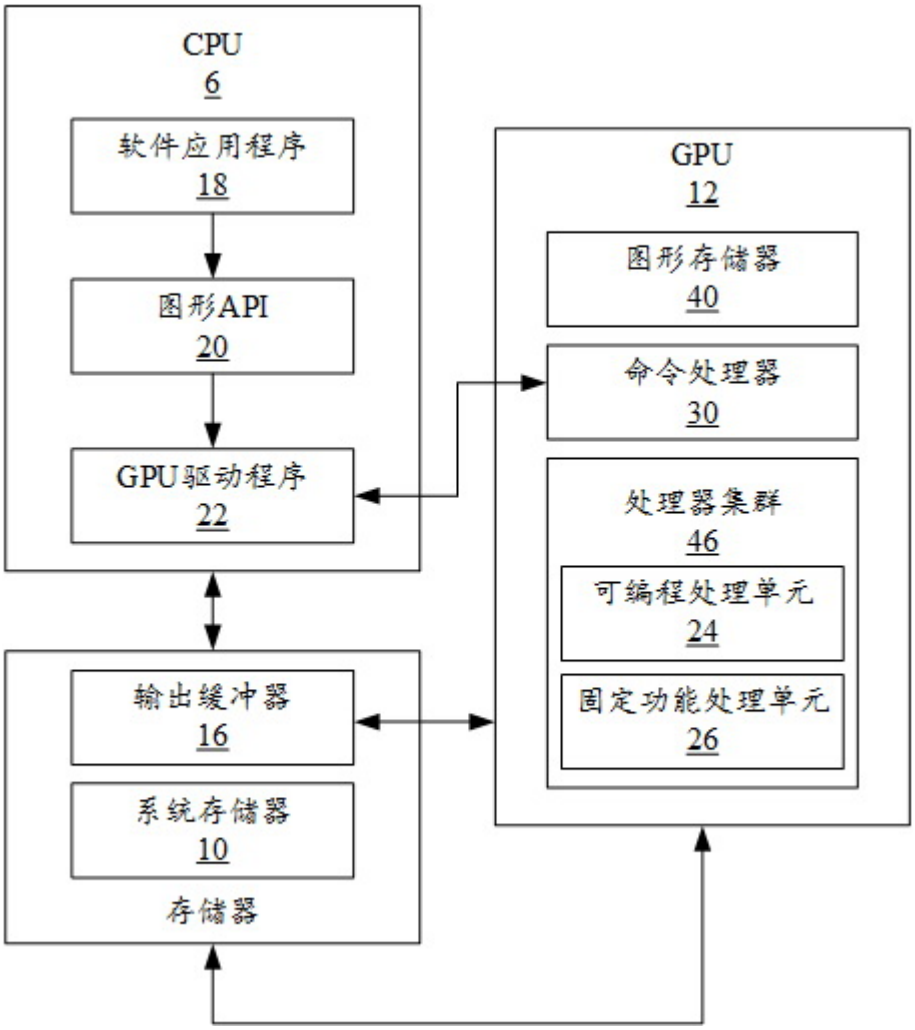

CPU
6
软件应用程序
18
图形API
20
GPU驱动程序
22
输出缓冲器
16
系统存储器
10
存储器
GPU
12
图形存储器
40
命令处理器
30
处理器集群
46
可编程处理单元
24
固定功能处理单元
26

图 2

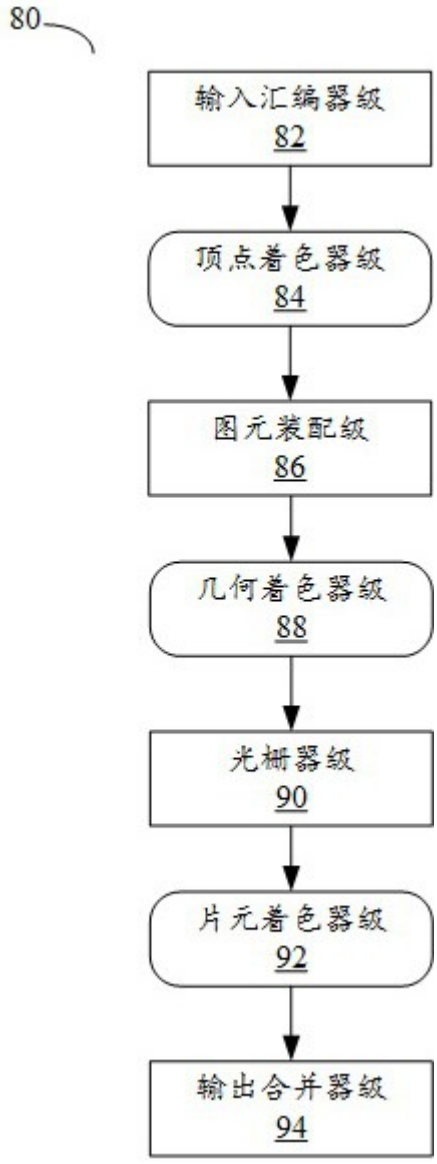

80
输入汇编器级
82
顶点着色器级
84
图元装配级
86
几何着色器级
88
光栅器级
90
片元着色器级
92
输出合并器级
94

图 3


86
顶点抓取模块
8601
顶点数据缓存
8602
顶点发送模块
8603
图元装配模块
8604
图元数据缓存
8605
包围盒生成模块
8606

图 4

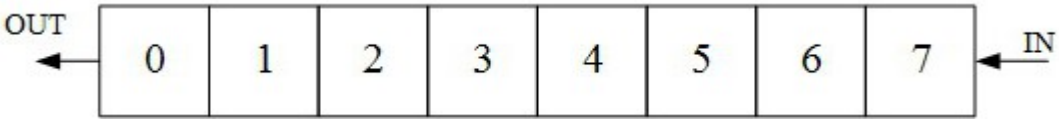

OUT
0
1
2
3
4
5
6
7
IN

图 5

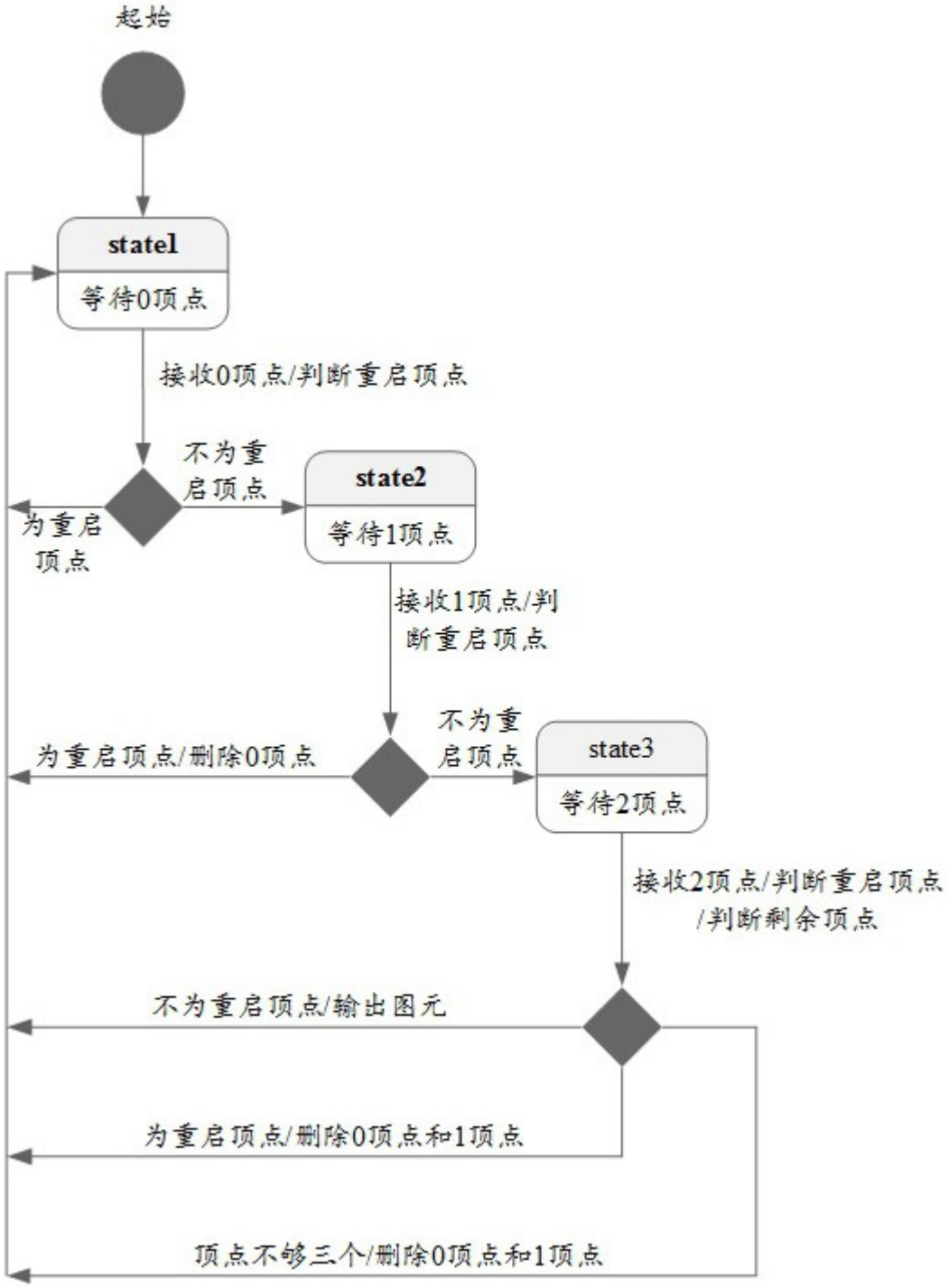

起始
state1
等待0顶点
接收0顶点/判断重启顶点
不为重启顶点
为重启顶点
state2
等待1顶点
接收1顶点/判断重启顶点
为重启顶点/删除0顶点
不为重启顶点
state3
等待2顶点
接收2顶点/判断重启顶点/判断剩余顶点
不为重启顶点/输出图元
为重启顶点/删除0顶点和1顶点
顶点不够三个/删除0顶点和1顶点

图 6

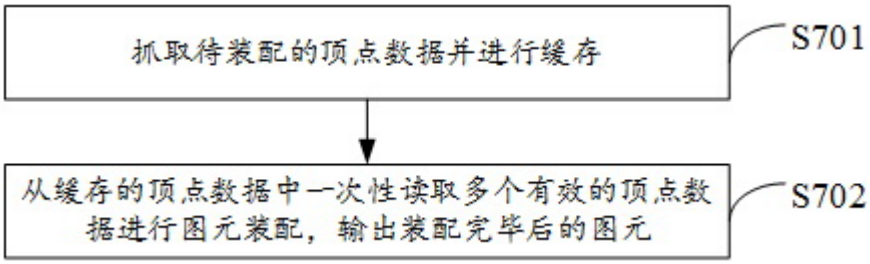

抓取待装配的顶点数据并进行缓存
S701
从缓存的顶点数据中一次性读取多个有效的顶点数据进行图元装配，输出装配完毕后的图元
S702

图 7