(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 101978352 A
(43)申请公布日 2011.02.16

(21)申请号 200880126821.2

(22)申请日 2008.12.15

(30)优先权数据
61/013,527 2007.12.13 US

(85)PCT申请进入国家阶段日
2010.08.13

(86)PCT申请的申请数据
PCT/US2008/086835 2008.12.15

(87)PCT申请的公布数据
WO2009/076671 EN 2009.06.18

(71)申请人 先进微装置公司
地址 美国加利福尼亚州

(72)发明人 P·布林则

(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 龚颐雯

(51)Int.Cl.
G06F 9/445(2006.01)
G06F 9/48(2006.01)

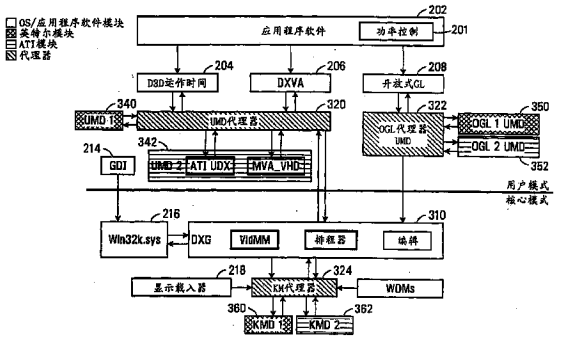
权利要求书 2 页 说明书 17 页 附图 14 页

(54)发明名称

用于具有多重图形子系统、减少的功率消耗模式的计算装置的驱动程序架构、软件和方法

(57)摘要

现今许多的计算装置也许包含二个或更多个图形子系统。多重图形子系统可以具有不同的能力，并且可以例如消耗不同数量的电力，因为一个子系统较其它的子系统消耗更多的平均功率。较高的功率消耗图形子系统可以耦接到装置，并且此外可以用来取代较低的功率消耗图形子系统，导致较高的性能或额外的能力，但是增加总功率消耗。藉由从使用之较高的功率消耗图形子系统转变至较低的功率消耗图形子系统，同时设置该较高的功率消耗图形子系统于较低的功率消耗模式，则减少总功率消耗。处理器执行应用程序软件和驱动程序软件。驱动程序软件包含第一和第二驱动程序组件用来分别控制该第一和第二图形子系统之操作。其它代理器驱动程序组件根据该第一和第二图形子系统之哪一个是在使用中而路由呼叫（例如，API/DDI 呼叫）至该第一和第二驱动程序组件其中之一。



CN 101978352 A

1. 一种电子装置，包括：

第一图形子系统，用于绘成图形；

第二图形子系统，用于绘成图形；

至少一个显示器，与该第一图形子系统和该第二图形子系统的至少其中一个通信；

处理器，执行应用程序软件和驱动程序软件，该驱动程序软件包括第一和第二驱动程序组件和代理器驱动程序组件，该第一和第二驱动程序组件用来分别控制该第一和该第二图形子系统的操作，而该代理器驱动程序组件用来依据该第一和第二图形子系统的哪一个是在使用中而路由呼叫从该应用程序软件至该第一和第二驱动程序组件的其中一个。

2. 如权利要求 1 所述的电子装置，其中，该处理器执行指令，使得该处理器将该电子装置从其中该第二图形子系统绘成图形于该显示器上的第一模式转变至其中该第一图形子系统绘成图形于该显示器上的第二模式，以及该第二图形子系统是处在较低功率消耗模式。

3. 如权利要求 1 所述的电子装置，其中，该第一和第二驱动程序组件包括执行于该处理器的用户模式的驱动程序组件。

4. 如权利要求 1 所述的电子装置，其中，该第一和第二驱动程序组件包括执行于该处理器的核心模式的驱动程序组件。

5. 如权利要求 1 所述的电子装置，其中，该第一和第二图形子系统的哪一个是在使用中是根据该电子装置的功率状态而定。

6. 如权利要求 1 所述的电子装置，其中，该代理器驱动程序组件路由呼叫从在该装置的操作系统至该第一和第二驱动程序组件的其中一个，是根据该第一和第二图形子系统的哪一个是在使用中而定。

7. 如权利要求 1 所述的电子装置，其中，该代理器驱动程序组件建立一致结构确认于该第一和第二驱动程序组件中相等的驱动程序功能用来实施该路由。

8. 一种电子装置，包括：

第一图形子系统，用于绘成图形；

第二图形子系统，用于绘成图形；

显示器，与该第一图形子系统和该第二图形子系统二者通信；

处理器，执行应用程序软件和驱动程序软件，该驱动程序软件包括：

第一和第二用户模式驱动程序组件，用来分别控制该第一和第二图形子系统的操作，和用户模式代理器驱动程序组件，用来根据该第一和第二图形子系统的哪一个是在使用中而路由呼叫从该应用程序软件至该第一和第二用户模式驱动程序组件的其中一个；

第一和第二核心模式驱动程序组件，用来分别控制该第一和第二图形子系统的操作，以及核心模式代理器驱动程序组件，用来根据该第一和第二图形子系统的哪一个是在使用中而路由呼叫从该用户模式驱动程序组件的其中一个至该第一和第二核心模式驱动程序组件的其中一个。

9. 一种用来执行于计算机装置上的计算机可读取媒体存储驱动程序软件包括，该计算装置包括：

第一图形子系统，用于绘成图形；

第二图形子系统，用于绘成图形；

显示器，与该第一图形子系统和该第二图形子系统二者通信；以及

处理器；

该驱动程序软件包括代理器驱动程序组件，其用来根据该第一和第二图形子系统的哪一个是在使用中而路由呼叫从该应用程序软件至第一和第二驱动程序组件的其中一个，该第一和第二驱动程序组件用来分别控制该第一和第二图形子系统的操作。

10. 如权利要求 9 所述的计算机可读取媒体，其中，该代理器驱动程序组件建立一致结构确认于该第一和第二驱动程序组件中相等的驱动程序功能用来实施该路由。

11. 如权利要求 10 所述的计算机可读取媒体，其中，该代理器驱动程序组件根据该第一和第二图形子系统的哪一个是在使用中而路由呼叫从在该装置的操作系统至该第一和第二驱动程序组件的其中一个。

12. 如权利要求 9 所述的计算机可读取媒体，其中，该第一和第二驱动程序组件包括执行于该处理器的用户模式的驱动程序组件。

13. 如权利要求 9 所述的计算机可读取媒体，其中，该第一和第二驱动程序组件包括执行于该处理器的核心模式的驱动程序组件。

14. 一种操作电子装置的方法，该电子装置包括用于绘成图形的第一和第二图形子系统，该方法包括下列步骤：

接收来自软件应用程序的驱动程序呼叫或执行于该电子装置的操作系统；

根据该第一和第二图形子系统的哪一个是在使用中而路由来自软件应用程序的驱动程序呼叫至用来分别控制该第一和第二图形子系统的操作的第一和第二软件驱动程序组件的其中一个。

15. 如权利要求 14 所述的方法，进一步包括：

建立一致结构确认于该第一和第二驱动程序组件中相等的驱动程序功能用来实施该路由。

用于具有多重图形子系统、减少的功率消耗模式的计算装置的驱动程序架构、软件和方法

[0001] 本申请案主张于2007年12月13日提出申请之美国临时专利申请案序号61/013,527,案名“用于具有多重图形子系统、减少的功率消耗模式之计算装置之驱动程序架构,软件和方法(DRIVER ARCHITECTURE FOR COMPUTING DEVICE HAVING MULTIPLE GRAPHICS SUBSYSTEMS,REDUCED POWER CONSUMPTION MODES,SOFTWARE AND METHODS)”之优先权,该案发明人为Phil Mummah和Paul Blinzer,而所有权人为本案受让人,该案之全部内容由此结合本案作为参考。本申请案亦相关于2006年5月30日提出申请之美国专利申请案序号11/421,005,和2007年5月30日提出申请之美国专利申请案序号11/755,625,该二个申请案之内容结合本案作为参考。

技术领域

[0002] 本发明大体上系关于计算装置,尤系关于具有多重图形子系统、和关联之软件驱动程序之装置。于一个态样中,本发明亦系关于用来降低此等装置中功率消耗的方法。

背景技术

[0003] 许多电子装置,譬如习知的计算装置现在包含图形子系统能够描绘二或三维图形;译码和编码动式视讯:等等。欲提供这些特征和所希望之处理速度,现代的图形子系统包含持续增加数目之晶体管。毫不奇怪的,增加晶体管数量已导致由图形子系统所造成之对应较高的电力消耗。

[0004] 结果,最快的和最富特征的图形子系统已大部分预留给能够符合增加功率要求之装置。譬如膝上型、个人数字助理、视频和音频播放器、手机、等等之可携式计算装置通常已装备成受到功能限制,但是具有电性效率(亦即,低功率)之组件。

[0005] 时常这些图形子系统整合于其它的计算组件中,譬如处理器互连接电路(时常称之为“芯片组”)。

[0006] 最近,有提供用于可携式装置之图形特征和性能以对抗静止式计算机之那些功能之倾向。往往,此藉由允许附加视情况选用之外部的高功率图形子系统于可携式装置而达成。例如,PCI快捷(PCIe)标准考虑PCI快捷之互连接兼容图形卡,包含图形子系统,作为至膝上型计算机装置之外部组件。

[0007] 于存在之多重图形子系统中,时常希望切换计算装置之操作状态使用一个或另一个图形子系统而不须重新起动(例如,重新开机(rebooting))计算装置。

[0008] 不幸的是,一些操作系统之软件架构仅仅考虑使用单一的图形驱动程序。于是,于存在之多重图形子系统中,此单一驱动程序需要控制所有的多重子系统之操作。此也许是不切实际的,尤其是如果子系统由不同的制造者所提供时。

[0009] 因此,需要可以使用多重图形驱动程序之软件和装置。

发明内容

[0010] 现今许多计算装置可以包含二个或更多个图形子系统。该多重图形子系统可以具有不同的能力，以及可以例如消耗不同数量之电力，因为一个子系统较其它的子系统消耗更多的平均功率。较高的功率消耗图形子系统可以耦接到装置，并且此外可以用来取代，或附加至较低的功率消耗图形子系统，导致较高的性能或额外的能力，但是增加总功率消耗。藉由从使用之较高的功率消耗图形子系统转变至较低的功率消耗图形子系统，同时设置该较高的功率消耗图形子系统于较低的功率消耗模式，则减少总功率消耗。

[0011] 处理器执行应用程序软件和驱动程序软件。驱动程序软件包含第一和第二驱动程序组件用来分别控制该第一和第二图形子系统之操作。其它代理器驱动程序组件依于该第一和第二图形子系统之哪一个是在使用中而路由呼叫（例如，API/DDI 呼叫）从该应用程序软件至该第一和第二驱动程序组件之其中之一。

[0012] 习知使用上，此代理器驱动程序组件表现单一接口至操作系统和应用程序软件，同时允许使用二个分离的驱动程序组件。此代理器驱动程序组件可以是窗口维斯塔 (Windows Vista) 用户模式驱动程序 (user mode driver，UMD) 组件和 / 或核心模式驱动程序 (kernel mode driver，KMD) 组件。

[0013] 依照本发明之态样，提供一种电子装置。该电子装置包括：第一图形子系统，用于绘成图形；第二图形子系统，用于绘成图形；至少一个显示器，与该第一图形子系统和该第二图形子系统之至少其中之一通信通信；处理器，执行应用程序软件和驱动程序软件，该驱动程序软件包括第一和第二驱动程序组件用来分别控制该第一和第二图形子系统之操作，和代理器驱动程序组件用来依于该第一和第二图形子系统之哪一个是在使用中而路由呼叫从该应用程序软件至该第一和第二驱动程序组件之其中之一。

[0014] 依照本发明之另一个态样，提供一种电子装置，包括：第一图形子系统，用于绘成图形；第二图形子系统，用于绘成图形；显示器，与该第一图形子系统和该第二图形子系统二者通信；处理器，执行应用程序软件和驱动程序软件，该驱动程序软件包括第一和第二用户模式驱动程序组件用于分别控制该第一和第二图形子系统之操作，和用户模式代理器驱动程序组件用来依于该第一和第二图形子系统之哪一个是在使用中而路由呼叫从该应用程序软件至该第一和第二用户模式驱动程序组件之其中之一；第一和第二核心模式驱动程序组件，用来分别控制该第一和第二图形子系统之操作，以及核心模式代理器驱动程序组件，用来依于该第一和第二图形子系统之哪一个是在使用中而路由呼叫从该用户模式驱动程序组件其中之一至该第一和第二核心驱动程序组件之其中之一。

[0015] 依照本发明之又另一个态样，提供一种操作电子装置之方法，该电子装置包括用于绘成图形的第一和第二图形子系统。该方法包括下列步骤：接收来自软件应用程序之驱动程序呼叫或执行于该电子装置之操作系统；依于该第一和第二图形子系统之哪一个是在使用中而路由来自软件应用程序之驱动程序呼叫至用来分别控制该第一和第二图形子系统之操作之第一和第二软件驱动程序组件之其中之一。

[0016] 于查阅本发明之特定实施例结合所附图式之下列说明，本发明之其它态样和特征对于熟悉此项技术而言将变得很清楚。

附图说明

[0017] 于仅以举例方式例示之图形中，本发明之实施例：

[0018] 图 1 为本发明之范例实施例，计算装置之简化示意方块图；

[0019] 图 2 为本发明之范例实施例，计算装置之简化示意方块图；

[0020] 图 3 为习知软件之简化功能方块图；

[0021] 图 4 为于图 2 之计算装置范例软件之简化功能方块图，包含本发明之范例实施例之驱动程序软件；

[0022] 图 5 为本发明之范例实施例，于图 2 之装置由软件所实施之详细步骤之流程图；

[0023] 图 6 示意地显示于图 4 之软件中 API/DDI 呼叫。

[0024] 图 7 为本发明之范例实施例，于图 2 之装置由软件所实施之详细步骤之流程图；

[0025] 图 8 为图 2 之计算装置之进一步简化示意方块图；

[0026] 图 9 为本发明之范例实施例，于图 2 之装置由软件所实施之详细步骤之流程图；

[0027] 图 10 为本发明之另一范例实施例，计算装置之部分之进一步部分简化示意方块图；

[0028] 图 11 为本发明之范例实施例，于图 10 之装置由软件所实施之详细步骤之流程图；

[0029] 图 12A 和 12B 为例示图 10 之装置之操作之简化方块图；

[0030] 图 13 为本发明之另一范例实施例，计算装置之部分之进一步部分简化示意方块图。

具体实施方式

[0031] 图 1 为电子装置 10 之简化、高阶层方块图，包含二个图形子系统 30 和 40、和显示器 26。如将变得很清楚，各图形子系统 30 和 40 包含特殊化之电子电路能够绘制计算机图形，于一个或多个之 2D 图、3D 图、译码动式视讯、等等之形式。

[0032] 一个图形子系统 40 可能消耗较其它的图形子系统 30 为高的平均功率。典型的情况是，图形子系统 40 消耗较高的平均功率，而具有较图形子系统 30 为大的图形绘制能力。图形子系统 40 例如也许能够较消耗较低之平均功率之图形子系统以较高之帧率 (frame rate) 绘制 2D 或 3D 图形。同样情况，图形子系统 30、40 不需具有相同的能力。图形子系统 40 较图形子系统 30 典型包含更多的功能区块。

[0033] 图形子系统 30 和 40 二者可以被实体或逻辑耦接至相同的显示器 26，于此显示器上显示绘制之图形。本发明之范例实施例，装置 10 可以从较高的功率消耗模式（于此模式至显示器 26 之图形由较高功率消耗之图形子系统 40 绘制成）切换至较低的功率消耗模式（于此模式至显示器 26 之图形由较低功率消耗之图形子系统 30 绘制成），并且图形子系统 40 被部分、完全或实质上禁能。

[0034] 习用上，可以用动态方式将高功率消耗模式转变至低功率消耗模式，而不需要装置 10 循环供电（亦即，电源切断和重新起动），并且可以于软件控制下由处理器 12 生效。如此情况，软件可以包含应用程序软件、韧体、装置驱动程序、BIOS、等等。

[0035] 本发明可以形成包含二个图形子系统之实际上任何电子装置之部分。就此种情况，装置 10 能够采取桌上型计算装置、可携式计算装置（包含膝上型计算机、PDA、移动式电话、视频和音频播放器、媒体中心、等等）之形式。

[0036] 于下文说明之范例实施例中，本发明之实施例揭示为形成移动式（膝上型）计算

装置之部分。

[0037] 详言之，图 2 为本发明之范例实施例，特定的移动式计算装置 10 之简化方块图。图 1 中显示描绘之装置 10 为根据习知的英特尔 x86 计算机架构之计算装置。然而，熟悉此项技术者将容易了解到本发明可以是具有其它架构，譬如，功率 PC(PowerPC) 架构、AMD x86、或其它已知的架构之计算装置之实施例。

[0038] 如所例示，范例装置 10 包含形成为中央处理单元 (CPU) 之处理器 12、主存储器 14、和全都透过整合之接口电路 16 和 18(亦称之为北桥 16 和南桥 18) 互连接之周边装置。所有这些装置可以形成在主机板上。

[0039] 接口电路 16 为高速接口，其藉由高速扩充总线 20 之方式而与 CPU 12、内存 14、和周边装置互相连接。接口电路 16 进一步互连接 CPU 12 至低速接口电路 18。一个或多个周边扩充凹槽 22 可以藉由高速扩充总线 20 之方式互连接至接口电路 16。范例高速扩充总线 20 为 PCI 快捷 (PCIe) 总线，其具有频宽每秒千兆位组范围，并且允许于此频宽资料转移读取和写入。

[0040] 接口电路 16 进一步包含第一图形子系统 30，具体实施为整合图形处理器 (integrated graphics processor，IGP)，适合用来产生视频讯号显示在显示器 26 上，该显示器 26 可以是于监视器、LCD 面板、电视、等等之形式。

[0041] 额外的第二图形子系统 40 形成装置 10 之部分。于此范例实施例中，图形子系统 40 被具体实施为形成在周边扩充卡 46 上之外部图形处理器。周边扩充卡 46 亦藉由于扩充总线 20 上之扩充凹槽 22 之方式连接至接口电路 16。如将变得很清楚，藉由设置第二图形子系统 40，装置 10 可以提供扩大的图形能力，此能力在其它情况不表现于装置 10 中。由第二图形子系统使用为帧缓冲器之图形内存 50 可以包含在周边扩充卡 46 上。同样情况，与图形子系统 40 通信之功率控制器 60 可以视需要选择形成于扩充卡 46 上，并且可以控制图形子系统 40 之操作。详言之，功率控制器 60 可以调节由图形子系统 40 之组件所使用之时脉，譬如内存和像素时脉；禁能（或者断接）图形子系统 40 之功能区块；降低施加到图形子系统 40 之部分之电压；或者不然设置子系统 40 于其中功率消耗以已知方式减少之一个或多个模式中。

[0042] 另一个视需要选用之功率控制器 70 可以与第一图形子系统 30 通信，并且可以调节由图形子系统 30 之组件所使用之时脉，譬如内存和像素时脉；禁能（或者断接）图形子系统 30 之功能区块；降低施加到图形子系统 30 之部分之电压；或者不然设置子系统 30 于其中功率消耗以已知方式减少之一个或多个模式中。

[0043] 虽然例示的图形子系统 40 形成在周边扩充卡 46 上，但是熟悉此项技术者将容易了解到图形子系统 40 仅能够容易形成在装置 10 之主机板上，或其它地方。

[0044] 接口电路 18 互连接较低速度周边装置和互连接，譬如光驱 28、和藉由整合 IDE/SATA 埠（未显示）方式于硬盘机形式之永久储存内存 34、和列表机，以及藉由并联或 USB 埠（未显示）方式之其它的周边装置。又其它的周边装置可以藉由较低速度扩充总线 24 之方式互连接，例如遵从已知的 PCI 或 ISA 标准。譬如声卡网络接口（未显示）之其它组件可以同样地藉由低速度扩充总线 224 之方式，或其它方式互连接至接口电路 18。

[0045] 如所提示的，装置 10 可以方便地形成为于膝上型或较小计算装置形式之可携式计算机装置。如此情况，单一外壳可以包含 DC 电源 38、显示器 26、和上述主机板和组件。第

二图形子系统 40 可以附加至容装计算装置之剩余部分之单一外壳，或者可以形成扩展坞之部分，该扩展坞当装置 10 实体互连接至其上时仅形成装置 10 之部分。

[0046] 装置 10 可操作于至少二种功率消耗模式：较高功率消耗模式和较低功率消耗模式。于所描述之实施例中，当装置 10 藉由连接到 AC（主）供电之电源 36 供电时，装置 10 可假设在该较高功率消耗模式；当装置 10 藉由使用一个或多个电池、燃料电池、等等 DC 电源 38 供电时，可以假设在该较低功率消耗模式。或可选择使用，功率消耗模式可以根据例如用户喜好、执行软件应用程序之类型、电池之位准、等等，而由用互选择、软件控制。

[0047] 于所描述之实施例中，装置 10 执行储存于系统内存内之软件 200，如图 4 中所例示。系统内存包含永久储存内存 34 和主存储器 14（图 2），并且可以进一步包含额外的随机存取内存、只读存储器和光盘储存内存之适当的组合，由装置 10 所使用以储存和执行软件 200。范例软件 200 能够例如储存于只读存储器中，或者从譬如光驱 28（图 1）之外部周边、或者经由计算机网络（未显示）加载。

[0048] 于所例示之实施例中，软件 200 系基于微软维斯塔 (Vista) 平台。然而，于本发明之范例实施例方式之软件操作装置 10 不需要根据此平台。取而代之，范例软件可以结合其它已知的计算机操作系统（譬如 Linux、MacOSX、或其它的操作系统）工作。用不同的操作系统，软件架构可以与图 4 中所描绘之软件架构在材料方面不同。

[0049] 于窗口维斯塔操作系统环境中，硬件组件（譬如图形子系统 30、40）之低阶层控制典型由一般称之为驱动程序之软件组件所执行。各硬件组件之操作由一个或多个此种组件所控制。

[0050] 驱动程序架构，于窗口维斯塔操作系统情况被称之为窗口装置驱动程序模型 (Windows Device Driver Model，WDDM)，并且更详细说明于微软窗口驱动程序备份工具软件应用程序接口 (Microsoft Windows Driver Kit API)。详细说明可以于下列网址获得：http://www.microsoft.com/whdc/devtools/WDK/AboutWDK.mspx 和 http://msdn2.microsoft.com/en-us/library/aa480220.aspx，该说明之内容由此加入作为参考。

[0051] 在解说图 4 中所例示之软件 200 之操作之前，适合先说明习知的窗口维斯塔 (Windows Vista) 架构。欲达此目的，图 3 描绘根据窗口维斯塔操作系统之习知的软件 200'。如所例示，软件 200' 包含应用程序软件 202'、数个属性、操作系统图形组件 204'、206'、208'、210'（称之为运作时间 (run-time) 组件）；以及数个硬件特定图形装置驱动程序组件 220、222、和 224，定义装置驱动程序用于图形子系统，像是子系统 30 和 40。而且，亦说明显示加载器模块 218'、窗口维斯塔核心 216'、和窗口图形装置接口 (GDI) 软件 214'。

[0052] 如将了解到，于窗口维斯塔操作系统下之驱动程序软件能够运作于用户模式或核心模式其中任一情况。用户模式驱动程序 (User-mode driver，UMD) 组件执行于由处理器 12 所支持之非优先的模式，并且仅被允许限制存取于内存、缓存器、等等。于图 2 中，组件 220 和 222 为 UMD 组件。其它的软件，包含应用程序软件 202'，亦执行于此模式。UMD 组件 220 和 222 和应用程序软件 202' 不能获得直接存取至低阶层系统资料。他们同样不能够存取内存之所有的区域、所有的缓存器、等等。然而他们可以呼叫至执行于核心模式之软件。

[0053] 相比之下，核心模式驱动程序 (kernel-mode driver，KMD) 组件执行于与该操作系统核心相同的处理器模式，而因此一般已释放存取至装置 10 之内存、缓存器、和其它资源。UMD 组件可以呼叫 KMD 组件，以获得较低阶层存取至计算装置 10 之资源。组件 224 为 UMD 组

件。KMD 驱动程序架构进一步详细说明于 2006 年 5 月 10 日提出申请之“用于核心模式驱动程序框架之样本驱动程序 (Sample Drivers for the Kernel Mode Driver Framework)”，可以从下列网址获得，http://www.microsoft.com/whdc/driver/wdf/KMDF-samp.mspx，以及于 WDM 导论，网址为 http://msdn2.microsoft.com/en-us/library/aa4490246.aspx。

[0054] UMD 组件和 KMD 组件具有不同的结构、不同的登录点、和不同的系统接口。装置是否需要 UMD 或 KMD，依于装置之类型和于操作系统中已经对其提供之支持而定。用于图形子系统（譬如图形子系统 30、40）之驱动程序典型上包含至少一个运作于核心模式之组件。再者，于窗口维斯塔操作系统中，KMD 组件被加载于系统初始化（亦即，升高供电），而 UMD 组件当需要时可经要求而加载。

[0055] 详言之，于窗口维斯塔架构中，为了获得低阶层存取于由这些子系统所使用之资源，用于图形子系统之 KMD 组件与对应之 KMD 组件通信。典型的情况是，各 KMD 组件提供装置驱动程序接口 (device driver interface,DDI)，对于对应之 UMD 组件为已知。DDI 可以包含功能呼叫、目标（包含方法）、输入 / 输出控制 (input/output control, IOCTL) 呼叫和关联之数据结构。如将了解的，功能呼叫透过其时常组构于此种数据结构中之功能 / 方法参数接收资料。此数据结构之正确性质特定于 DDI 定义中。

[0056] 于描述之实施例中，操作系统图形组件 204’、206’、和 208’ 为由操作系统制造商（如此情况为微软公司）所提供之运作时间组件；而于 UMD 和 KMD 组件形式之硬件特定图形装置驱动程序组件 220、222、和 224 可以由第三方制造商（譬如图形子系统 30 或 40 之制造商）提供。

[0057] 操作系统图形组件包含运作时间 3D 图形运作时间组件 204’（直接 3D 运作时间）、硬件加速图形接口运作时间组件 (DXVA)206’、3D 开放式图形库 (Open Graphics Library, OpenGL) 运作时间图形组件 208’。对应之硬件特定 UMD 组件 220 和 222 提供对应于直接 3D、DXVA、和开放式 GL API 之 API 硬件呼叫之硬件特定执行。

[0058] 运作时间组件 204’、206’、208’ 和 UMD 组件 220、和 222 提供联合的图形应用程序软件规划器接口 (API)，由应用程序软件 202’ 和操作系统之剩余的部分使用。详言之，API 提供功能呼叫、目标、等等，其引起于 UMD 组件 220’ 和 222’ 或于运作时间组件 204’、206’、208’ 中驱动程序软件例程（例如，功能或方法）之执行，如由微软公司详细说明。API 呼叫可以藉由部分之操作系统（例如，模块 204’、206’、208’），或藉由其它的驱动程序由应用程序软件 200’ 完成。UMD 组件 220 和 222 对应于 API 呼叫依次执行软件码（典型于功能、或目标方法之形式）。由 D3D、DXVA 和开放式 GL UMD 组件 220 和 222 所提供之功能和回呼详细说明于窗口驱动程序备份工具软件：显示装置窗口维斯塔显示驱动程序模型参考 (Display Devices Windows Vista Display Driver Model Reference)，其可从微软研发之网络 (Microsoft Developer’s Network)（网址：www.msdn.com）取得，该软件内容由此结合入本说明书作为参考。

[0059] 详言之，于 UMD 组件 220、222 或 KMD 组件 224 由操作系统之加载例程加载后，定位该驱动程序组件登录点 - 著名的驱动程序登录 ()(DriverEntry())（用于核心模式驱动程序）或者 DII 主要 ()(DIIMain()) 或者否则其它方面（例如，开放式适配器 ()(OpenAdapter())）用于 UMD 组件 220、222。UMD/KMD 组件 220、222、224 包含从 OS 接收已熟知的结构功能，该 OS 由在 UMD/KMD 组件 220、222、224 之登录点之码所填满，指向在执行所

期望之性能之 UMD/KMD 组件 220、222、224 内之功能。这些名称由 DDI 规格所定义，并且透过所称为之定义档案来执行。

[0060] 例如，KMD 组件 224 接收驱动程序_初始化_资料 (DRIVER_INITIALIZATION_DATA) 结构，该结构包含一组用于其为 DDI 之驱动程序执行功能组之部分之多种操作之功能指针。当需要时，UMD 组件 220、222 之剩余的部分可以呼叫这些功能以起始在驱动程序中之适当的操作，该驱动程序于许多情况（但是并非需要全部）将引致存取至硬件（通常藉由呼叫一些其它的 KMD 驱动程序内部功能）。

[0061] UMD 组件 220 和 222 可以形成为动态链接库 (dynamic linked library，DLL)，并且顺应 WDDM。就此种情况，各 UMD 组件 220、222 依照 WDDM 之 IOCT 提供收集之功能和目标。概略而言，各驱动程序组件包含定义的登录点驱动程序登录 (DriverEntry())、定义的目标等级、驱动程序登录点、定义的功能和回呼叫。加载各驱动程序后，于其登录点之软件码被执行 DriverEntry()，并且定义之驱动程序例程被暂存于预期的结构中。

[0062] DIIMain() 登录点用来分配和初始 UMD 组件之基本数据结构，或者当卸载 UMD 时于控制之机构卸除这些组件。

[0063] 于 WDDM UMD 组件之情况，OpenAdapter() 于与 DriverEntry() 呼叫 KMD 组件相似之方式呼叫工作。也就是说，OpenAdapter() 呼叫接收具有一组 UMD 组件 220、222 填满指向 UMD 组件内适当功能之功能指针之数据结构。

[0064] 在 UMD 内名称和支持之功能 / 目标和关联之码地址藉由此结构方式因此被返回至操作系统之剩余的部分。此外，由 D3D、DXVA 和 OpenGL UMD 组件 220 和 222 支持之 UMD 功能和结构被详细说明于窗口驱动程序备份工具软件 :显示装置窗口维斯塔显示驱动程序模型参考，其可以从微软研发之网络“supra”取得。

[0065] 于是，于加载各 UMD 组件 220、222 之结束，API 功能 / 目标，和 IOCTL 对于应用程序软件 202’和操作系统软件之剩余的部分为已知。API 目标可以由应用程序软件 200 藉由于相同的地址创造目标之例子而引证。能够执行功能 / 方法和 IOCTL 于他们的对应地址。

[0066] 操作系统图形模块进一步包含核心模式图形核心软件组件 210’（称之为于窗口维斯塔操作系统中直接 X 核心 (DirectX Core))、和 KMD 组件 224。图形核心软件组件 210’提供 API 用于 UMD 组件 220、222，允许这些 UMGLD 组件 220、222 获得核心模式存取至装置 10 之资源。核心模式图形核心软件组件 210’可以进一步包含视频内存管理器、排程器、和转换（或编辑）某 API/DDI 呼叫用于兼容性之例程。KMD 组件 224 可以符合窗口驱动程序模型，或窗口驱动程序框架，如上述详细说明。如此情况，KMD 组件 224 包含定义的目标等级、功能和结构，提供 DDI。

[0067] 像 UMD 组件 220 和 222，KMD 组件 224 包含初始例程、驱动程序登录 ()，该驱动程序登录 () 典型藉由名称和内存地址返回目标等级、功能和结构之识别符，提供所需的 DDI 用于 KMD 组件 224。如所提及的，KMD 组件 224 典型于系统起动被加载（和初始化）。

[0068] 此外，KMD 组件 224 可以包含 DDI 未明确已知或报告于操作系统之剩余的部分，但是已知于互补的 UMD 组件 220 或 222。

[0069] 软件 200 被层化，具有较高阶层层使用较低层提供某些功能。那么，应用程序软件 202 藉由制作呼叫至运作时间操作系统图形组件 204’、206’、208’、或 UMD 组件 220、222、或 GDI 214’而典型使绘制图形。图形模块 204’、206’、和 208’仅包含属性、共享于所有第三

方视频驱动程序之硬件独立码。运作时间组件 204’、206’、208’依次可以制作 API/DDI 呼叫至 UMD 组件 220 和 222。定义于数据结构之已知的参数例如藉由至所存在的结构之适当的指针而被递送至 UMD 组件 220 和 222。UMD 组件 220、222 包含硬件特定码、和结构。然而，如所提及，UMD 组件 220、222 仅仅具有用户阶层存取。UMD 组件 220、222 与 KMD 组件 224 通信，直接使用由 KMD 组件 224 提供之已知的 API/DDI，或者透过核心模式图形核心软件组件 210’。

[0070] KMD 组件 224 依次包含功能、目标、等等，其能够传递适当的低阶层指令至位于下方硬件（例如，图形子系统 30 或 40）。可以进一步排列多重呼叫至 KMD 组件 224。低阶层指令依次可以由位于下方硬件执行。举例而言，低阶层指令可以包含可执行指令等之图形处理器。

[0071] 不像许多其它已知之操作系统，窗口维斯塔仅允许加载单一显示驱动程序 KMD 组件 224’。其它补助的应用程序软件用作为显示驱动程序加载器 218’。加载器 218’一般依于起动而执行，但是亦可以于起动后执行。加载器 218’加载第三方 KMD 组件（例如，组件 224’），并且将其初始化，用来由图形核心软件组件 210 所存取。虽然可以使用加载器 218 加载 / 卸载核心模式驱动程序组件，像是 KMD 组件 224’，但是一个图形 KMD 组件 224 之加载，需要另一个驱动程序之卸载。

[0072] 现在，于存在之二个图形子系统中，像子系统 30、40（第 1 图），UMD 组件 220 和 222 以及 KMD 组件 224 可以控制二个图形子系统之操作，并且允许选择性地切换该二个图形子系统之间之操作。至驱动程序组件 220、222、和 224 之 API 呼叫可以确认多个子系统之哪一个将被寻址。然而，单一 UMD/KMD 提供支持多个图形子系统之要求引入限制。举例而言，对于单一驱动程序支持很多项之不同的适配器那是不切实际的。若子系统由不同的制造商提供则恶化了此问题。

[0073] 图 4 因此显示了本发明之范例实施例之软件和图形模块。再者，范例软件描绘于窗口维斯塔环境之情况。如此情形，形成操作系统之部分之模块和组件相同于描绘于图 3 中者，并且因此于图 4 中用相同的数字（但是并没有（’）符号）描述，而将不作进一步之详细之说明。

[0074] 然而，值得注意的是，UMD 组件 220 和 222 以及 KMD 组件 224（图 3）已经各被代理器驱动程序组件取代 --UMD 代理器组件 320、322 和 KMD 代理器组件 324。如将变得很清楚，UMD 代理器组件 320、322 和 KMD 代理器组件 324 出现于操作系统之剩余的部分，单一组之 APIs/DDI，以及表现为单一图形驱动程序。如此情况，运作时间组件 204、206、208、应用程序软件 202、和操作系统之剩余的部分可以请求（或呼叫）图形 API 功能 / 目标，如图 3 中所示。

[0075] 也对附加的功率控制应用程序软件 201 的功能加以描述。可清楚了解，功率控制应用程序软件 201 可以控制图 1 之图形子系统 30、40 之全部功率消耗状态。功率控制应用程序软件 201 可以是独立执行之应用程序软件，或者可以形成全部用户 / 图形子系统控制和组构应用程序软件 -- 譬如触媒控制中心应用程序软件（可以从 ATI/AMD 公司构得）之一部分。

[0076] UMD 代理器组件 320 和 322 以及 KMD 代理器组件 326 依次路由呼叫此种图形功能至多个个别硬件特定 UMD 图形驱动程序组件 340、342、350、352、以及 KMD 图形驱动程序组

件 370 和 372。详言之，UMD 代理器组件 320 路由呼叫至 UMD 组件 340 或 342 ;UMD 代理器组件 322 路由呼叫至 UMD 组件 350 或 352 ;以及 KMD 代理器组件 324 至 KMD 代理器组件 360 或 KMD 代理器组件 362，如下文之详细说明。

[0077] UMD 组件 340、350 和 342、352 为对应于图形子系统 30 和 40 之硬件特定 UMD 组件，该图形子系统 30 和 40 像 UMD 组件 220 包含功能、目标、IOCTL、等等，该等功能、目标、IOCTL 等为硬件特定 -- 分别至图形子系统 30 和 40。KMD 组件 360、362 同样相似于 KMD 组件 224，并且包含功能、目标、IOCTL、等等，设计于核心模式与图形子系统 30 和 40 互动。

[0078] 举例而言，UMD 组件 340、350 和 KMD 组件 360 可以分别提供用户模式 DirectX 驱动程序软件、OpenGL 驱动程序软件、和核心模式驱动程序软件组件用于第一图形子系统 30 ；同时 UMD 组件 342、352 和 KMD 组件 362 可以提供用户模式驱动程序软件、OpenGL 驱动程序软件、和核心模式驱动程序软件用于第二图形子系统 40。习用上，组件 340、350、和 360（或者组件 342、352、362）可以是习知的，于此习知的组件中他们可以由图 3 中所描绘之操作系统直接加载，取代图形 UMD/KMD 组件 220、222、和 224。

[0079] 于此种方式，各 UMD/KMD 组件 340、342 ;350、352 ;360、362 可以实施低阶层图形功能，并且以特定于包含之图形硬件之方式存取图形硬件。

[0080] 习用上，UMD 代理器驱动程序组件 320、322 和 KMD 代理器驱动程序组件 324 一方面表现一致的 API/DDI 于操作系统之剩余的部分，另一方面，路由呼叫、IOCTLs、请求、存在目标、等等（集体 API/DDI 呼叫）至硬件特定 UMD/KMD 驱动程序组件 340、350、和 360，或者 342、352、和 362。

[0081] 于操作中，UMD 代理器驱动程序组件 320、322 和 KMD 代理器驱动程序组件 324 由操作系统之剩余的部分加载。

[0082] 一旦加载 UMD 代理器驱动程序组件 320、322，则执行其登录例程（例如，DIIMain()/OpenAdapter()）。如将变得很清楚，UMD 代理器驱动程序组件 320、322 之登录例程加载 UMD 组件 340、342 之一者或二者，并且提供至维斯塔操作系统之剩余的部分，期望的数据结构确认于 UMD 代理器驱动程序组件 320、322 中支持之 API/DDI 呼叫与 UMD 组件 222 和 224 所作非常相同的方式。再者，期望之 API/DDI 之地址以地址之形式藉由预期之结构之方式提供至功能、目标、IOCTLs。

[0083] UMD 组件 340 和 342 由 UMD 代理器驱动程序组件 320 内之软件码所加载，如更详细说明于图 5 中方块步骤 S500 中。详言之，于方块 S502，加载譬如 UMD 组件 340 或 342 之 UMD 驱动程序组件，典型作为动态链接库。其次，可以藉由 UMD 代理器驱动程序组件 320 呼叫新加载 UMD 组件 340 或 342 之驱动程序初始化例程 DIIMain()/OpenAdapter()、等等。于方块 S504 中，新加载 UMD 组件 340 或 342 之驱动程序初始化例程返回返回支持功能、IOCTLs 等之名称和地址（于与 UMD 组件 220、222 返回此等名称、地址等相同的方式）至 UMD 代理器组件 320。其次，于方块 S508 中，UMD 代理器驱动程序组件 320 形成表示将由负载之 UMD 组件 340 或 342 所支持之目标等级、功能、IOCTLs、等等之间一致性之数据结构。

[0084] 举例而言，UMD 代理器驱动程序组件 320 被设计成支持 DXVA 功能呼叫和目标，并且因此形成此种已知 DXVA 功能呼叫和目标与在加载之 UMD 组件 340 或 342 内他们的登录点之间之一致性。于方块 S506 之结论，UMD 代理器驱动程序组件 320 可以形成于内存中结构，该内存中设有对应于由 UMD 代理器驱动程序组件 320 所支持之各支持之 DXVA 功能、目

标、IOCTLs 等之于 UMD 组件 340 或 342 中之地址。

[0085] 可以藉由 UMD 代理器驱动程序组件 320 和 322 或有需要时动态地加载特定的 UMD 组件 340 或 342、350 或 352。尤其是，若任何图形子系统 30 和 40 未使用，则其对应之 UMD 驱动程序组件不需被加载。

[0086] 对于待由 UMD 代理器驱动程序组件 320 加载之各 UMD 驱动程序 340 或 342 可以实施对应于方块步骤 S500 之软件。步骤 S500 由用作为用于 UMD 组件 350 和 352 之代理器驱动程序组件之 UMD 代理器驱动程序组件 322 以类似方式实施。

[0087] 而且，亦在 KMD 代理器驱动程序组件 324 之初始化后 -- 典型为系统起动时，实施方块步骤 S500。KMD 代理器驱动程序组件 324 执行 KMD 驱动程序组件 360 和 362（例如，OpenAdapter()）之初始化例程。

[0088] 一旦已藉由各 UMD 代理器驱动程序组件 320、322 和藉由用于各支持之驱动程序组件（例如，UMD 组件 340、342 ；UMD 组件 350、352 ；和 KMD 组件 360、362）之 KMD 代理器驱动程序组件 324 实施方块步骤 S500（或者他们的相等步骤），则 UMD/KMD 代理器驱动程序组件 320、322 和 324 将已经加载特定于图形子系统 30、40 之 UMD/KMD 组件，并且将已确定于加载之 UMD 组件 340、342、350、352，和 KMD 组件 360、362 中支持之功能、IOCTLs、等等之对应之内存地址 / 登录点。

[0089] 为了提供系统支持信息至操作系统之剩余的部分之目的，仅仅 KMD 代理器驱动程序组件 324 将可直接看到和可安装于二者子系统 30、40。可以合并用于二者子系统 30、40 之驱动程序特定登录项目，因此代理器驱动程序组件可以读取和暴露该等登录项目至 UMD 组件 340、350 ；342、352。

[0090] UMD 组件 340、350 ；342、352 亦可以返回由 UMD 代理器组件 320、322 所聚集和调整之许多的特质，以确保返回之特质与用来与多个图形子系统互动之单一驱动程序一致。这些特质可以由 UMD 代理器驱动程序组件 320、322 传递至操作系统。举例而言，视频内存蓄积 (Video memory heap)、GPU 引擎特质、DMM 形态、等等也许需要由 UMD 代理器组件 320、322 结合。

[0091] 作为另一个替代者加载 UMD/KMD 组件 340、342 ；350、352 ；和 360、362，当这些组件被建立时，这些组件能够以统计方式链接至 UMD 代理器驱动程序组件 320、322，和 KMD 代理器组件 324。此当然将需要存取至用于 UMD/KMD 组件 340、342 ；350、352 ；和 360、362，或者可链接目标模块之来源码。

[0092] 结果，UMD 代理器组件 320、322，和 KMD 代理器组件 324 加载 / 链接驱动程序 UMD 组件 340、342 ；350、352 ；和 360、362，UMD 代理器驱动程序组件 320、322，和 KMD 代理器 324 现在可以依于哪一个图形子系统 30 或 40 正由应用程序软件 202 或操作系统之剩余的部分寻址，而路由 API/DDI 呼叫至 UMD/KMD 代理器组件 320、322 和 324 至个别的 UMD/KMD 组件 340 或 342 ；350 或 352 ；和 360 或 362。此以图形方式例示于图 6 中。

[0093] 由 UMD 代理器 320、322 和 KMD 代理器 324 所支持之 API/DDI 呼叫之细节可以藉由存在具有路由例程之细节用来支持 API/DDI 呼叫之数据结构而暴露于剩余的应用程序和操作系统，该 API/DDI 呼叫由 UMD 组件 340、342 ；350、352 ；和 KMD 组件 360、362 实际实施。

[0094] API/DDI 呼叫可以藉由这些路由功能（或目标）而路由在 UMD/KMD 代理器驱动程序组件 320、322 和 324 内。各路由例程之地址或 UMD/KMD 代理器驱动程序组件 320、322 和

324 之目标与待路由之特殊的 API 功能 / 呼叫 / 目标 /IOCTL、等等相关联而可以暴露于操作系统和其它的应用程序软件。各路由例程或目标依次路由呼叫至 UMD/KMD 驱动程序组件之其中之一，对于该驱动程序组件该代理器驱动程序组件用作为代理器。于此种方式，API/DDI 呼叫至代理器驱动程序组件 320、322 和 324 可以被路由至在 UMD 驱动程序 340、342 或 350、352 或 KMD 组件 360、362 中对应之驱动程序功能 / 呼叫 / 目标 /IOCTL。

[0095] 详言之，如图 7 中所例示，各 API/DDI 呼叫可以根据在 UMD 组件中对应之功能 / 呼叫 / 目标 /IOCTL 之地址而由 UMD/KMD 代理器驱动程序组件 320、322 和 324 仅仅被重新路由，对于该地址对应之登录点于方块步骤 S700 中于方块 S508 中已经被决定。详言之，于方块 S702 中，依于步骤 S700 之执行，UMD/KMD 代理器组件 320、322 或 KMD 代理器组件 324 决定哪一个 UMD/KMD 组件（例如，UMD 组件 340、342 ;350、352 ;或 KMD 组件 360、362）将处理该 API/DDI 呼叫。此情形例如可以藉由剖析该 API/DDI 呼叫或者关联之资料以确认有关的图形子系统，或者仅仅藉由决定多个图形子系统之哪一个现在正在使用而实施。如下文之详细说明，现在正在使用之图形子系统 30 或 40 可以根据装置 10 而取决。现在正在使用之子系统典型绘成待显示和由终端用户观看之图形 / 视讯。

[0096] 一旦已经确认了有关的驱动程序，则 UMD/KMD 代理器驱动程序组件 320、322 或 KMD 组件 324 决定关于形成于方块 S706 中方块 508 中一致结构之 API/DDI 呼叫之地址。一旦已经决定了该地址，则可以作成在有关的 UMD/KMD 组件 340 或 342 ;350 或 352 ;360 或 362 内之 API/DDI 呼叫。然后 UMD/KMD 组件执行对应于该 API/DDI 呼叫之码。

[0097] 值得注意的是，至 UMD 驱动程序 340、342 或 350、352 之呼叫可以制造其它的 API/DDI 呼叫。这些可以导致 API/DDI 呼叫至 KMD 代理器组件 324。KMD 代理器组件 324，像 UMD 代理器组件 320、322 路由 DDI 呼叫至适当的 KMD 组件 360、362，如图 7 中详细说明。KMD 代理器组件 324 可以决定 KMD 组件 360、362 之哪一个组件，核心模式 API/DDI 呼叫将被路由与 UMD 代理器组件 320、322 制造评估非常相同的方式 -- 亦即，依于呼叫之性质或者依于现时主动的图形子系统。

[0098] 对于伴随着资料之呼叫（例如，透过创造之目标、或者藉由功能参数之方式），可以藉由 UMD 代理器组件 320、322 或 KMD 代理器组件 324 递送指向资料之指针至决定之 UMD 组件 340、342、350、352 或者 KMD 组件 360、362。若资料不能被递送为指针，则该资料传隧至对应之资料节构。

[0099] 一些 API/DDI 呼叫可以不被操作系统之剩余的部分知道，并且根据初始化（例如，依于执行之 DllMain() 或 AdapterQpen()）情况而不返回到 UMD 代理器驱动程序组件 320、322 或 KMD 代理器驱动程序组件 324。此情况对于譬如 KMD 组件 360、或 362 之 KMD 组件也许尤其显著，该 KMD 组件 360、362 与例如由单一制造商 / 供货商提供之譬如 UMD 组件 340、342 之互补 UMD 组件互动。虽然可以支持 DDI 呼叫，但是他们不需要暴露于操作系统。此种呼叫能够典型不由 KMD 代理器驱动程序组件 324 路由，而没有进一步之知晓。欲避免此情况，各 KMD 组件 360、362 将报告所有支持之 API/DDI，该 API/DDI 例如可以包含于用于操作系统排程器情况切换和寻呼请求通信之一般操作系统中，反应于执行驱动程序初始化例程。在此为不可能之情况下，KMD 组件 360、362 可以进一步包含询问例程以返回关于将由 KMD 代理器驱动程序组件 324 要求之 API/DDI 呼叫之信息。

[0100] 代理器驱动程序组件亦将接收来自 UMD/KMD 组件之呼叫，该 UMD/KMD 组件对 API/

DDI 呼叫处理以确保其能够追踪影响图形子系统 30 和 40 之任何状态改变。

[0101] 图 8 显示图 2 之装置 10 之部分之进一步简化方块图，于此图中可以使用图 4 之软件 200（以及尤其 UMD 代理器组件 320、322 和 KMD 代理器组件 324）。如所例示，接口电路 16 互连接中央处理器 12 和系统内存 14。图形子系统 30（具体实施为在接口电路 16 上之图形处理器）包含图形引擎 32、内存控制器 72、显示器接口 74、和总线接口 78。

[0102] 图形引擎 32 为能够绘成 2D 图形或 3D 图形译码视频、等等之功能的区块。如将了解到，图形子系统 30 可以包含多个图形引擎。

[0103] 内存控制器 72 允许图形子系统 30 提供存取至图形内存和主存储器 14。于所描绘之实施例中，由图形子系统 30 所使用之图形内存形成主存储器 14 之部分。然而，熟悉此项技术者将容易了解到，图形子系统 30 可以包含或者结合其自己的局部内存。总线接口 78 致能子系统 30 经由总线 20 通信。

[0104] 如将了解到，显示器接口 74 可以是用来转变显示于由埠 78 互连接之显示装置 26 上缓冲器内资料之任何适合的接口。举例而言，显示器接口 74 可以采用随机存取内存、数字至模拟转换器 (RAMDAC) 之形式。一个或多个视频连接器允许图形子系统 30 互连接至一个或多个显示装置，譬如 LCD 面板、监视器、电视机、等等。输出埠 78 可以是在 VGA 埠、混合的视频埠、DVI 埠、LVDS 埠、DVO 埠、SDVO 埠、等等之形式。

[0105] 图形子系统 40（例如，形成在图 2 之周边扩充卡 46 上）亦藉由在高速扩充总线 20 上之扩充槽之方式连接至接口电路 16。图形子系统 40 包含图形引擎 42、内存控制器 52、总线接口 58、和显示器接口 54。图形子系统 40 包含图形内存 50，或者与图形内存 50 通信。

[0106] 图形引擎 42，像图形引擎 32，为能够绘成 2D 图形或 3D 图形译码视频、等等之功能的区块。如将了解到，图形子系统可以包含多个图形引擎。可能的情况是，图形引擎 42 可以提供不仅由图形引擎 32 所提供之功能。

[0107] 内存控制器 52 允许图形子系统 40 存取内存 50 和主存储器 14。总线接口 58 致能图形子系统 40 经由总线 20 通信。

[0108] 显示器接口 54 藉由内存控制器 52 之方式取样于图形内存 50 中之帧缓冲器，并且表现图像于视频连接器。于此种方式，可以显示由外部的图形引擎 42 绘成于内存 50 中帧缓冲器中之图像。视频连接器可以直接连接至外部显示器，或者至装置 10 之主机板，此处视频讯号可以路由至整合的显示器，或者用来附加外部显示器至装置 10 之连接器。再者，显示器接口 54 可以是任合适合的接口，用来转变在缓冲器内的资料用来显示于显示装置 32 上，譬如 RAMDAC、单端或不同的发送器、等等。

[0109] 如所提及的，功率控制器 60 系与图形子系统 40 通信，并且使用习知的功率消耗技术，譬如时脉和电压调节、降低供电、或者不然禁能所有的或一些的该等组件，来控制显示器接口 54、内存控制器 52、图形引擎 42、总线接口 58、和图形内存 50 之各个或某些和其中的一个或多个之功率消耗。功率控制器 60 可以藉由于总线 20 上的讯号或者其它的方式控制，并且例如可以与 ACPI 标准兼容。

[0110] 图形子系统 30 与图形子系统 40 以非常相似的方法操作。如此情况，图形子系统 30 使用内存控制器 72 存取保持于主存储器 14 或者于图形子系统 30 之局部之内存中之帧缓冲器。此帧缓冲器由显示器接口 74 取样，并且图像表示于视频输出连接器，该连接器能够直接连接至显示器。在致力于提供价廉之整合的组件情况，图形子系统 30 提供有限的功

能。举例而言，图形子系统 30 之分辨率、内存、图形处理器速度、3D 图形能力、等等也许相当地受限制，并且也许较图形子系统 40 之外部图形处里器 42 操作更慢。

[0111] 藉由图形子系统 40 而更有效地实施譬如三维图形、游戏图形、等等之计算密集图形。因此在装置 10 内附加图形子系统 40 之使用允许终端用户经验最新的图形密集应用程序，譬如游戏、计算机辅助设计软件、动画软件、绘成软件 (rendering software)、等等。方便来说，可以由终端用户选择附加上的图形子系统 40，并且当需要时被取代和保持现用。在过去，额外的图形计算能力仅在工作站计算装置可取用。于移动式计算装置上扩充槽问世后，现在此种计算能力能够于譬如膝上型计算机之可携式计算机之拥有者所使用。当然，使用在图形子系统 40 上较高（或不同）性能图形引擎 42 增加装置 10 之全部功率消耗。此增加之功率消耗在由电池电源供电之计算装置上也许不足以支撑。

[0112] 同时，在具有图形引擎 42 之额外的图形子系统 40 存在的情况下，则图形子系统 30 也许是多余的。举例而言，若多个实体显示器未连接至装置 10，则图形子系统 30 也许不会发挥作用。图形子系统 30 因此可以被禁能。或可取而代之，在控制图形子系统 30 之操作之存在之功率控制器 70 中，当图形子系统 30 未使用时，其也许亦被设置在较低功率模式。再者，功率控制器 70 可以禁能或断接部分之图形子系统 30，或者图形子系统 30 之时脉或电压调节部分。

[0113] 本发明之范例实施例，软件 200（图 4）用来允许装置 10 选择性地禁能存在于子系统 30 中之一个较高的功率图形子系统 40。

[0114] 欲达此目的，以及如图 8 中所示，计算装置 10 进一步包含开关 56。开关 56 接收由子系统 40 和子系统 30 于第一和第二输入所产生之视频讯号。开关 56 可以是任何适当的视频开关，譬如多功器，并且用于表示在其视频输出连接器之在其二个讯号输入之习知的视频讯号其中之一。表示之视频讯号于开关 56 之输入可以是习知的视频讯号，譬如数字讯号（譬如 LVDS 或 TMDS 格式等）或者模拟讯号（譬如 VGA 格式）。若组构开关 56 以接收数字和模拟输入讯号，或者提供视频于任一的输出，则开关 56 可以包含格式转变器。而且，开关 56 可以包含一个或多个视频输出，使可以连接数字和模拟显示装置 32 其中任一者或者二者。

[0115] 开关 56 进一步包含控制输入 (CNTRL)。此控制输入控制哪一个讯号输入被提供于开关 56 之视频输出。于所描绘之实施例中，控制输入由处理器 12 反应于所要求或希望之装置 10 之功率模式中侦测或决定改变，藉由一般目的输入输出 (general purpose input output，GPIO) 接口（未显示）之方式切换。如将变得很清楚的，若装置 10 正操作于较低功率消耗模式，则开关 56 组构成使得选择由图形子系统 30 所产生的习知的视频讯号。反之，若装置 10 正操作于较高功率消耗模式，则选择由较高性能外部图形子系统 40 所产生的视频讯号用于显示。同样情况，可以减少或禁能提供至图形子系统 40 或图形子系统 30 之功率。切换可以动态地实现，同时计算装置 10 是在使用中，而不需要装置 10 重新起动（亦即，冷或热起动）。

[0116] 欲完成此目的，计算装置 10 亦可以包含至少一个上述之功率控制器 60。于此描述之实施例中，功率控制器 60 形成装载图形子系统 40 之周边扩充卡 46 之部分。然而，功率控制器 60 仅亦能够形成计算装置 10 之主机板之部分，或者作为接口 16 之部分。若功率控制器 60 形成扩充卡 46 之部分，则其可以具有较大的弹性控制子系统 40 之操作。若功率控

制器 60 形成计算装置 10 之部分，则其可以仅具有禁能供电至图形子系统 40 之能力。

[0117] 为了组构和控制开关 56 和功率控制器 60，使用在系统内存 12 内之软件 200。图 9 因此为流程图，例示本发明之范例实施例中范例软件方块步骤 S900 用来切换装置 10 于二个有效的功率消耗模式之间。

[0118] 现在，本发明之范例实施例，当装置 10 被起始供电时，评估装置 10 之功率状态。当需要时，功率控制应用程序软件 201 组构图形子系统 30 和 40 和开关 56，以及如下之详细说明。

[0119] 可以藉由处理器 12 于系统内存 10 内功率控制应用程序软件 201 之控制下实施本发明之范例实施例软件方块步骤 S900。每次装置 10 经历状态改变可以实施方块步骤 S900，对于此改变将因此配置图形子系统 30 和 40。如所例示，于方块 S902 中功率控制应用程序软件 201 决定是否装置 10 将假定其较高功率消耗模式，或其较低功率消耗模式。

[0120] 应用程序软件 201（图 4）可以维持指示图形子系统 30、40 之哪一个系用于特定图形功率状态之较佳装置之表。各用户功率状态映对至图形子系统 30、40，并且对应用于该子系统之状态。用于个别图形子系统 30、40 之功率状态可以操作现时主动图形子系统 30 或 40 之参数，譬如时脉速度、内存速度、像素时脉速度、绘成能力、等等。

[0121] 于范例实施例中，当装置 10 正由 AC 电源 36 操作时，可以使用较高功率消耗模式；当装置 10 由 DC 电源供应器 38 之方式操作，而当 DC 电源 38 为低、或等等时，可以使用较低功率消耗模式。于较高功率消耗模式，图形子系统 40 为主动的。于较低功率消耗模式，图形子系统 30 为主动的。

[0122] 若装置 10 再开始（或者转变至）其高功率消耗模式，则执行方块 S904 至 S910。于方块 S904 若子系统 40 尚未设置于其全操作（高功率消耗）模式，则将其设置在此模式。此可以藉由适当的驱动程序呼叫实施 --（对于 AMD 或 ATI 图形子系统可以使用习知的 ATPX ACPI 方法，可以使用相似的方法或功能于驱动程序用于来自其它制造商之图形子系统），藉由处理器 12 提供适当的讯号至功率控制器 60。其次，于方块 S906 致能子系统 40。此可以藉由使用于组构管理器 (Configuration manager) 中窗口 PnP 呼叫 (Windows PnP call) 致能子系统 40 实施。一旦子系统被致能，则操作系统可以列举所有的装置以获得新的装置名称。为了此目的可以使用窗口列举显示装置 ()API 功能 (Windows EnumDisplayDevices()API function)。其后，可以使用窗口改变显示设定 EX() API 呼叫 (Windows ChangeDisplaySetting()API call) 创造具有二种装置之扩充的桌上型计算机。应用程序软件 201 现在可以感知适配器之数目已经改变（例如，藉由接收 WM_DISPLAYMODECHANGE 讯息）。应用程序软件 201 可以发出重新起动命令。可以使用对于 ATI/AMD 图形子系统用于驱动程序功能之 CWDDEDI、或者使用来自其它制造商对于图形子系统用于驱动程序之相似的功能、以及禁能整合之 I^2C 控制器、使用控制方法呼叫之禁能 I^2C 缓冲器，所有这些于方块 S906 中关断显示器电源。于方块 S908 中可以组构开关 56 以选择从图形子系统 40 之输出，例如使用 ATPX ACPI 方法用于 ATI/AMD 图形子系统，或者使用来自其它制造商之对于图形子系统用于驱动程序之相似的功能。于方块 S910 中，可以逻辑方式致能新致能之图形子系统 40，而使得对于操作系统之剩余的部分为可观察的。此可以使用 Windows ChangeDisplaySettingEX()API 呼叫完成。而且，可以藉由从显示之桌上型计算机卸下子系统而逻辑上禁能图形子系统 30。此可以使用 Windows ChangeDisplaySettingEX()

API 呼叫完成。当由操作系统询问时，可以作成另外隐私的 API（避开）呼叫至驱动程序以隐藏整合的显示，由此隐藏该禁能之图形子系统。

[0123] 如所提及的，于 Vista 下，仅仅一个图形 KMD 能够是主动的。欲支持通常由二个不同的装置驱动程序控制之二个图形子系统 30、40，KMD 代理器驱动程序组件 324 用作为用于二个图形子系统 30、40 之核心模式驱动程序。同样情况，UMD 代理器驱动程序组件 320、322 用作为单一 UMD。

[0124] 当装置 10 转变至，或者重回至其低功率消耗模式时，执行方块 S912 至 S918。广言之，图形子系统 40 被禁能和设置在其低功率消耗模式，同时致能图形子系统 40。欲达成此情况，于方块 S912 和 S914 中致能图形子系统 30。再者，可以藉由在方块 S912 逻辑致能关联于图形子系统 30 之显示，和于方块 S914 逻辑禁能有关子系统 40 之显示，而实施此情况。可以藉由适当的操作系统 API 呼叫，譬如上述之 EnumDisplayDevices() 和 ChangeDisplaySettingEX() 呼叫，或者直接与硬件通信，而实施方块 S912 和 S914。

[0125] 于方块 S918 中，于该显示被逻辑禁能后，可以使用 API 呼叫至 KMD 组件 360、362 以实际设置图形子系统 40 于其低功率模式。如此情况，处理器 12 提供适当的讯号至功率控制器 60 设置图形子系统 40 于其低功率状态。于其最简单之形式，功率控制器 60 断接电源至图形子系统 40 或图形子系统 40 之组件。或可取而代之，功率控制应用程序软件 201 可以指令功率控制器 60 设置图形子系统 40 进入低功率休眠模式，譬如由 ACPI 规格所定义之装置功率状态之其中之一。无论如何，于此种较低功率消耗模式，调节电压，和 / 或降低供电至所有的或部分的图形子系统 40 和 / 或减慢由图形子系统 40 使用之选择的时脉。

[0126] 一旦图形子系统 30、40 之特定的其中一个被逻辑上和实体上致能后，UMD 代理器组件 320、322 和 KMD 代理器组件 324 被提供为现时主动子系统之指示，以及用于图形子系统之 API/DDI 呼叫如适当的被路由至 UMD/KMD 组件 340、350、360 或 342、352、362，对应于现时致能之图形子系统 30、40，如上述说明。

[0127] 欲确保适当地设定开关 56，用于装置 10 之 BIOS 能够允许用户选择哪一个装置在起动时将是主动的。

[0128] 有利的情况是，如所述之组构开关 56 和图形子系统 40 和图形子系统 30，减少功率消耗并且引致装置 10 要求二个图形处理器之仅其中一个消耗功率，由此减少总能源消耗并且保护电池寿命。举例而言，可携式计算机典型由商务旅行者使用于电池操作模式 (DC 电源）。此种用户当旅行时之典型使用样式将包含文字处理、表示，和电子邮件应用程序软件。这些应用程序软件不需要由外部图形子系统 40 提供之重任务图形加速 (heavy duty graphics acceleration)。从使用之第二（例如，外部）图形子系统 40 转变至使用之第一（例如，整合之）图形子系统 30，具有较低之平均功率消耗，帮助高性能图形处理与较低功率消耗之间之平衡而没有牺牲总系统性能。

[0129] 图 10 为本发明之另一个范例实施例，计算装置 10’之部分之范例简化方块图。计算装置 10’实质上相似于计算装置 10。功能上相等于装置 10 之组件之装置 10’之组件系标以“’”符号，而因此将不再作详细之说明。然而，简言之，装置 10’包含二个图形子系统 30’、40’。而且，图形子系统 30’包含图形引擎 32’、内存控制器 72’、显示器接口 74’、和总线接口 78’。第二图形子系统 40’藉由高速总线 20’之方式与图形子系统 30’通信。图形子系统 40’包含其自己的图形引擎 42’、内存控制器 52’、显示器接口 54’。图形子系统 40’

进一步与图形内存 50’通信。值得注意的是，装置 10’不包含用来控制哪一个图形子系统 30’和图形子系统 40’与显示器 26’互相连接之开关。取而代之，以及将变得很清楚的，图形子系统 40’被调适绘成图形横越总线 20’至内存 14’中。

[0130] 装置 10’之软件控制操作机构相似于装置 10 之软件控制操作机构。然而，装置 10’之软件控制操作之部分当装置 10’转换于高和低功率消耗状态之间时不同于装置 10 者。

[0131] 特别地，图 11 描绘本发明之范例实施例之软件方块步骤 S1100，该实施例可以在装置 10’之系统内存内软件之控制下由处理器 12’实施。再者，每次装置 10’经历状态改变可以实施方块步骤 S1100，对于此改变将因此配置图形子系统 30 和 40。如所例示，于方块 S1102 中软件 201 决定是否装置 10’将假定其较高功率消耗模式，或其较低功率消耗模式。

[0132] 当装置 10’再开始（或者转变至）其高功率消耗模式时，则执行方块 S1104 至 S1110。于方块 1104 若子系统 40’尚未设置于其全操作（高功率消耗）模式，则将其设置在此模式。此可以藉由提供适当的讯号透过驱动程序控制图形子系统 40’至功率控制器 60’而实施。其次，于方块 S1106 和 S1108 致能图形子系统 40’。再者，此可以藉由于方块 S1104 中以逻辑方式禁能任何与图形子系统 30’相关联之互连接之显示，并且于方块 S808 中以逻辑方式致能与图形子系统 40’连接之显示而实施。可以藉由适当的操作系统 API 呼叫，譬如上述之 EnumDisplayDevices() 和 ChangeDisplaySetting() 呼叫，或者透过与硬件直接通信，再实施方块 S1106 和 S1108。

[0133] 值得注意的是，没有实际的显示器连接到图形子系统 40’。于方块 S1110，在缺乏开关 56（图 4 之装置 10）之情况下，组构图形子系统 40’之驱动程序软件控制操作绘成图像于图形子系统 30’之缓冲器 14’中，而不在关联之内存 50’内。方便地于存在高速总线 20 之情况（例如，具体实施为 PCLe 总线），由于部分转移速度由总线致能，此种绘成可能横越总线 20。

[0134] 而且，进一步组构用于图形子系统 30’之驱动程序以导致图形子系统 30’之显示器接口 74’取样于内存 14’中之帧缓冲器，以便将由图形子系统 40’绘成于内存 14’中帧缓冲器中之图像表现于互连接显示器 26’中。同时，用于图形子系统 30’之驱动程序可以指导图形子系统 30’之图形引擎 32’保持实质地静止或闲置。此种操作模式被示意地描绘于图 12A 中，仅仅将图形子系统 40’和图形子系统 30’之主动区块用网目线作成阴影。

[0135] 很显然的，于图 12A 之实施例中，未使用内存 50’和显示器接口 54’。如此情况，能够从图形子系统 40’去除这些功能区块以便降低成本。当能够制造子系统 40’以补偿由子系统 30’所提供之功能时，制造这种图形子系统也许是很有益处的。举例而言，子系统能够设有图形引擎 42’，该图形引擎 42’提供 3D 图形或视频译码能力。图形引擎 32’可以不包含这些能力。同时，由图形引擎 32’提供之 2D 图形能力不需包含于子系统 40 中。消费者，仅当需要额外的功能时，能够依次加上图形子系统 30’。

[0136] 当装置 10’欲转变至或者重回至其低功率消耗模式时，执行方块 S1112 至 S1118。广言之，图形子系统 40’被部分或完全地禁能和设置在其低功率消耗模式，并且由图形子系统 30’再实施绘成。欲达成此情况，于方块 S812 中可以致能互连接关联于图形子系统 30’之任何显示，以及于方块 S1114 可以逻辑上禁能与图形子系统 40’实际连接之任何显示。其次，再度组构图形子系统 30’之驱动程序软件控制操作以导致图形子系统 30’绘成图像于

内存 14’中。显示器接口 74’继续取样内存 14’以表现图像于与端口 78’相互连接之显示器 26’上。而且，处理器 12’首先提供适当的讯号至方块 S818 中之功率控制器 60’，设置图形子系统 40’于低功率状态。于其最简单之形式，功率控制器 60’断接电源至图形子系统 40’或者设置图形子系统 40’成较低功率修眠模式。再者，于此较低功率修眠模式，调节电压，和 / 或所有的或部分的图形子系统 40’被降低供电和 / 或减慢由图形子系统 40’所使用之选择的时脉。详言之，图形子系统之图形引擎 42’保持闲置或实质上闲置（例如，其可以减慢、禁能或降低供电）。此种操作模式被示意地描绘于图 12B 中，仅仅将图形子系统 40’和图形子系统 30’之主动功能区块用网目线作成阴影。非主动 / 闲置功能区块可以被整个禁能，或者操作于减少之电压或时脉速度。

[0137] 可视需要选择使用，当图形引擎 32’未使用时，可以禁能图形子系统 30’之部分。此能够藉由设置图形引擎 32’和其它的组件于一个或多个电压岛而促成，该电压岛可以藉由 GPIO 或任何时间图形子系统 40’负责绘成图像之相似电路之方法而被禁能。

[0138] 其它的变化亦将是明显的。举例而言，于描绘于图 12A 中之高功率模式中，图形子系统 30’和图形子系统 40’能够绘成于内存 14’或内存 50’中。于此种方式，二个图形子系统 30’和 40’可以一致操作，各绘成替代的帧于内存 14’中或者绘成各帧之部分于内存 14’中、。

[0139] 又于其它的实施例中，额外的显示器可以连接至图形子系统 30’和 40’，允许共同使用多个显示器于高功率消耗模式。于此种方式，能够使用显示器接口 54 以驱动第二显示器。依于转变至较低功率消耗模式，能够组构装置 10’以如图 9B 中描绘方式操作。

[0140] 同样情况，装置 10’（或 10）能够包含多个连接至总线 20’（或 20）之额外的图形子系统，所有的该等图形子系统在高功率消耗模式能够是主动的，并且透过图形子系统 30’之显示器接口 74’绘成图形。依于转变至较低功率消耗模式，能够禁能这些图形子系统，并且绘成之图像能够留在图形子系统 30’之图形引擎 32’中。

[0141] 又于图 13 中之另一个实施例中，计算装置 10 可底包含直接内存存取 (DMA) 控制器 90。DMA 控制器 90 可以将资料从内存 50’转移至内存 14’。于此种方式，于装置 10’之较高功率修眠模式，图形子系统 40’能够绘成图像至内存 50’。然后这些绘成之图像能够由 DMA 控制器 90 转移至内存 14’中之帧缓冲器。DMA 控制器 90 能够形成部分之图形子系统 30’或 40’（例如作为图形引擎 32’或 42’之 DMA 引擎），或者不然位在计算装置 10’中。资料可以横越总线 20’转移，或者不然从内存 50’直接至内存 14’。显示器接口 74’将继续操作如上所揭示，取样于内存 14’中之帧缓冲器表现绘成之图像于显示器 26’上。再者，图 13 之装置 10’之主动区块，于其高功率消耗模式于图 13 中以网目阴影线例示。

[0142] 当然，上述之实施例仅欲作例示用而不能用作限制。实现本发明之说明之实施例可接受许多形式、部件和细部之配置、和操作次序之修饰。本发明确切地说将包括在其范围内之所有的此种修饰，如由权利要求书所界定者。

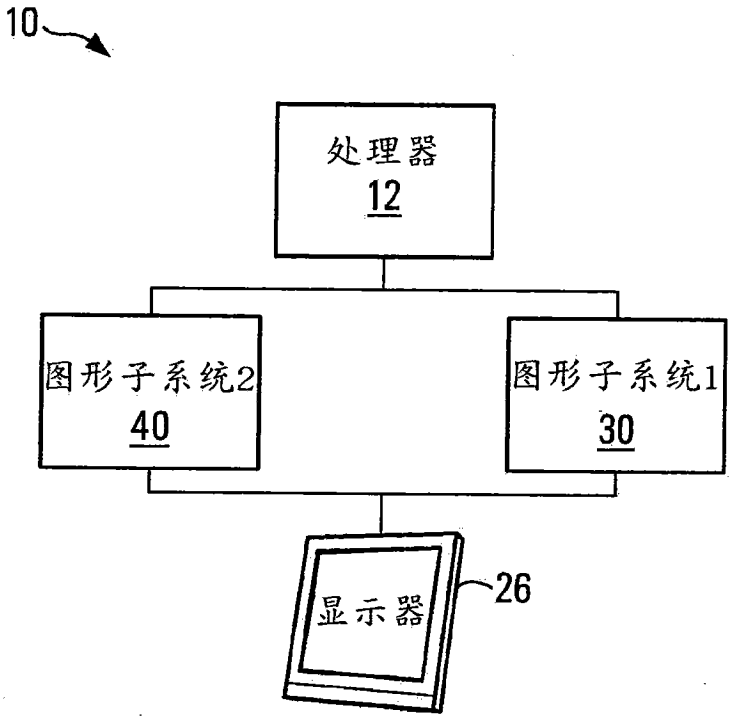


图 1

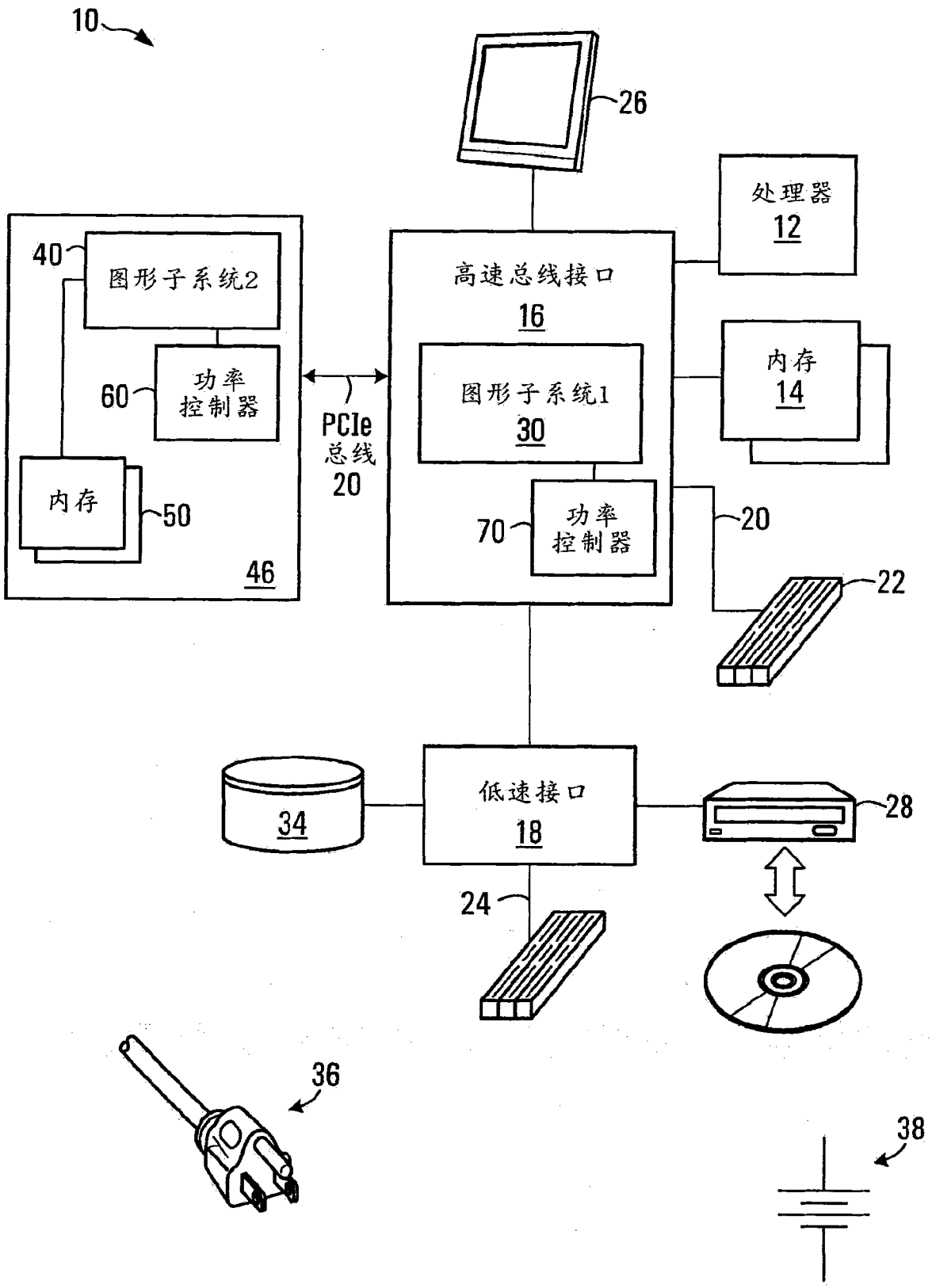

10
26
处理器
12
40
图形子系统2
高速总线接口
16
60
功率
控制器
PCIe
总线
20
图形子系统1
30
内存
14
内存
50
70
功率
控制器
20
22
46
34
低速接口
18
28
24
36
38

图 2

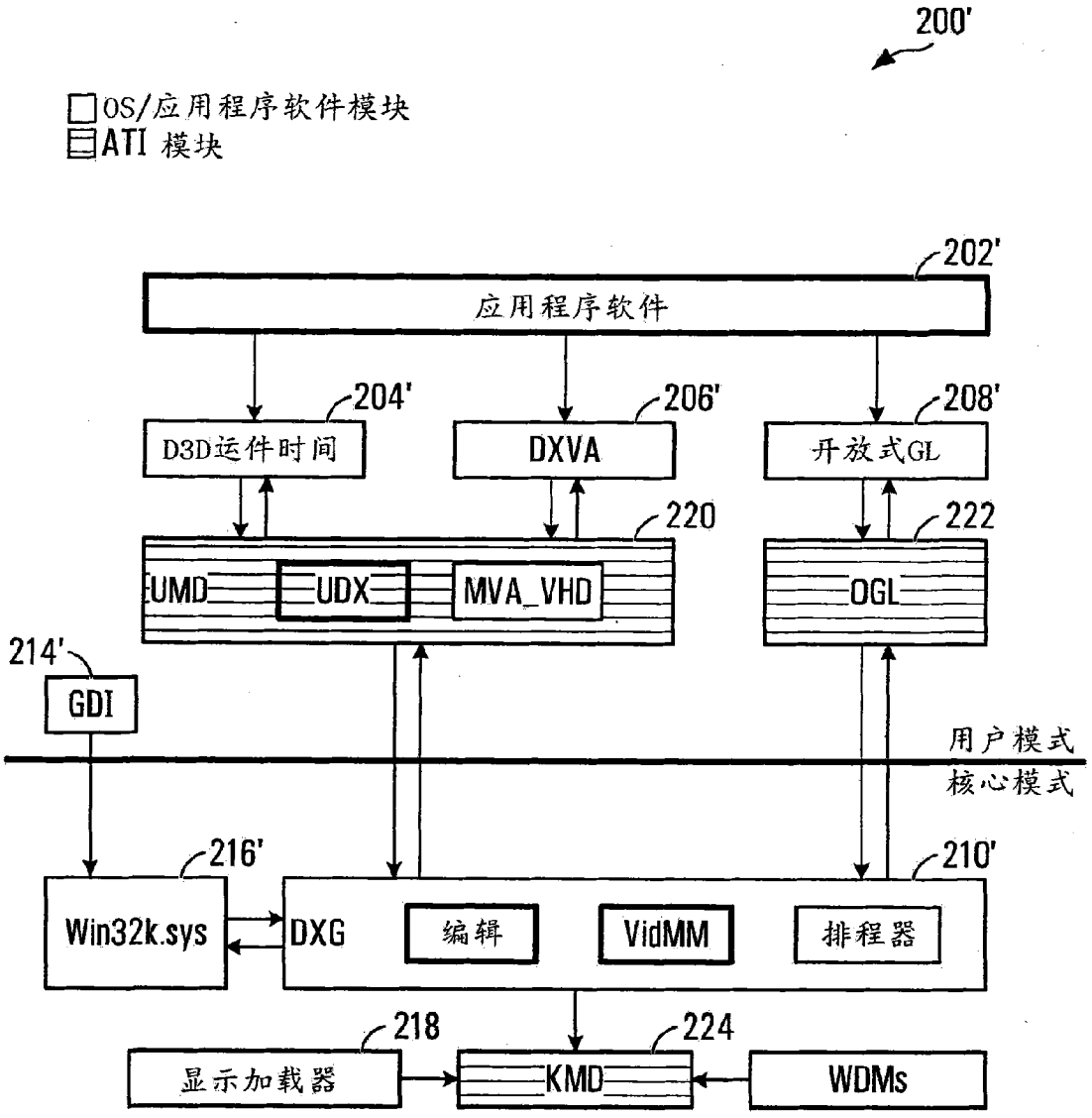

200'
OS/应用程序软件模块
ATI 模块
202'
应用程序软件
204'
206'
208'
D3D运件时间
DXVA
开放式GL
220
222
UMD
UDX
MVA_VHD
OGL
214'
GDI
用户模式
核心模式
216'
210'
Win32k.sys
DXG
编辑
VidMM
排程器
218
224
显示加载器
KMD
WDMs

图 3

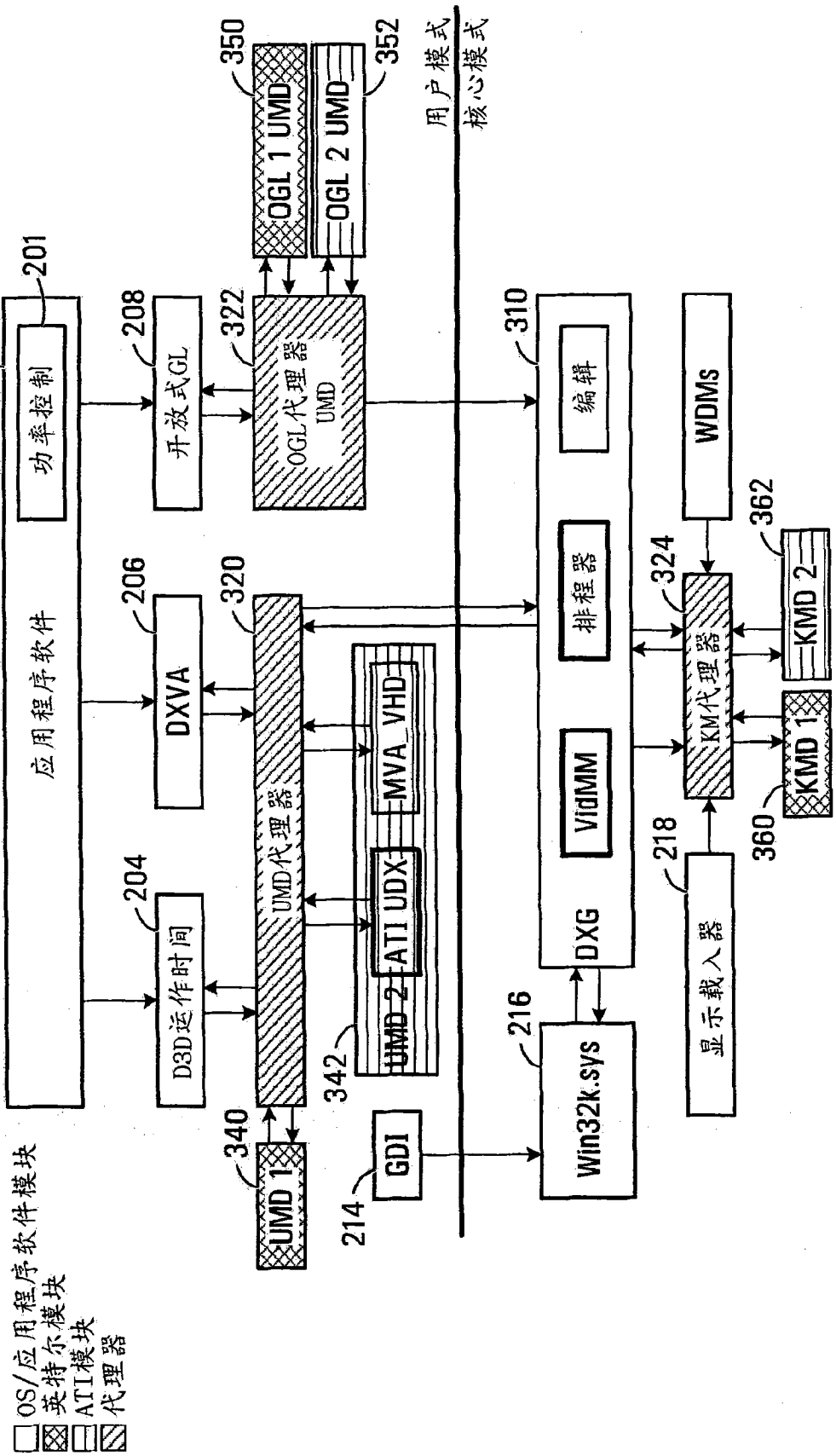

OS/应用程序软件模块
英特尔模块
ATI模块
代理器
应用程序软件
201
202
功率控制
204
D3D运作时间
206
DXVA
208
开放式GL
310
编辑
排程器
VidMM
DXG
Win32k.sys
216
218
显示载入器
320
UMD代理器
322
OGL代理器
UMD
UMD
340
UMD 1
342
UMD 2
MVA_VHD
ATI UDX
GDI
214
350
OGL 1 UMD
352
OGL 2 UMD
用户模式
核心模式
WDMs
324
KM代理器
362
KMD 2
KMD 1
360

图 4

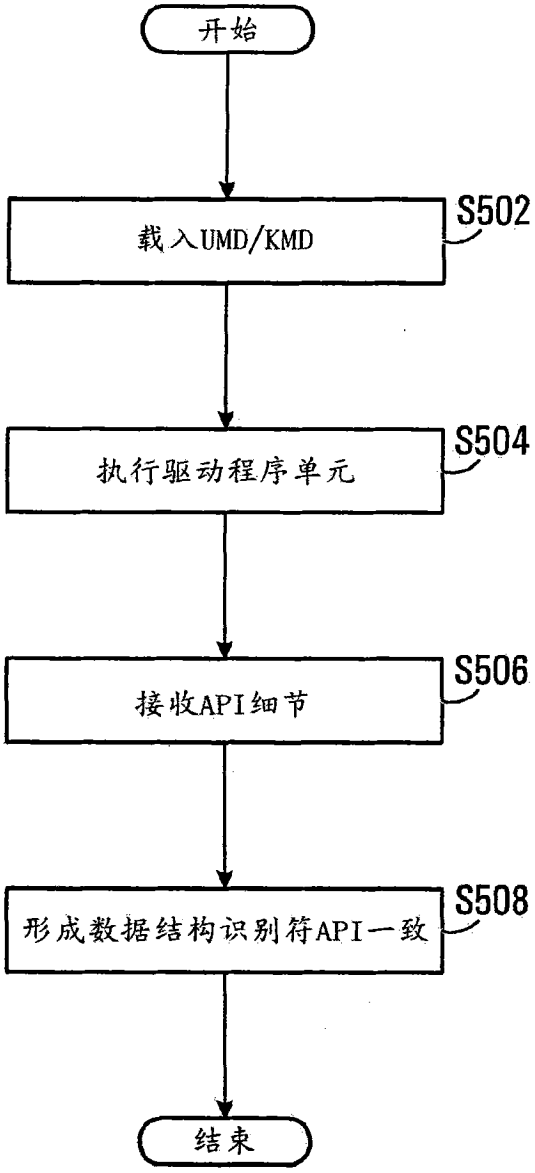

开始
载入UMD/KMD
S502
执行驱动程序单元
S504
接收API细节
S506
形成数据结构识别符API一致
S508
结束

图 5

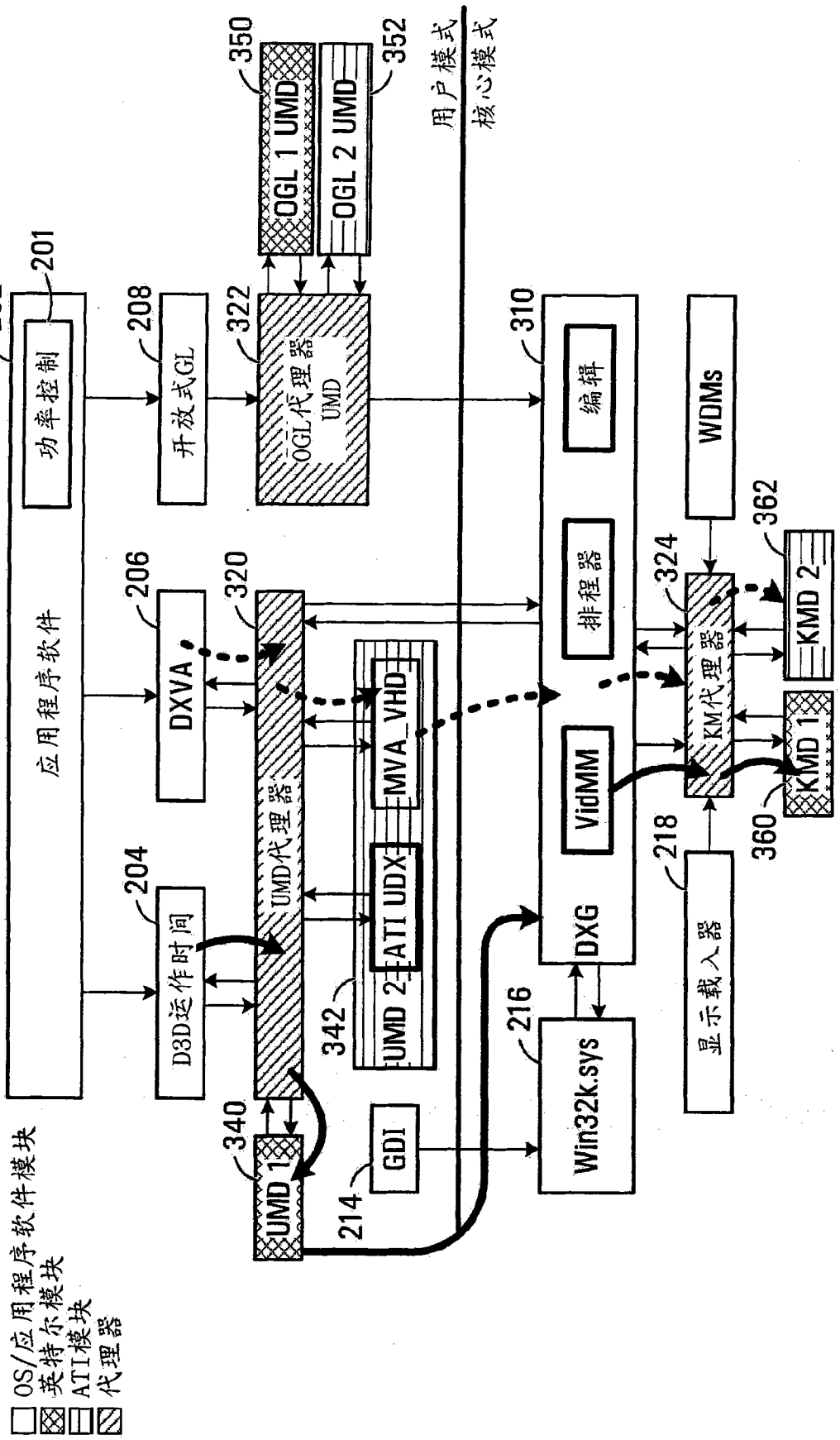

用户模式
核心模式
应用程序软件
功率控制
开放式GL
OGL代理器
UMD
OGL 1 UMD
OGL 2 UMD
DXVA
UMD代理器
D3D运作时间
MVA_VHD
ATI UDX
UMD 2
UMD 1
GDI
Win32k.sys
DXG
VidMM
排程器
编辑
WDMs
KM代理器
KMD 2
KMD 1
显示载入器
201
202
204
206
208
310
320
322
324
340
342
350
352
360
362
214
216
218
OS/应用程序软件模块
英特尔模块
ATI模块
代理器

图 6

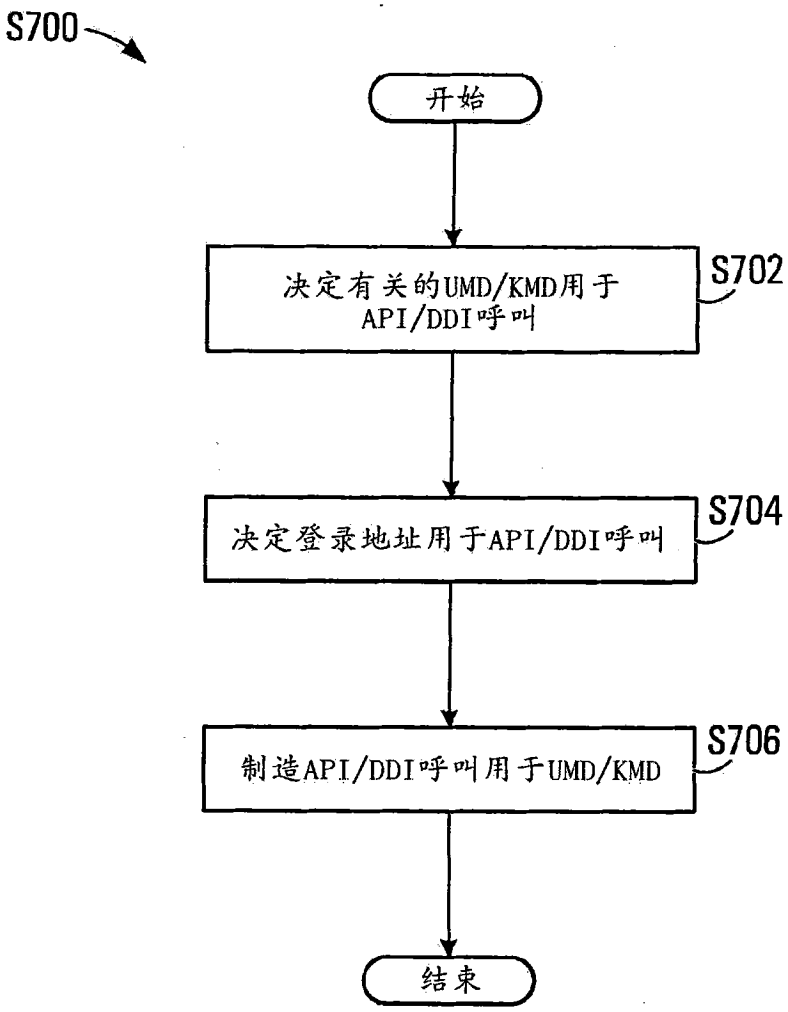

S700
开始
决定有关的UMD/KMD用于API/DDI呼叫
S702
决定登录地址用于API/DDI呼叫
S704
制造API/DDI呼叫用于UMD/KMD
S706
结束

图 7

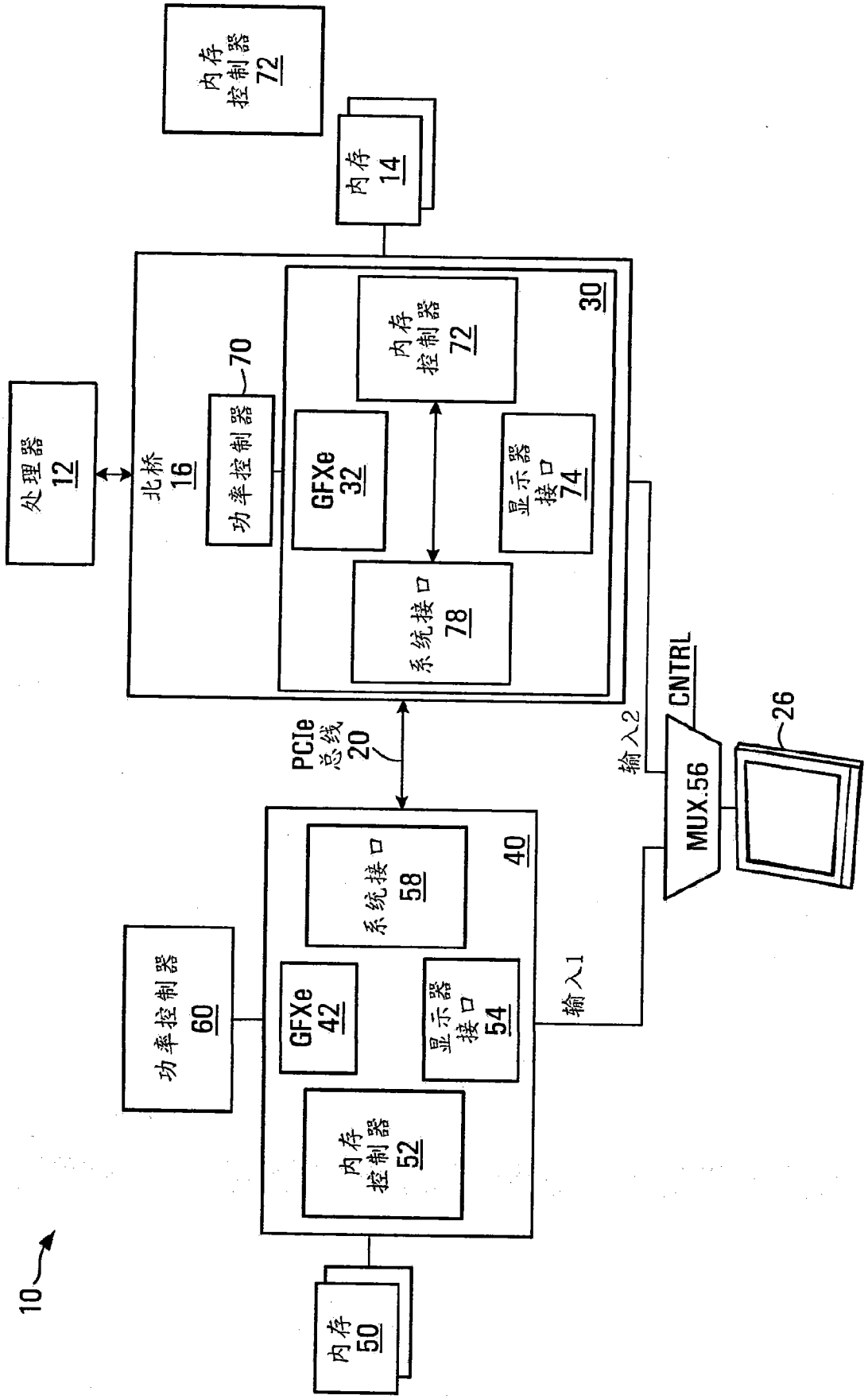

内存控制器 72
内存 14
处理器 12
北桥 16
功率控制器 70
GFXe 32
内存控制器 72
30
显示器接口 74
系统接口 78
PCIe 总线 20
输入2
CNTRL
MUX.56
26
功率控制器 60
GFXe 42
系统接口 58
40
显示器接口 54
输入1
内存控制器 52
内存 50
10

图 8

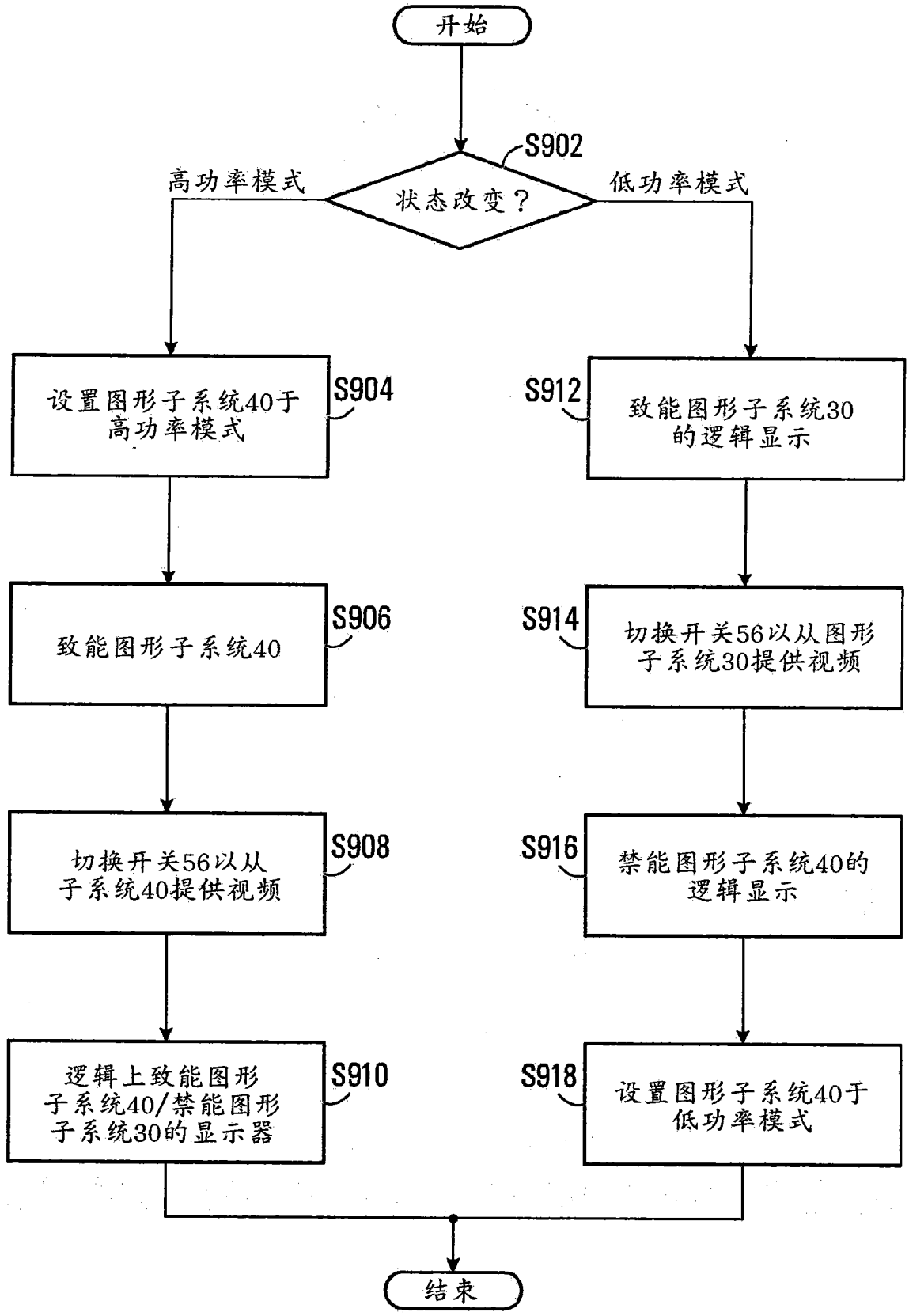

开始
S902
状态改变？
高功率模式
低功率模式
设置图形子系统40于高功率模式
S904
致能图形子系统40
S906
切换开关56以从子系统40提供视频
S908
逻辑上致能图形子系统40/禁能图形子系统30的显示器
S910
S912
致能图形子系统30的逻辑显示
S914
切换开关56以从图形子系统30提供视频
S916
禁能图形子系统40的逻辑显示
S918
设置图形子系统40于低功率模式
结束

图 9

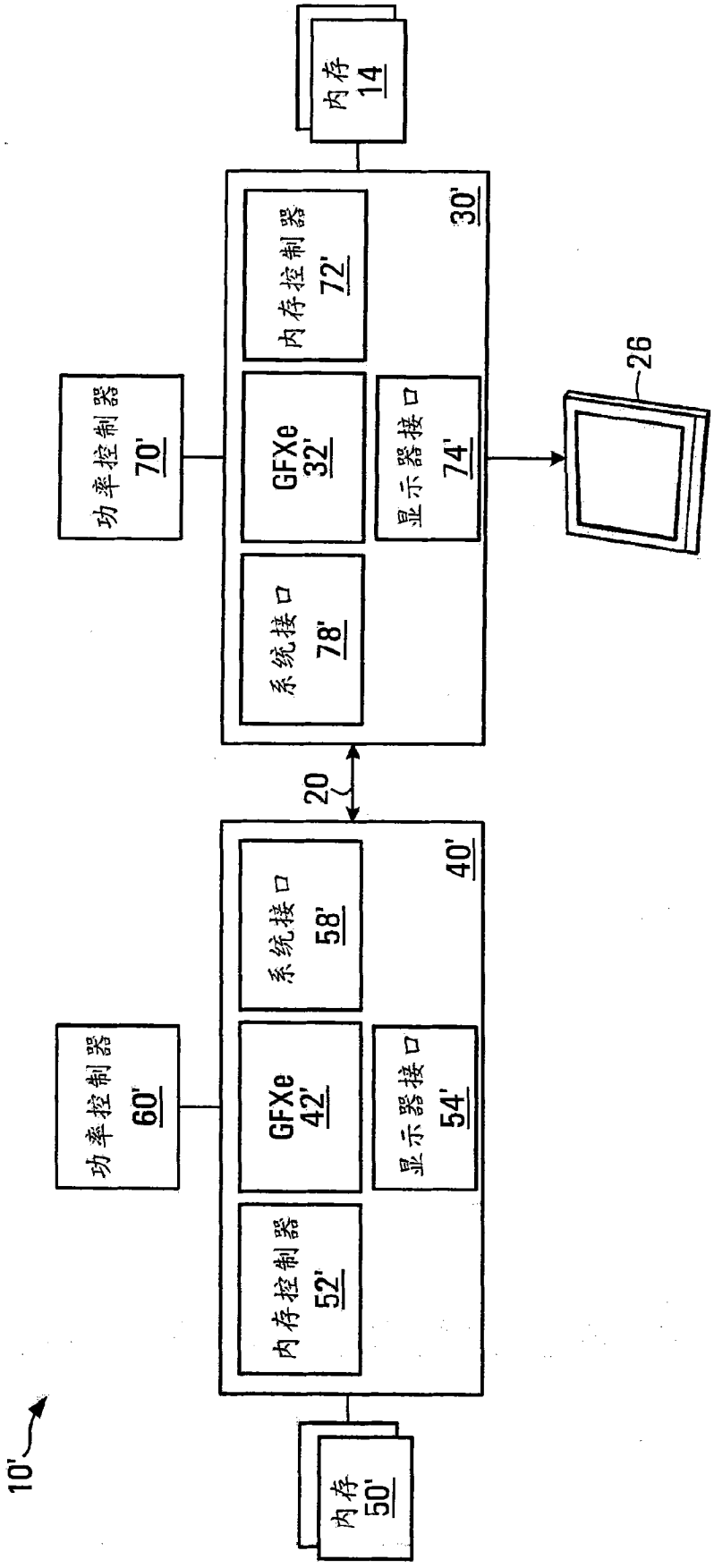

内存
14
30'
内存控制器
72'
功率控制器
70'
GFXe
32'
显示器接口
74'
26
系统接口
78'
20
40'
系统接口
58'
功率控制器
60'
GFXe
42'
显示器接口
54'
内存控制器
52'
内存
50'
10'

图 10

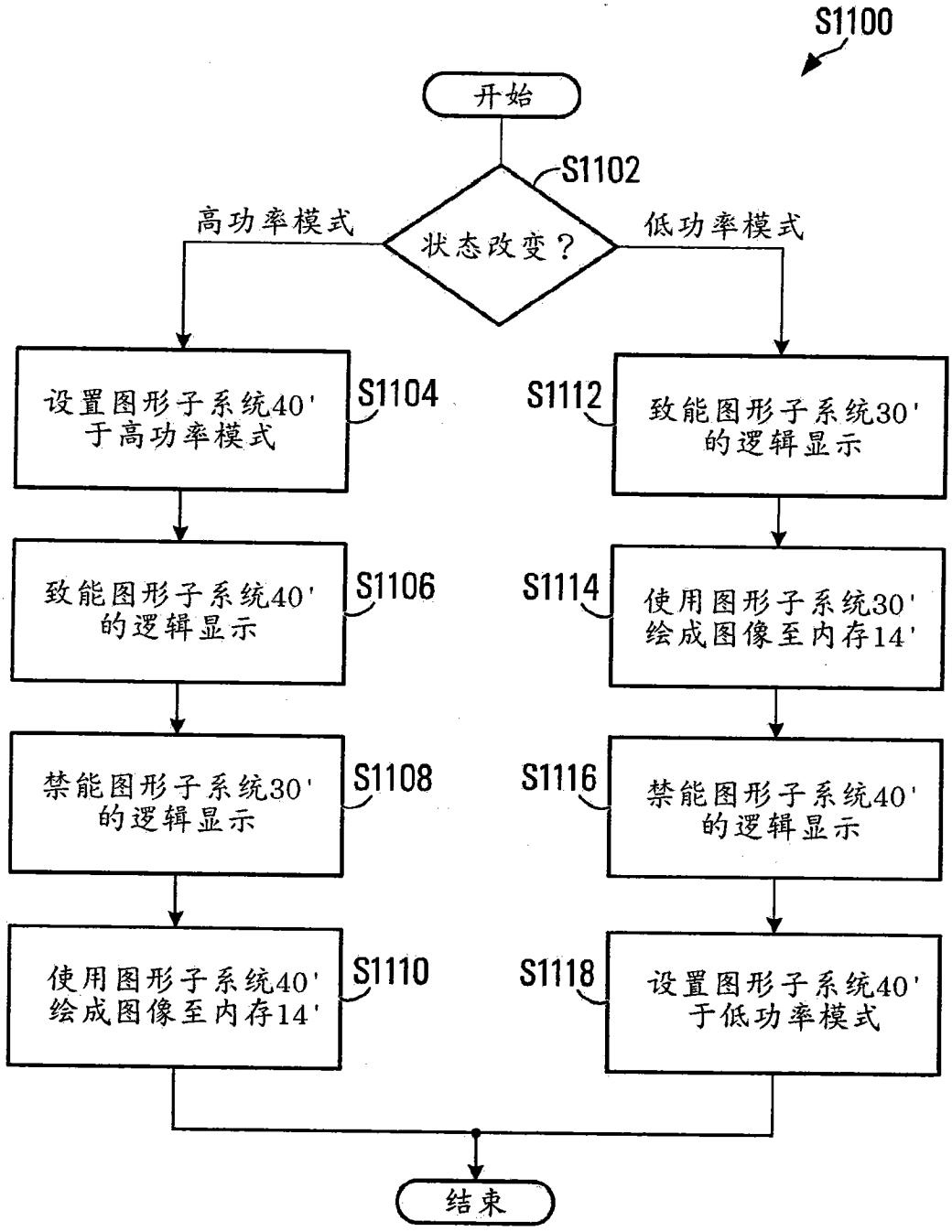

S1100
开始
S1102
状态改变？
高功率模式
低功率模式
设置图形子系统40'于高功率模式
S1104
致能图形子系统40'的逻辑显示
S1106
禁能图形子系统30'的逻辑显示
S1108
使用图形子系统40'绘成图像至内存14'
S1110
S1112
致能图形子系统30'的逻辑显示
S1114
使用图形子系统30'绘成图像至内存14'
S1116
禁能图形子系统40'的逻辑显示
S1118
设置图形子系统40'于低功率模式
结束

图 11

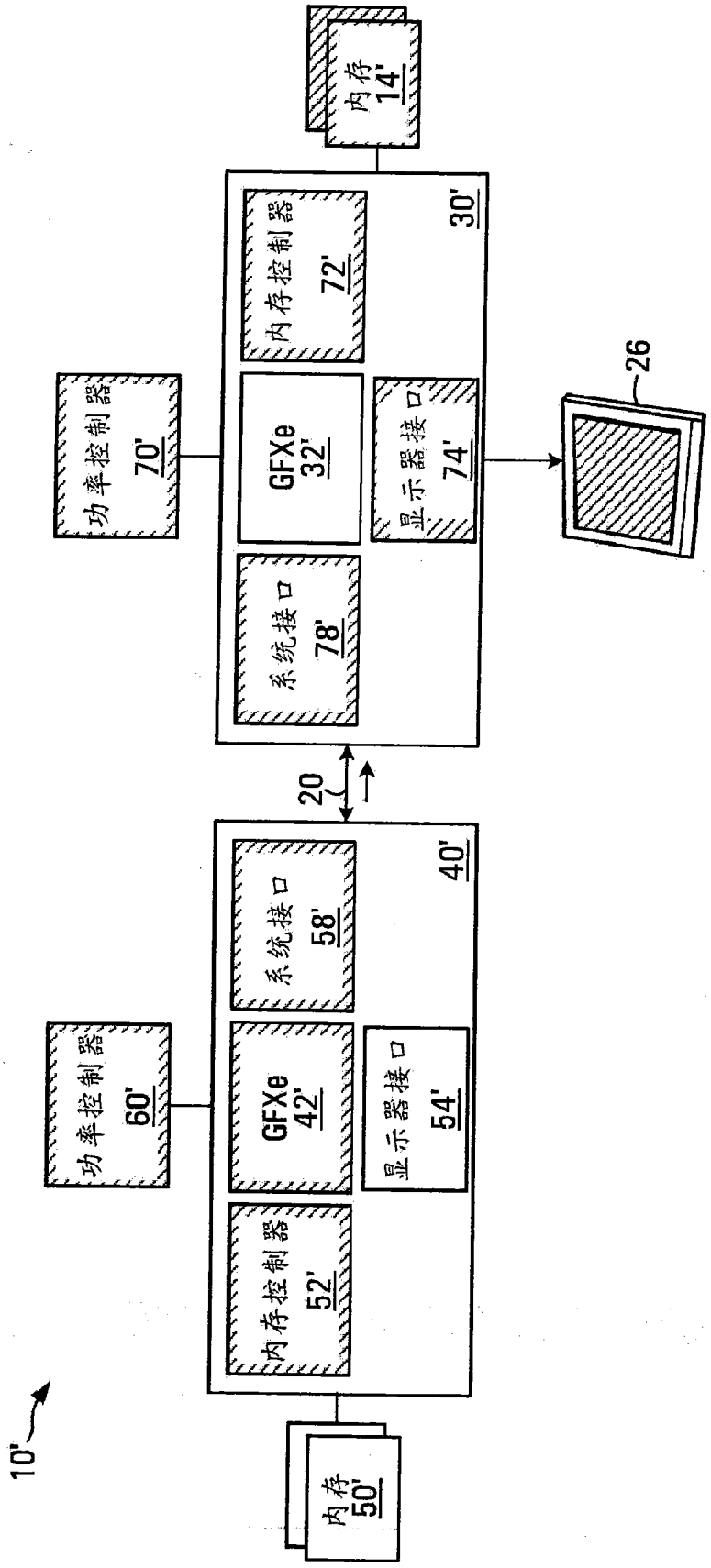

内存
14'
30'
内存控制器
72'
GFXθ
32'
显示器接口
74'
26
功率控制器
70'
系统接口
78'
20
40'
系统接口
58'
GFXθ
42'
显示器接口
54'
功率控制器
60'
内存控制器
52'
10'
内存
50'

图 12A

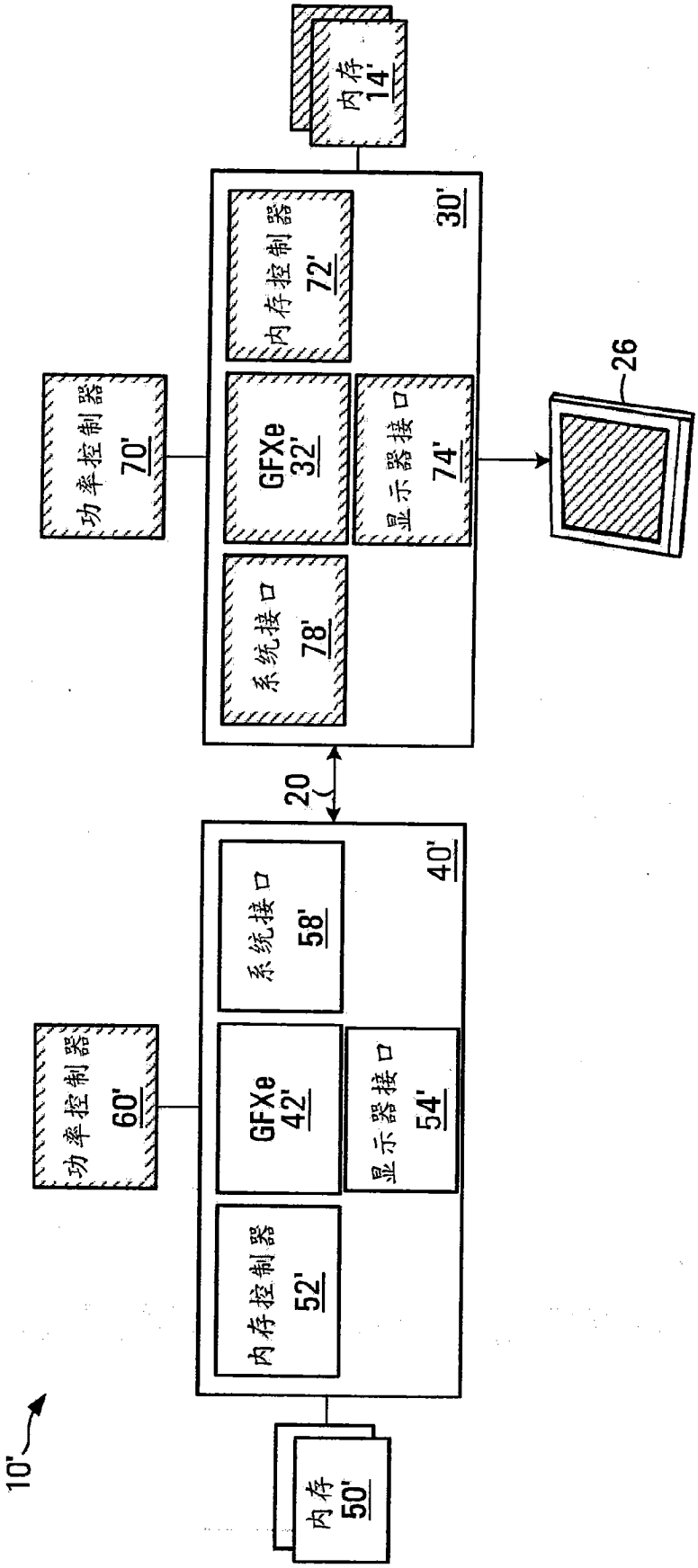

内存
14'
30'
内存控制器
72'
功率控制器
70'
GFXe
32'
显示器接口
74'
26
系统接口
78'
20
40'
系统接口
58'
功率控制器
60'
GFXe
42'
显示器接口
54'
内存控制器
52'
10'
内存
50'

图 12B

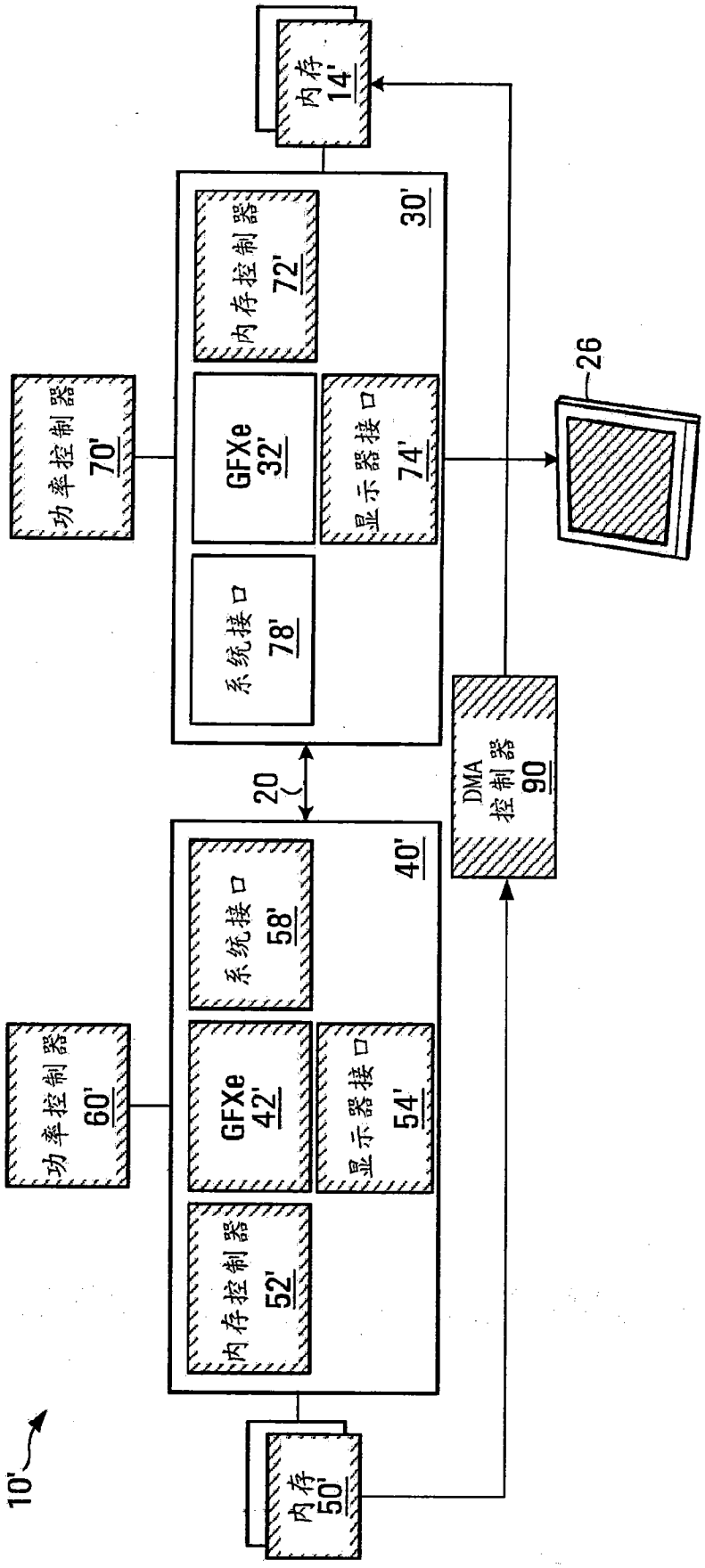

内存
14'
30'
内存控制器
72'
GFXe
32'
显示器接口
74'
系统接口
78'
功率控制器
70'
26
20
DMA
控制器
90
40'
系统接口
58'
GFXe
42'
显示器接口
54'
内存控制器
52'
功率控制器
60'
内存
50'
10'

图 13