(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 102081556 B
(45)授权公告日 2013.12.04

(21)申请号 201110026729.2
(22)申请日 2006.12.29
(30)优先权数据
11/321,779 2005.12.29 US
(62)分案原申请数据
200610064226.3 2006.12.29
(73)专利权人 英特尔公司
地址 美国加利福尼亚州
(72)发明人 H·王 J·沈 H·蒋 R·汉金斯 P·哈马隆德 D·罗杰斯 G·蔡亚 B·帕特尔 S·考施克 B·比格比 G·希菲尔 Y·塔尔加姆 Y·尤塞夫 J·P·赫尔德
(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 汤春龙 卢江
(51)Int.Cl.
G06F 9/50(2006.01)

(56)对比文件
CN 1384431 A,2002.12.11, 全文.
US 6799265 B1,2004.09.28, 全文.
US 6944746 B2,2005.09.13, 全文.

审查员 何明伦

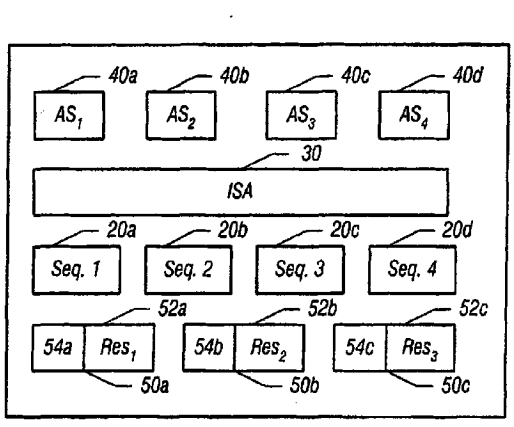
权利要求书2页 说明书12页 附图7页

(54)发明名称

与不同种类的资源通信的基于指令系统结构的内定序器

(57)摘要

在一个实施例中,本发明包括一种方法,用于在加速器和耦合加速器的指令定序器之间直接通信,其中加速器是相对于指令定序器的不同种类的资源。接口可以用于提供这些资源之间的通信。通过这种通信机制,用户级应用程序可以直接与加速器通信而不需要操作系统的支持。进一步,指令定序器和加速器可以并行执行操作。描述并要求了其他实施例。



CN 102081556 B

1. 一种基于指令系统结构的内定序器与不同种类的资源通信的方法，包括：

通过第一指令定序器将请求从用户级应用程序传送到与第一指令定序器耦合的加速器，其中加速器包括相对于第一指令定序器的不同种类的资源；

通过第一指令定序器的第一指令系统结构的指令，通过耦合到加速器的接口逻辑向加速器提供请求，其中加速器还包括第二指令系统结构的资源；以及

响应请求，并行于第一指令定序器中的第二功能执行加速器中的第一功能，其中所述第一功能和所述第二功能是所述应用程序的线程的不同部分。

2. 根据权利要求 1 所述的方法，其中通过第一指令定序器将请求从用户级应用程序传送到与第一指令定序器耦合的加速器包括根据接口逻辑和加速器之间的专用协议，发送请求到接口逻辑并且从接口逻辑传递请求到加速器。

3. 根据权利要求 1 所述的方法，进一步包括传送请求而不需要操作系统的支持，其中加速器对操作系统是透明的。

4. 根据权利要求 1 所述的方法，进一步包括响应请求将加速器的体系结构状态的子集提供到用户级应用程序。

5. 根据权利要求 1 所述的方法，进一步包括响应请求将接口逻辑和加速器的集合体系结构状态提供到用户级应用程序。

6. 根据权利要求 1 所述的方法，进一步包括根据系统的配置识别该系统的禁止资源，其中禁止资源被禁止以支持其它系统元件，并且将禁止资源配置为加速器。

7. 根据权利要求 6 所述的方法，进一步包括通过禁止资源而不是通过操作系统级介质驱动程序来执行用户级介质应用程序。

8. 根据权利要求 1 所述的方法，进一步包括通过接口逻辑虚拟化加速器，以便加速器内的功能性第一子集对用户级应用程序是可见的并且加速器内的功能性第二子集对操作系统是可见的。

9. 一种用于基于指令系统结构的内定序器与不同种类的资源通信的装置，包括：

用于执行指令的第一指令定序器；以及

耦合到第一指令定序器的资源，其包括：

加速器，用于对从第一指令定序器接收的数据执行至少一个操作，其中加速器包括相对于第一指令定序器的不同种类的资源；以及

与加速器耦合的接口逻辑，用于通过从第一指令定序器的第一指令组到第二指令定序器的第二指令组的内定序器通信的转换，使内定序器能够在用户级控制下在第一指令定序器和加速器之间通信，其中该加速器包括：

将被配置成在用户级控制下由第一指令定序器使用的第一部分；以及

将被配置成通过启用操作系统的应用程序而使用的第二部分。

10. 如权利要求 9 所述的装置，其中加速器包括固定功能单元，而接口逻辑包括与固定功能单元耦合的有限状态机。

11. 如权利要求 9 所述的装置，其中装置包括具有单个衬底的处理器，该单个衬底包括所述第一指令定序器和所述加速器。

12. 如权利要求 9 所述的装置，其中一旦完成至少一个操作，加速器将通知第一指令定序器。

13. 如权利要求 12 所述的装置，其中通知之后，在用户级控制下，第一指令定序器将执行事件处理程序，其中事件处理程序将接收和处理来自于至少一个操作的结果数据。

14. 如权利要求 9 所述的装置，其中内定序器通信包括直接通信而不需要操作系统介入。

15. 如权利要求 9 所述的装置，其中第一指令定序器包括本地指令系统结构的处理引擎，而加速器包括非本地指令系统结构的处理引擎。

16. 如权利要求 9 所述的装置，其中接口逻辑包括 ：

第一部件，用于启动对于加速器的体系结构状态信息的进入通信以及对于第一指令定序器的状态信息的发出通信 ；以及

第二部件，用于虚拟化在用户级控制下和在操作系统控制下使用的加速器。

17. 一种用于基于指令系统结构的内定序器与不同种类的资源通信的系统，包括 ：

第一处理器，其包括第一定序器，用于执行第一指令系统结构的指令 ；

与第一处理器耦合的第一系统元件，用于执行专门的功能 ；

与第一处理器耦合的第二系统元件，该第二系统元件包括第二定序器，该第二定序器包括通过内定序器通信协议与第一定序器通信的接口，通过该内定序器通信协议的通信包括向所述第二定序器发送指向在第二定序器上要执行的代码段的指令指针，第二定序器进一步包括与接口耦合的计算资源，该计算资源包括相对于将由第一系统元件执行的专门功能而具有双重功能性的第二系统元件的处理引擎，并且其中该计算资源通过该系统的配置被禁止执行专门功能并且被配置成用作加速器并且在应用程序的请求下执行处理功能 ；以及

耦合到第一定序器和第二定序器的动态随机存取存储器。

18. 如权利要求 17 所述的系统，其中第一定序器和第二定序器包括处理器的不同种类的资源。

19. 如权利要求 17 所述的系统，其中处理引擎由用户级应用程序使用以执行不同于专门功能的第一任务而不需要操作系统的支持。

20. 如权利要求 17 所述的系统，其中第一定序器将向第二定序器提供虚拟命令，其中接口将虚拟命令映射为与计算资源有关的第二指令系统结构的一个或多个指令。

与不同种类的资源通信的基于指令系统结构的内定序器

技术领域

[0001] 本发明的实施例涉及改善基于处理器的系统中的通信，尤其涉及一种包括多个定序器的系统。

背景技术

[0002] 计算机系统包括各种组件以便处理和传送数据。一般的系统包括一个或多个处理器，每个处理器可以包括多个核，以及相关的存储器，输入 / 输出 (I/O) 设备和其它这样的组件。为了提高计算效率，计算加速器，专用的 I/O 设备及其他这样的专用单元可以通过一个或多个专用的组件提供，在这里一般被称为辅助单元。然而，在使用这种辅助单元的过程中可能出现低效率，因为在执行通用处理器的一般的计算环境和行业标准操作系统 (OS) 环境中，软件栈可能阻止高效的使用。也就是说，在一般的操作系统环境中，通过不同的特权级别将系统软件与应用软件隔离，并且在这些不同的特权级别的每一个中的操作受限于 OS 环境的保存和恢复操作，以及其它限制。

[0003] 因此每当包括诸如专用加速器之类的辅助单元时，该单元通常被展示为一个设备而用户级应用程序只能通过 OS 的设备驱动程序软件栈间接地使用辅助单元，而该应用程序可以直接访问原始物理辅助单元资源。因此，通过相关的设备驱动程序的辅助单元资源是全系统范围的资源而不是诸如通过上下文转换而虚拟化的通用寄存器、虚拟存储器或定序器之类的应用程序级资源。

[0004] 不得不使用设备驱动程序来访问辅助单元的问题是效率低（根据从驱动器应用程序到辅助单元的路径长度），以及由于强加的 OS 限制相关的″标准化″驱动程序接口而不灵活。

[0005] 计算加速器的范例是诸如数学协处理器（像所谓的用于早期的英特尔®体系结构 (IA)-32 处理器的 x87 浮点协处理器）之类的协处理器。一般地，这种协处理器通过协处理器接口与主处理器（例如，中央处理单元 (CPU)）耦合，其就像主处理器一样具有普通的指令系统结构 (ISA)。此外，通过传统的退出 / 等待信号协议在这些资源之间进行交互，其中主处理器处于等待状态同时协处理器执行它的请求功能，在交互结束时控制返回主处理器。然而，在协处理器操作期间，主处理器不能执行有用功而是等待来自于协处理器的结果。也就是说，协处理器被集成以致它结构上用主处理器的控制流的程序指令顺序地操作。这导致处理器利用的低效率，特别是，当协处理器能够和主处理器上的计算同时操作时。因此存在对改进与这种辅助单元通信以及使用这种辅助单元的方式的需要。

附图说明

[0006] 图 1 是根据本发明的一个实施例的处理器的方框图。

[0007] 图 2 是根据本发明的一个实施例的系统的一部分的方框图。

[0008] 图 3 是根据本发明的另一个实施例的系统的一部分的方框图。

[0009] 图 4 是根据本发明的一个实施例的系统的方框图。

[0010] 图 5 是根据本发明的一个实施例，与处理器中执行的操作相应的方法的流程图。

[0011] 图 6 是根据本发明的实施例，与加速器中执行的操作相应的方法的流程图。

[0012] 图 7 是根据本发明的另一个实施例的系统的方框图。

具体实施方式

[0013] 在各种实施例中，提供多种结构以使基于指令系统结构 (ISA) 的内定序器能够通信。正如这里所使用的，" 定序器 " 是独特的线程执行资源并且可能是任何能够执行线程的物理或逻辑单元。定序器可能是逻辑线程单元或物理线程单元，并且可能包括下一指令指针逻辑以便确定将对给定的线程执行的下一指令。

[0014] 更具体来讲，可以在第一 ISA 的第一定序器和不同种类的第二资源之间实现基于 ISA 的内定序器通信，第二资源可以是定序器或非定序器。也就是说，第二资源可以是不同 ISA 的定序器或者可以是非定序器资源，诸如固定功能单元 (FFU)、专用集成电路 (ASIC) 或其他预编程逻辑。在各种实施例中，中间物或接口，在此称为 " 外骨骼 "，可以提供在这些不同种类的资源之间的通信。在不同的实施例中外骨骼可以采用各种形式，包括软件、硬件和 / 或固件。在一些实施例中，可以在紧紧地耦合到不同种类资源的有限状态机 (FSM) 中实现该外骨骼。当然，其它实现方式是可能的。

[0015] 现在参考图 1，示出了根据本发明的一个实施例的处理器的方框图。如图 1 中所示，处理器 10 包括各种不同的资源。在不同的实现方式中，处理器 10 可以是单核处理器或多核处理器。可以在不同类型的系统中实现这样的处理器，包括芯片多处理器 (CMP) 系统或同步的多线程 (SMT) 系统或接通事件多线程 (SoeMT) 系统，以及其它这样的系统。

[0016] 如图 1 中所示，处理器 10 包括多个定序器 20a，20b，20c，和 20d(即，定序器 1-4，和一般的定序器 20)。虽然在图 1 的实施例中用示出了四个这样的定序器，应当理解本发明的范围不因此受限。如图 1 中所示，处理器 10 中的定序器 20 实现 ISA30，在一个实施例中其可以是英特尔®体系结构 (IA-32) 指令系统结构和 / 或它的 64 位扩展名 (也叫英特尔®扩展内存 64 位技术 (EM64T))。处理器 10 进一步包括其它资源，包括第一资源 (即，资源 1)50a，第二资源 (即，资源 2)50b，和第三资源 50c(即，资源 3)(以及一般的资源 50)。这些资源可以是不同种类的资源，它们没有实现处理器 10 的 ISA30。虽然图 1 的实施例中显示为包括三个这样的资源，但是在不同的实施例中可能包括更多的或更少的这样的资源。

[0017] 每个资源 50 包括定序器 (其可以实现为不同于 ISA30 的 ISA)、非定序器处理引擎、或其它特殊功能逻辑，在这里一般称为加速器。在不同的实施例中，不同类型的资源可以实现为加速器，包括图形处理单元 (GPU)(典型的定序器)、加密单元 (典型的非定序器)、物理处理单元 (PPU)(典型的非定序器)、固定功能单元 (FFU)(典型的非定序器) 等等。如图 1 中所示，每个资源 50 可以包括加速器 52(统称)，更具体地，每个加速器 52a，52b，和 52c 与资源 50a-50c 之一有关。在这里加速器 52 也被称为辅助单元。因为资源 50a-50c 可以是另一个 ISA 或者甚至可以是非定序器并且因而可能相对于定序器 20 是不同种类的，一个接口可用于提供与这种资源通信的能力。特别地如图 1 中所示，外骨骼 54a，54b，和 54c(统称外骨骼 54) 可以与资源 50 中的每一个有关。每个资源 50 因此可以被称为表示外骨骼 54 和它的相关加速器 52 之间紧密耦合的 " 外定序器 "。用这样的方式，这些不同种类的资源可以和支持内定序器通信 (和 / 或如果可用的话基于共享存储器的寻址) 的统一

的 ISA 结构中的同类定序器资源集成在一起。此外，各种资源可以以并行方式，例如，以多指令多数据 (MIMD) 方式执行，以便可以同时使用每个资源以提高性能。

[0018] 然而在其它实施例中，资源 50 可以是相对于定序器 20 同类的定序器资源并且可以是对称的核以致它们包括与定序器 20 相同或类似的体系结构。以这种方式，可以实现并行光纤并且可以增强传统 OS 的可量测性。更进一步，在其它实现方式中资源 50 可以是不对称的核。换句话说，这些资源可以是与定序器 20 相同的 ISA，但是是不同的微体系结构。这种实施例可以帮助管理不对称并且提供与传统 OS 的兼容性。

[0019] 对于实现不同种类的资源的实施例，根据一个实施例，外骨骼可能提供这些不同种类的资源具有通用的 ISA 以便对于内定序器通信实现最小限度符合的错觉。因此在各种实施例中，不同种类的资源可以起用户级功能单元资源（而不是系统级设备）的作用。换句话说，各种用户级应用程序可以直接与加速器通信并且访问加速器以致它变成用户级功能单元。以这种方式，各种加速器资源可以变成管理的 ISA 的不同种类的组件。

[0020] 例如，每个外定序器或资源 50 可以展示为具有像″包装纸″的外骨骼 54 的定序器的专用计算加速器，以便可以由作为用户级不同种类的计算资源，诸如 MIMD 资源的应用程序直接使用这些加速器 52。

[0021] 因此，应用程序设计员可以直接使用用户级 ISA30 以编程这些辅助单元 52，即使辅助单元本身物理上不必要具有 ISA30。进一步，程序员可以使用统一的 ISA 结构来编程不同组不同种类的辅助单元，每个辅助单元具有独特的″特征″（根据 ISA 或设备属性）。实际上，根据本发明的实施例，外定序器允许程序员短路传统设备驱动程序软件栈。为此，外定序器因而可以配备具有表层外骨骼的辅助单元并且使辅助单元看起来像是最小的定序器，该定序器可以参与处理器或其它设备的基于定序器 - 感知 ISA 的内定序器操作。根据软件栈的透视图，具有外定序器的应用程序或用户运行时间可以提供应用级软件层来管理加速器为应用程序资源，并且为紧密耦合的应用程序二进制的一部分，而不需要使用基于松散耦合 OS 的设备驱动程序来管理加速器为系统级资源。

[0022] 此外，在一些实现方式中，一个或多个外定序器可以包括外在的 MIMD 多定序器 ISA 接口，其中每个参与的辅助单元在结构上用作最小的定序器资源，通过定序器 - 感知同步的操作或通过最轻度级的用户级事件产生机构的异步内定序器交互作用与主处理器（即，第一定序器）交互。即使辅助单元和第一定序器紧密耦合到相同的 OS 线程，结构上，主处理器和配备外骨骼的辅助单元交互作用为两个独特的 MIMD 定序器。特别是，通过外骨骼接口在第一定序器和辅助单元之间的数据和控制交换在结构上等于内定序器交换。

[0023] 虽然在图 1 的实施例中用具体的资源显示，但是可以理解的是，处理器 10 可以是单个物理处理器，其可以支持多个硬件线程环境（在没有透明性损失的情况下，也称作″线程环境″，注意这个与软件线程环境不相同），每个包括一组体系结构状态。在一些实施例中，某些资源对这些线程环境来说可以是可见的，而其它资源是不可见的。因而如图 1 中所示，定序器 20 中的每一个可以对应于一个线程环境。当产生对操作系统是可见的至少一些线程环境（例如，m 在 n 的范围之外，$m \leq n$）时，这些线程环境有时被称为逻辑处理器或管理的 OS 定序器。每个线程环境分别保持一组体系结构状态 AS_1-AS_n。所述体系结构状态包括，例如，数据寄存器、段寄存器、控制寄存器、调试寄存器、和大部分模式特殊寄存器。线程环境可以共享物理处理器的大部分微体系结构资源，诸如高速缓存器、执行单元、分支预

测器、控制逻辑和总线。尽管这样的特征可以被共享，处理器 10 的每个线程环境可以独立地产生下一个指令地址（并且执行，例如，从指令高速缓冲存储器、执行指令高速缓存器、或追踪高速缓存器中的读取）。与线程环境相应的定序器 20 中的每一个与对应的体系结构状态 40（统称）有关。更具体地说，例如体系结构状态 (AS1)40a 可以与定序器 20a 有关，$AS_2$40b 可以与定序器 20b 有关，$AS_3$40c 可以与定序器 20c 有关，而 $As_4$40d 可以与定序器 20d 有关。

[0024] 使用处理器 10 或类似的这种处理器，基于 ISA 的内定序器通信可以在不涉及 OS 的情况下发生。例如，在共享存储器多处理范例中，应用程序设计员可以把软件程序（即，应用程序或处理）拆分成为多个将被同时运行的任务以便表示并行。相同软件程序的所有线程（" 处理 "）共享存储器地址空间的公共逻辑视图。然而，OS 线程可以与也许不被创建、调度、或相反由操作系统管理的多用户级线程有关。这种用户级线程可以被称为 " 碎片 "，以便把它们与 OS 线程区分。这些碎片也许对于 OS 调度程序是不可见的，因此 OS 不管理相关的 OS 线程何时或如何调度碎片以在分配的逻辑定序器地址上运行。OS 线程本身通常负责调度何时以及怎样运行它的一个碎片。

[0025] 对基于 ISA 的内定序器通信的体系结构支持可以包括对于 ISA 的扩建，以致提供一个或多个指令来允许用户直接操作定序器之间的控制和状态转换，其可以是所谓的定序器 - 感知或定序器 - 算术指令。这种指令可以包括如下指令，即该指令或者为第一定序器作准备以发信号给另一个（即，第二）定序器（一个指令在这里被称为碎片转换或 " SXFR" 指令，其可以发送称作外出方案的外出控制信息，并且也可以携带数据有效负载）或者为安装第二定序器作准备以监控这种信号（在这里称为碎片监控或者 " SEMONITOR" 指令）并且一旦异步地接收所述信号（称作进入方案）就对控制器执行控制转换。

[0026] 定序器 - 感知指令也可以包括诸如定序器 - 感知状态保存和恢复指令之类的其它指令。一旦执行这种状态保存指令，第一定序器可以创建第二定序器的体系结构状态的快照拷贝。定序器 - 感知恢复指令可以指定保存的体系结构状态被加载到指定的定序器。

[0027] 对于至少一个实施例，一个或多个定序器 - 感知指令可以由程序员编码成为属于 OS 线程的碎片。当在操作 OS 线程期间执行时，这种指令可以促使生成、控制转换、环境保存、环境恢复或对碎片的其它操作，而不需要插入 OS 调度逻辑。

[0028] 以这种方式，根据一个实施例，基于 ISA 的内定序器通信减少了系统开销，提高了性能。根据本发明的一个实施例，除了在相同 ISA 的定序器之间通信之外，在各种实施例中可以例如，通过外骨骼在不同种类的定序器之间或在定序器和非定序器之间发生基于 ISA 的内定序器通信。

[0029] 现在参考图 2，示出了根据本发明的一个实施例一个系统的一部分的方框图。如图 2 中所示，系统 100 包括定序器 110 和加速器 130。加速器 130 可以在不同的实施例中采用许多不同的形式，但是为了在这里讨论的目的，假定加速器 130 具有与定序器 110 不同种类的特性。换句话说，加速器 130 可以具有不同的 ISA 或可以是非定序器。然而在各种实施例中，定序器 110 和加速器 130 可以在单个衬底上实现（例如，作为多核处理器的一部分）。另一方面，定序器 110 和加速器 130 可以在集成电路 (IC) 内的不同片的硅中，或在诸如位于封装或母板上之类的不同的 IC 中，或以另外的方式实现。

[0030] 为了使基于 ISA 的内定序器能够通信，外骨骼 120 可以与加速器 130 耦合。外骨骼

120 和加速器 130 一起在这里也被称为外定序器 135。在其中加速器 130 具有不同种类的 ISA 或是非定序器的实施例中，可以是一个有限状态机 (FSM) 或虚拟化层的外骨骼 120 可以被实现（在硬件、固件乃至在软件中，取决于具体实施例）以便加速器 130 可以参与内定序器通信。这种基于 ISA 的内定序器通信在进入方向上提供信号协议到加速器 130 中，以便它可以监控和响应由 SXFR 从另外的定序器或外定序器发送的进入方案，包括用于外定序器的体系结构状态的 GET 和 / 或 SET 指令。此外，信号协议包括从加速器 130 到信号定序器 110 与包括用于诸如代理执行对这种事件的请求的异常处理的指示作为页面错误的外出方案进行外出通信。此外，在一些实施例中，通过外骨骼 120 的加速器 130 可以参与面向通信活动的性能报告以致一旦有来自于定序器 110 或另外的资源的查询，外骨骼 120 可以传送关于加速器 130 性能的信息以允许它们的更有效的使用。

[0031] 如图 2 中所示，外骨骼 120 可以与加速器 130 紧密耦合。进一步如图 2 中所示，系统 100 包括用户级运行时间软件层，表示为碎片库 150 以管理和调度碎片，即，用户级线程。在不同的实现方式中，碎片库 150 可以执行操作以支持和管理在诸如定序器 110 之类的定序器上的碎片。例如，碎片库 150 可以管理碎片调度、碎片环境保存、切换、和恢复，等等。以这种方式，消耗了低系统开销因为 OS 190 不涉及这些操作。

[0032] 可以在系统 100 内执行用户应用程序 160 并且可以请求执行专门的或计算密集的功能。为了能够提高性能，尤其在 MIMD 或并行的环境中，用户应用程序 160 可以请求定序器 110 与加速器 130 通信以致加速器 130 将并行于执行应用程序其它有用功（或其它将由定序器 110 执行的碎片）的定序器 110 而执行功能。用这样的方式，改进执行得以实现，因为定序器 110 和加速器 130 两者可以在作为碎片的应用程序线程的不同部分上并行执行。因此，使用本发明的实施例，定序器 110 的控制流可以并行地并且与外定序器上执行的专用计算异步地运行，其作为单独的碎片有效地操作。

[0033] 为了减少系统开销，在定序器 110 和外定序器 135 之间通过定序器 - 感知指令的基于 ISA 的内定序器通信也许不需要 OS 190 的介入。以这种方式，可以避免 OS 190 的设备驱动程序栈并且相反在定序器 110 和外定序器 135 之间的直接通信可以实现。因而如图 2 中所示，在一个实施例中，可以在定序器 110 和外骨骼 120 之间通过定序器 - 感知指令直接进行基于 ISA 的内定序器通信。可以是用户应用程序的应用程序代码 160 可以通过碎片运行时间库 150 使用这种基于 ISA 的内定序器通信。最小的 OS 层 140 支持 OS 线程的环境保存和环境恢复操作。当 OS 线程在 OS 管理的定序器上执行环境保存或环境恢复指令时，用于所有应用程序管理的定序器以及与 OS 管理的定序器有关的外定序器的环境将因此被保存和恢复。OS 可以对每一 OS 线程提供足够的保存区，用于保存这些与 OS 线程环境转换相交叉的状态。那里，基于 ISA 的内定序器通信可以用这样的方式转换，即它可以被提供给外定序器 135 并且遵照外定序器 135 行事。在相反方向可能发生类似的通信流。在至少一个实施例中，在每个定序器或外定序器上一旦 OS 线程环境转换，可以由并行于相同 OS 线程中的其它定序器 / 外定序器的每个定序器或外定序器执行环境的保存和恢复。这种并行实现可以提高整个保存和恢复操作的性能。

[0034] 虽然图 2 的实施例中所示的为包括单个定序器 110，但是可以理解的是，本发明的范围不因此受限，并且在其它实施例中，可以提供多定序器系统。在这种系统中，外骨骼 120 可以进一步执行虚拟化功能以便单个物理的加速器 130 可以由多个定序器关联。因此，外

骨骼 120 可以执行对多个环境的存储，以便为不同的定序器存储加速器 130 的体系结构状态的多个副本，每个与逻辑外定序器相耦合。外骨骼 120 可以进一步包括逻辑，以便能够保存和恢复环境状态，以及为定序或多路复用不同的定序器作准备以使用加速器 130。以这种方式，加速器 130 可以服务一个系统的多个定序器。在至少一个实施例中，外骨骼可以通过接通事件多线程而执行多个逻辑外定序器。在这种实施例中，逻辑外定序器环境能够被执行为独特的芯片上寄存器堆或专用的暂时存储器，并且外定序器环境转换状态可以在逻辑中或在诸如微代码之类的固件中执行。在至少另一个的实施例中，可以为每个定序器实现独特的物理外定序器。在这个实施例中，多个物理的外定序器可以被实现为同步的多线程 (SMT) 或 CMP 系统。

[0035] 当加速器不具有和定序器 130 和 / 或剩余的系统相同的 ISA 或其中加速器是非定序器时，可以使用诸如图 2 中所示的实施例。在这种实施例中，图 2 中所示的数据流提供定序器 110 和加速器 130 之间的高效的基于 ISA 的内定序器通信而不需要 OS 190 的介入。

[0036] 然而，在其它实现方式中，加速器可以具有公共的 ISA 作为定序器或系统的其它部分。更进一步，在一些实现方式中，可以用剩余的系统接近地配置加速器。例如，在一些实现方式中，加速器可以是配置为执行专门操作的系统组件。然而，在给定的系统配置中，组件是无效的或有利于诸如外围设备或附加设备之类的其它组件的最低限度地的使用。例如，可以将集成的图形处理单元 (GPU) 实现为系统主板上的芯片组的一部分（诸如图形存储器控制器集线器 (GMCH)）。然而，在某些系统配置中，附加的离散图形卡也被配置为插入（例如，在外围设备组件互连 (PCI) 槽口上）到主板上。在这种实例中，集成的 GPU 可以被无效等而不使用。类似地，GPU 或其它这种组件可以包括多个不同的处理引擎，其中一些在某些系统配置中也许不被完全使用。

[0037] 在这种实现方式中，根据本发明的一个实施例，这些或使无效或最低限度使用的处理资源可以被配置为加速器。现在参照图 3，示出了系统 200 的一部分的方框图，其包括定序器 210 和加速器 230。在图 3 的实施例中，加速器 230 可以包括第一部分 232 和第二部分 234。第一部分 232 可以被配置为充当外定序器，而第二部分 234 可以被配置为执行诸如在正常的 OS 控制下专门的图形或介质功能之类的各种功能。因此，如图 3 中所示，第二部分 234 与 OS 290 的设备驱动程序栈 295 耦合，以致它可以通过 OS 应用编程接口 (API) 到设备驱动程序栈 295 与基于 OS 的执行模型的应用程序 260 通信。用这样的方式，第二部分 234 可以在应用程序 260 的请求下通过传统的 OS 通信路线执行处理功能。

[0038] 与此相反，第一部分 232 被配置为通过外骨骼 220 直接与定序器 210 通信。外骨骼 220 可以是硬件、软件、固件或它们的组合，以使能够在第一部分 232 和定序器 210 之间基于 ISA 通信。因此，用户级应用程序 256 可以使用基于 ISA 的内定序器指令用于通过外骨骼 220 在定序器 210 和第一部分 232 之间通信。一般地，可以使用用户级碎片库 250，并且 OS 支持 240 的最小层用于提供对如上所述的 OS 线程环境保存和恢复操作的支持。

[0039] 因而，在图 3 的实施例中，根据本发明的一个实施例，两个软件栈可以共存，即 OS 驱动程序栈和用户级运行时间栈。通过为资源共享加速器 230 作准备，可以实现改善性能，因为通过基于 ISA 的内定序器通信，基于传统的应用程序（例如，使用 OS 设备驱动程序模式）和为最小的系统开销作准备的用户级应用程序两者可以利用加速器 230 的资源。在一些实现方式中，外骨骼 220 可以相对于加速器 230 执行虚拟化任务，以致应用程序 256 和应

用程序 260 两者都相信它们已经拥有加速器 230 的完整资源（由于对于应用程序是可见的）。因而在不同的实施例中，外骨骼 220 可以执行虚拟化任务，该任务包括提供用于加速器 230 的体系结构状态的多个环境以及在类似接通事件多线程情况下提供对环境转换的支持。

[0040] 因而，根据本发明的一个实施例，加速器 230 中功能的子集可以通过 OS 290 与传统应用程序有关，而不同的功能子集可以通过基于 ISA 的内定序器通信协议与用户级应用程序有关。因而，加速器 230 的物理资源可以支持这两个全异范例的共存。

[0041] 现在参考图 4，示出了根据本发明的一个实施例一个系统的方框图。如图 4 中所示，系统 300 包括与图形存储器控制器集线器 (GMCH)320 耦合的处理器（例如，定序器）310，该图形存储器控制器集线器 320 依次与存储器 330 耦合，该存储器 330 例如可以是动态随机存取存储器 (DRAM)。此外，GMCH320 与显示器 340（诸如平板显示器）耦合。GMCH320 可以包括集成的图形加速器。GMCH320 进一步与输入 / 输出 (I/O) 控制器集线器 (ICH)350 耦合，该集线器可被用于将各种外围设备耦合到系统 300。例如在图 4 的实施例中示出的是外部图形设备 360，其可以是分离的图形设备，该设备连同另一个外围设备 370 一起耦合到 ICH350。

[0042] 因为系统 300 被配置为具有单独的外部分离图形设备 360，GMCH320 内集成的图形可能是无效的。例如，在 GMCH320 中系统基本输入 / 输出系统 (BIOS) 可以编程无效位，或另一个机构可以使图形功能无效。另外，根据本发明的一个实施例，在 GMCH320 中被用于图形处理的空闲进程资源可以替换地被转换为加速器。因此，例如，通过用户级应用程序，资源可被用于执行各种功能而不需要 OS 介入。以这种方式，可以实现改善性能，尤其其中通过 GMCH320 的图形资源的这种处理与处理器 310 的定序器中的任务并行（例如，以 MIMD 方式）执行。

[0043] 在一些实施例中，GMCH320 的集成图形中用于图形功能的外定序器可以包括执行互不相关的功能的各种图形处理单元。一个或多个这些处理资源可以被配置为基于 ISA 的介质加速器外定序器，以便实现系统 300 内的介质操作，而没有与系统 300 的 OS 有关的一个或多个介质设备驱动程序的介入。以这种方式，可以在具有处理器 310 的最少介入的系统 300 中执行诸如编码和解码数字介质之类的介质操作，而且不用遭受 OS 的设备驱动程序栈的系统开销。

[0044] 在各种实施例中，像定序器、外定序器是一种应用程序级体系结构资源，并且因此可能具有供定序器 - 感知用户级操作使用的唯一的和虚拟化的名字空间。因此，可以完全在用户级执行内定序器通信而不需要 OS 介入。随着定序器的虚拟化，逻辑的外定序器体系结构资源可以以类似于记录重命名和存储器虚拟化的方式被管理，其中物理资源的数目不需要与逻辑资源的数目相同。进一步地，根据包括体系结构、微体系结构、热量、功率损耗特性等等优点的量度的光谱，在物理资源之间可能存在各种差别。这些差别实际上在外定序器资源之中本身显现为不对称性和不均匀性。此外，作为实际上可寻址的资源，外定序器资源还能够受到基于性能的管理，其中可以由某一个硬件、固件、和抽象的软件层使用虚拟定序器地址中的位，以便表示物理的特定定序器的性能来管理同步的或异步的对外定序器资源的访问控制。

[0045] 在至少一个实施例中，外定序器可以通过提供对下列来自于第一定序器的两个规

范的进入方案的响应，而支持最少的内定序器交换协议 :GET 进入方案，其中第一定序器可以使用 SXFR 以便向外定序器发送 GET 方案从而读取 / 获得本地状态给外定序器 ;以及 SET 进入方案，其中第一定序器可以使用 SXFR 以便向外定序器发送 SET 方案从而写入 / 更新本地状态到外定序器。更进一步，外定序器可以使用 NOTIFY 外出方案和 SXFR，以便向第一定序器发送 NOTIFY 方案来执行例如结束、进程或异常条件的异步事件通知。PROXY 方案是 NOTIFY 外出方案的特殊形式，该 PROXY 方案可以代表第一定序器促使加速器以代理执行模式操作。

[0046] 有了这个最低限状态报告和更新性能，第一定序器可以构成高级的控制消息来操作数据和控制状态（单独地或整个）。尤其是，可以构建各种合成状态访问。例如使用环境保存 / 恢复，或者第一定序器可以重复地使用 SXFR 通过环境转换来读取或更新一组将被保存 / 恢复的状态，或者在可替换的实施例中，第一定序器可以使用一个 SXFR 来读取和更新外定序器本地重复过多状态的状态子集。此外，第一定序器可以使用 SXFR 来查询配置（即，性能列举），该配置包括状态、相关属性（写入，读取等等）、格式等等。以这种方式，传统上或者不是定序器或者不基于 OS 的设备驱动程序就无法与通用处理器工作的专用类的集成设备，可以根据需要由特殊的应用程序虚拟化，就像它们是处理器的功能部件并且它们的状态通过 OS 线程环境转换而经过 OS 线程环境保存和恢复。

[0047] 在一个实施例中，定序器类型的并且具有不同于第一定序器的 ISA 的辅助单元可以使用它专用的 ISA 以实现外骨骼有限状态机 (FSM) 来经由体系结构定义的用于第一定序器的输入和输出方案通信。用这样的方式，辅助单元可以选择使用它专用的 ISA 以响应由第一定序器请求的操作而实现各种算法，即使辅助单元硬件本身不直接支持所述操作。可以在任何物理的互连或接口上实现这个定序器 FSM 实现方式。互连的两个设备可以彼此发信号，并且该信号可以在第一定序器上被本地接收和转换为逻辑上和结构上的用户级事件并且作为触发脉冲输入给 FSM。在一个实施例中，受有关查询的环境保存 / 恢复进入 / 外出方案影响的外定序器的集合状态可以是辅助单元的初始状态的子集（或全集）。在该情况下，辅助单元的专用 ISA 中的外骨骼代码具有固件中实现的微代码例程的性质以支持第一定序器的定序器 - 感知操作。

[0048] 在另一个实施例中，与第一定序器的 ISA 相同的分离的定序器可以被用作外骨骼以代表辅助单元实现最小的 FSM。而外定序器和辅助单元之间的专用协议不是必须的本地 ISA 兼容，第一定序器可以与外骨骼交互，以便外骨骼为第一定序器呈现自身的集合状态和辅助单元（即，不同的 ISA）。可以由外骨骼定序器将辅助单元的状态到外定序器的集合状态映象为 1:1 或 M:1（其中 M ＞ 1）。此外，实现 FSM 以支持第一定序器的内定序器操作的代码序列类似于辅助单元硬件的基于微代码的控制。

[0049] 在另一个例子中，外定序器可以包括硬连接的 " 固定功能 " 辅助设备，该设备没有（任何 ISA 的）内部控制序列，例如，具有 I/O 逻辑的专用 ASIC 块。对于这种辅助单元，在至少一个实施例中，可以经由附着于固定功能单元的代理 FSM 而构造外骨骼。可以在具有物理访问 ASIC 的外骨骼 FSM 中实现最少的定序器状态列举和查询界面。所展示的 ASIC 状态到外骨骼定序器状态可以映象为 1:1 或 M:1，并且可以在外骨骼 FSM 中实现。FSM 逻辑可以物理地实现为到 ASIC 块的接口。在其他实施例中，外骨骼 FSM 可以是可编程的微控制器定序器，或硬布线的 ASIC 块，但是能够截取进入方案并且发布外出方案。

[0050] 在一个实施例中，与第一定序器的 ISA 相同的分离的定序器可以工作以实现外骨骼 FSM，而外骨骼定序器可以物理访问 ASIC 状态，并且逻辑上 / 结构上集合 ASIC 状态加上它自己的状态作为外定序器状态。在另一个实施例中，不同于第一定序器的 ISA 的分离的定序器可以工作以实现外骨骼 FSM。

[0051] 但不论是哪种情况，可以或者物理地通过保留静态贡献给给定辅助单元的物理定序器，或者动态地多路复用为多个逻辑分离的定序器来执行外骨骼定序器的分离，每个逻辑分离的定序器与唯一的辅助单元逻辑地关联。换言之，外骨骼定序器可以是一个通过对定序器地址虚拟化的任何想要的方案来虚拟化的定序器。

[0052] 在一个实施例中，加速器可以具有相对于第一定序器的非相干高速暂存存储器。为了使该辅助单元成为外定序器，非相干高速暂存存储器状态可以被映象为外定序器的″公布的″（即，可列举的）集合状态的一部分。如果暂时存储器的尺寸足够大，外骨骼仅仅可以得到暂存状态（包括状态的零数量）的有限子集作为对应的外定序器状态的一部分。例如，即使加速器可以处理 1 千字节 (Kb) 长度的数据帧，外定序器也仅仅可以展示为包括将被保存 / 恢复的 256 字节的缓冲状态的加速器。

[0053] 对于使用直接存储器存取 (DMA) 协议将数据传送到第一定序器 / 从第一定序器传送数据的辅助单元设备，在一个实施例中，辅助单元和外骨骼之间的专用协议可以保持原始的基于 DMA 的状态，而外定序器可以在辅助单元设备上为第一定序器呈现外骨骼的集合状态和 DMA 结构状态。然后，外骨骼上的 DMA 配置和外定序器“仿真 / 实施”FSM 状态可以经过环境保存 / 恢复（因此通过 OS 环境转换是可虚拟化的）。然而，第一定序器不需要结构上感知任何 DMA 配置信息，其是特定的实现。

[0054] 在另一个实施例中，外定序器（例如，作为用户级体系结构资源）和系统级（例如，特权级别）设备资源可以实际上共享公共的物理原始构件块资源。对于外定序器，公布的逻辑状态可以在构件块中被重命名为相同的物理状态，其是通过读 / 写命令由系统级逻辑设备可访问的。在物理辅助单元内，可以静态地或动态地管理资源的区域 / 部分，并且对于用户级外定序器和系统级设备两者的 ISA 是透明的。

[0055] 具备加密 / 解密加速器的某些服务器平台用于加速网络栈计算。这些隐藏引擎通常物理地与网络处理器耦合，该处理器包括可编程的微引擎以控制隐藏加速器。例如，加密引擎可以包括诸如供散列表计算使用的随机数发生器或伪随机数发生器。在这样一种实现方式中，处理器微引擎作为与隐藏引擎耦合的外骨骼 FSM 可以被重新编程成为一个外定序器。

[0056] 在一个实施例中，可以通过基于虚拟机扩展 (VMX) 的仿真器虚拟机监督系统 (VMM) 来仿真分离的加速器。可以在具有作为外定序器的集合状态的附加体系结构状态的定序器仿真顶上实现用于状态访问和更新的包括环境保存 / 恢复的进入 / 外出方案。从外定序器到第一定序器的异步 NOTIFY 信号发送可以作为 PROXY 方案实现。

[0057] 通过外骨骼，可以表示各种内定序器计算基元的附加进入方案可以被仿真或通过外定序器 FSM 仿真器传送并且被递交给分离的原始物理资源用于计算加速度。实际上，辅助单元的原始计算资源的子集可以由应用程序使用，就像它们是用户级 MD 定序器体系结构资源一样。

[0058] 现在参照图 5，示出了根据本发明的一个实施例一种方法的流程图。如图 5 中所

示，可以在定序器例如，处理器核等等中实现方法 400。如图 5 中所示，可以通过执行开始监视器（例如，SEMONITOR 指令）以便开始监视来自加速器的信号并且把定序器中的事件处理程序与该信号相关联而开始方法 400（方框 405）。具体来讲，这种信号可以是用于指示来自加速器的消息的识别或通知信号。加速器可以是相对于指令定序器的不同种类的资源，例如，是不同的 ISA 或非定序器。因此，可以通过外骨骼在指令定序器和加速器之间通信。因此，可以由定序器通过外骨骼接收从加速器接收的信号。

[0059] 为了配置和启动加速器以代表用户级应用程序执行操作，供应用程序使用的体系结构状态可以被传输给加速器（方框 410），例如，通过 SXFR。例如，指令定序器可以通过外骨骼传输用于与寄存器值相对应的碎片体系结构状态的各种信息，配置信息等等。

[0060] 其次，指令定序器可以为加速器准备命令信息，以便加速器可以代表用户级应用程序执行一个或多个操作（方框 415）。例如，指令定序器可以准备命令信息，其可以进一步的包括将由加速器和 / 或特定的将被应用在数据上的加速功能操作或处理的数据。然后，可以通过内定序器协议（方框 420）例如，包括使用 SXFR 指令传送这个命令信息。更具体地说，这个协议可以具有指令定序器的 ISA。可以使用所述协议直接将这个信息传送到外骨骼，其可以转换所述协议以便底层的数据可以被传送到加速器。

[0061] 在各种实施例中，可以存在执行这种内定序器协议通信的不同方式。例如，在一些实施例中，通信可以包括一个或多个用于存储在缓冲器中的每个基本工序的指令，其可以是在指令定序器和加速器之间共享的存储缓冲器。在其他实施例中，最小的命令信息可以从指令定序器，例如，指令指针 (IP) 中发送到想要在加速器上执行的代码段。然后，假定加速器本身是定序器，加速器可以执行操作以便从指示的位置中读取代码。发送命令信息的间隔尺寸也可以改变。例如，在不同的实施例中，可以以每个命令为基础或以充足的间隔尺寸发送命令信息。在又一个实施例中，第一定序器可以通过外骨骼而不需要通过存储器将本地命令发送到加速器，然后加速器能直接执行所述命令。

[0062] 仍然参考图 5，其次指令定序器可以在指令定序器的命令下执行并行于加速器的操作的互不相关的（例如，独立的）操作（方框 425）。也就是说，在一些实施例中，可以执行并行操作，例如，MD 操作，以便指令定序器和加速器两者都可以并行执行有用功。以这种方式，指令定序器无需等待由加速器产生的结果，并且可以代替地执行其它有用功。当然，指令定序器可以并行执行相关的操作。

[0063] 其次可以确定指令定序器是否已经从加速器中接收了事件（菱形 430）。这种事件可以是通知或其它消息的形式，表示加速器中任务的状态或完成。注意，这个确定无需通过同步的轮询，并且代替的通知可以是异步和事件驱动的并且因而基于非轮询。如果在菱形 430 处没有接收这样的事件，控制可以退回到上述的方框 425。代替的，当接收事件时，指令定序器可以启动事件处理程序（例如，在方框 405 中通过 SMONITOR 的最初记录）以便接收和处理来自于加速器的数据（方框 440）。在各种实现方式中，这个事件处理程序可以是轻度的用户级处理程序，以致避免了 OS 调用或环境转换的系统开销。以这种方式，可以存在改善的操作并且可以从加速器中获得结果数据并且用作用户级应用程序所期望的。可以在记录了下一个指令指针之后发生到处理程序的控制传输（例如，推到栈中），以致可以在事件处理程序完成之后恢复暂停的执行。

[0064] 现在参考图 6，示出了根据本发明的实施例与具有外骨骼的加速器中执行的操作

相应的方法的流程图。如上所述，这个加速器可以具有相对于定序器不同种类的特性。因此，外骨骼可以提供内定序器协议通信。响应方框 410（图 5 的，如上所述），加速器可以从定序器中接收碎片体系结构状态信息。因此，可以根据这个信息配置加速器（方框 460）。因为加速器的不同种类的特性，外骨骼可以用于接收这个内定序器通信并且转换它，以便可以由加速器处理它。

[0065] 其次，响应方框 420（图 5 的），可以通过外骨骼从定序器中接收命令信息（方框 465）。该命令信息可以包括控制和 / 或配置信息以及将被处理的数据。因此，加速器可以按照所述命令信息执行操作（方框 470）。可以并行于独立执行操作的定序器执行这些操作。也就是说，定序器无需在执行其它有用功之前等待加速器完成它的操作。如上所述，可以以 MIMD 方式执行定序器和加速器中的操作，并且在一些实现方式中可以对应于互不相关的操作。

[0066] 当加速器完成它的操作时，外骨骼可以相应地通知定序器（方框 475）。然后，在用户级代码的控制下，例如，启动用户级事件处理程序的轻度用户级产生机构，可以通过外骨骼将与加速器中获取的各种结果相应的数据传输到定序器（方框 480）。虽然在图 5 和图 6 的实施例中用这个具体的流程进行了描述，应当理解的是，本发明的范围不因此受限。

[0067] 使用根据本发明的实施例的外定序器，可以执行不同级别的提取。例如，在一些实现方式中，虚拟或复杂指令集计算机 (CISC) 指令可以发送给外骨骼，然后其执行扩展以便将这种指令映射到加速器的本地的物理指令或命令序列，该加速器可以是不同于第一定序器的 ISA 的定序器或根本是一个非定序器。因此，如果加速器的基本命令集随时间被修改或改善，则仍然可以通过这种提取的级别提供传统支持。以这种方式，即使对于传统用户级应用程序也可以实现在加速器上提高性能。在其它实现方式中，可以从定序器将直接或精简指令集计算 (RISC) 指令发送到可以直接执行加速器硬件上的指令的外定序器。

[0068] 可以以许多不同的系统类型实现这些实施例。现在参考图 7，示出了根据本发明的实施例的系统的方框图。如图 7 中所示，多处理器系统 500 是点对点互连系统，并且包括通过点对点互连 550 耦合的第一处理器 570 和第二处理器 580。如图 7 中所示，处理器 570 和 580 中的每一个可以是多核心处理器，包括第一和第二处理器核心（即，处理器核心 574a 和 574b 以及处理器核心 584a 和 584b）。处理器 570 和 580 中的每一个可以进一步包括外定序器，即，第一外定序器 575 和第二外定序器 585。如上所述，外定序器 575 和 585 可以是相对于处理器核心 570 和 580 的剩余资源的不同种类的资源。虽然仅仅用每个处理器的单个外定序器示出，可以理解的是，本发明的范围不因此受限。在其它实施例中，给定的处理器中可以存在多个外定序器。此外，一个或多个外定序器可以与处理器的每个独立核心有关。

[0069] 第一处理器 570 进一步包括存储控制器集线器 (MCH)572 和点对点 (P-P) 接口 576 和 578。类似地，第二处理器 580 包括 MCH 582 和 P-P 接口 586 和 588。如图 7 中所示，MCH 的 572 和 582 将处理器耦合到各自的存储器，即存储器 532 和存储器 534，其可以是与各自的处理器本地连接的主存储器的一部分。

[0070] 第一处理器 570 和第二处理器 580 可以分别通过 P-P 互连 552 和 554 与芯片组 590 耦合。如图 7 中所示，芯片组 590 包括 P-P 接口 594 和 598。此外，芯片组 590 包括接口 592，用于将高性能图形引擎 538 与芯片组 590 耦合。在一个实施例中，加速图形端口

(AGP) 总线 539 可以用于将图形引擎 538 耦合到芯片组 590。AGP 总线 539 可以遵循由加利福尼亚州，圣克拉拉，英特尔公司于 1998 年 5 月 4 日公布的加速图形端口接口规格，修订版 2.0(AcceleratedGraphics Port Interface Specification, Revision 2.0)。另一方面，点对点互连 539 可以耦合这些组件。

[0071] 依次，芯片组 590 可以通过接口 596 与第一总线 516 耦合。在一个实施例中，第一总线 516 可以是外围设备组件互连 (PCI) 总线，正如 PCI 本地总线规约，生产版本，修订版 2.1，注明日期 1995 年 6 月所定义的，或者诸如 PCIExpress 总线或另外的第三代 I/O 互连总线之类的总线，尽管本发明的范围不因此受限。

[0072] 如图 7 中所示，各种 I/O 设备 514 可以耦合到第一总线 516，以及将第一总线 516 耦合到第二总线 520 的总线桥接器 518。在一个实施例中，第二总线 520 可以是低引脚计数 (LPC) 总线。各种设备可以耦合到第二总线 520，各种设备包括例如，键盘 / 鼠标 522、通信设备 526 以及诸如在一个实施例中可以包括代码 530 的磁盘驱动器或其它大容量存储设备之类的数据存储单元 528。进一步，音频 I/O 524 可以与第二总线 520 耦合。注意，其它体系结构是可能的。例如，代替图 7 的点对点体系结构，系统可以实现多支路总线或另外的这种体系结构。

[0073] 可以用代码实现实施例并且可以存储到已经在其上存储了指令的存储介质上，存储的指令可被用于编程系统以执行所述指令。存储介质可以包括但不局限于任何类型的盘，其包括软盘、光盘、高密度盘只读存储器 (CD-ROM)、高密度可重写盘 (CD-RW)、和磁光盘，诸如只读存储器 (ROM)，诸如动态的随机存取存储器 (DRAM)、静态的随机存取存储器 (SRAM) 的随机存取存储器 (RAM)，可擦除可编程只读存储器 (EPROM)，闪存，电可擦除可编程只读存储器 (EEPROM)，磁或光卡之类的半导体装置，或任何其它类型的适用于存储电子指令的介质。

[0074] 虽然本发明已经描述了相对有限数目的实施例，本领域中的那些普通技术人员可以理解的是，可以由此产生大量的修改和变化。意图是附加的权利要求覆盖了所有的这种修改和变化，正如落在本发明的真实精神和范围内。

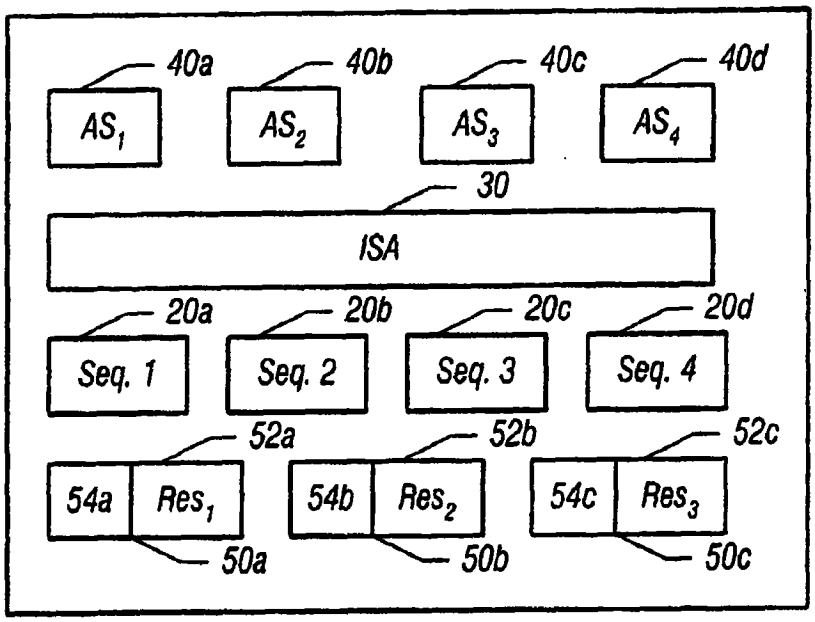

10
40a
40b
40c
40d
AS1
AS2
AS3
AS4
30
ISA
20a
20b
20c
20d
Seq. 1
Seq. 2
Seq. 3
Seq. 4
52a
52b
52c
54a
Res1
54b
Res2
54c
Res3
50a
50b
50c

图 1

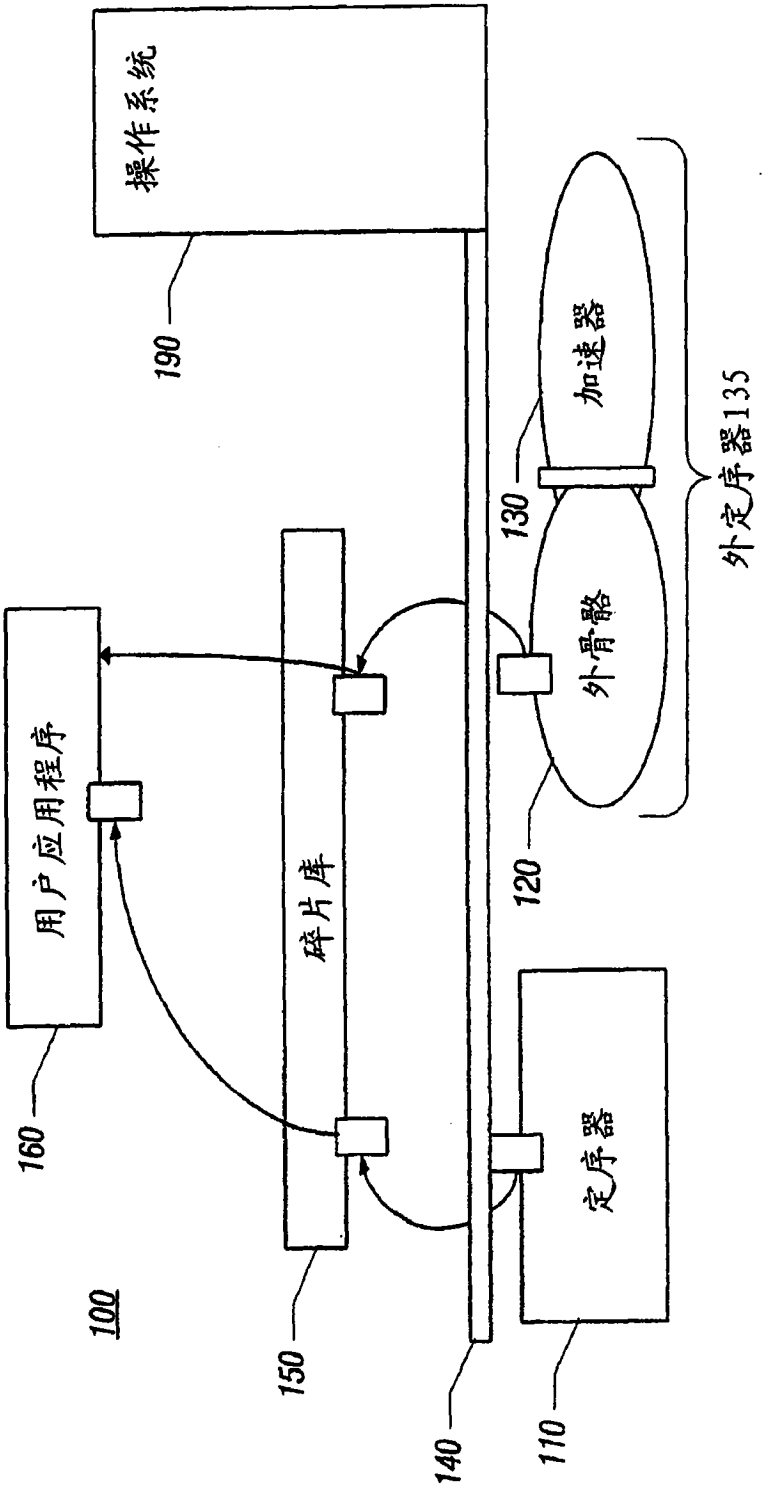

操作系统
190
加速器
130
外骨骼
120
外定序器135
用户应用程序
160
碎片库
150
定序器
110
140
100

图 2

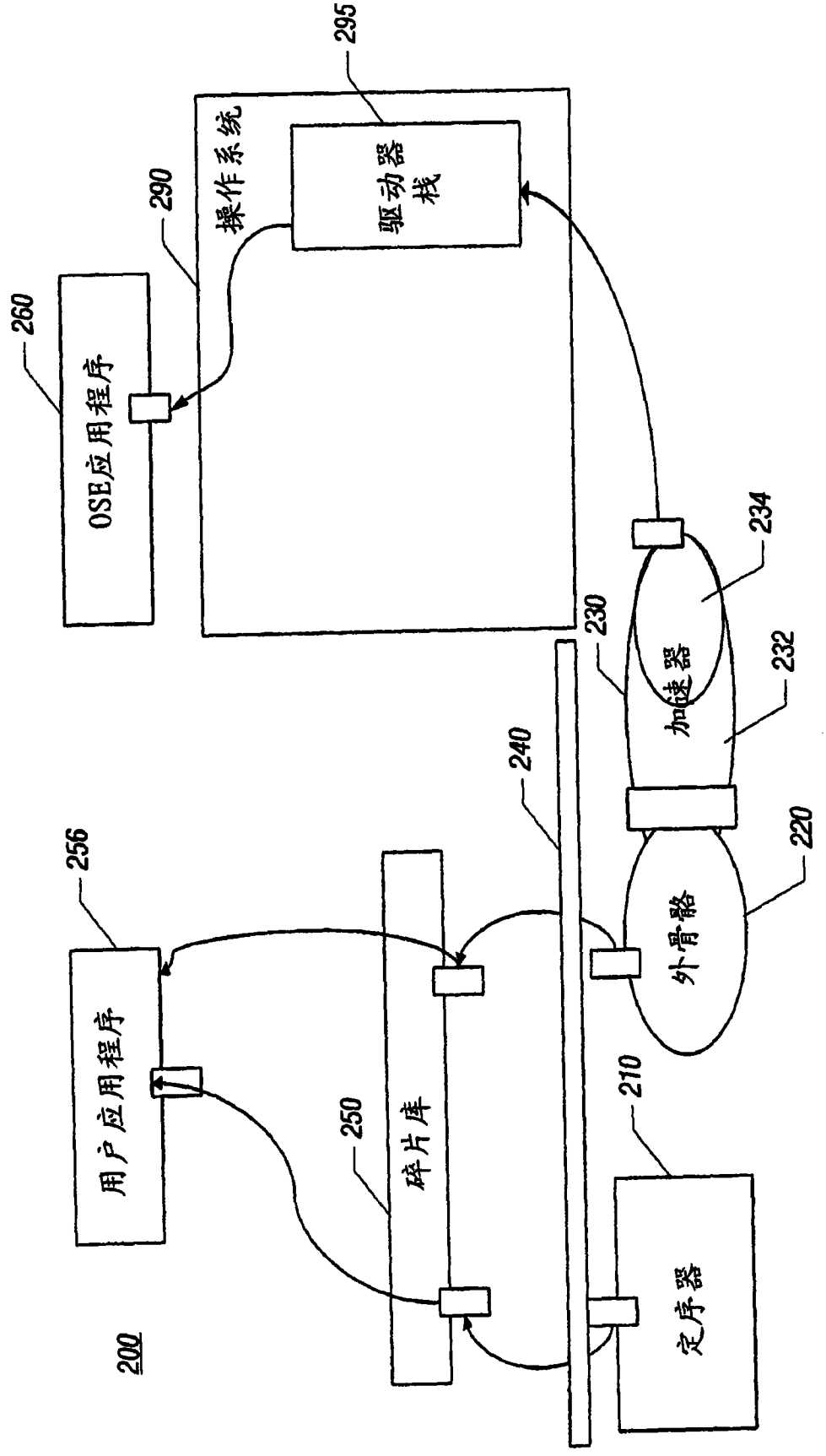

295
操作系统
驱动器栈
290
260
OSE应用程序
234
加速器
230
232
240
220
外骨骼
256
用户应用程序
250
碎片库
210
定序器
200

图 3

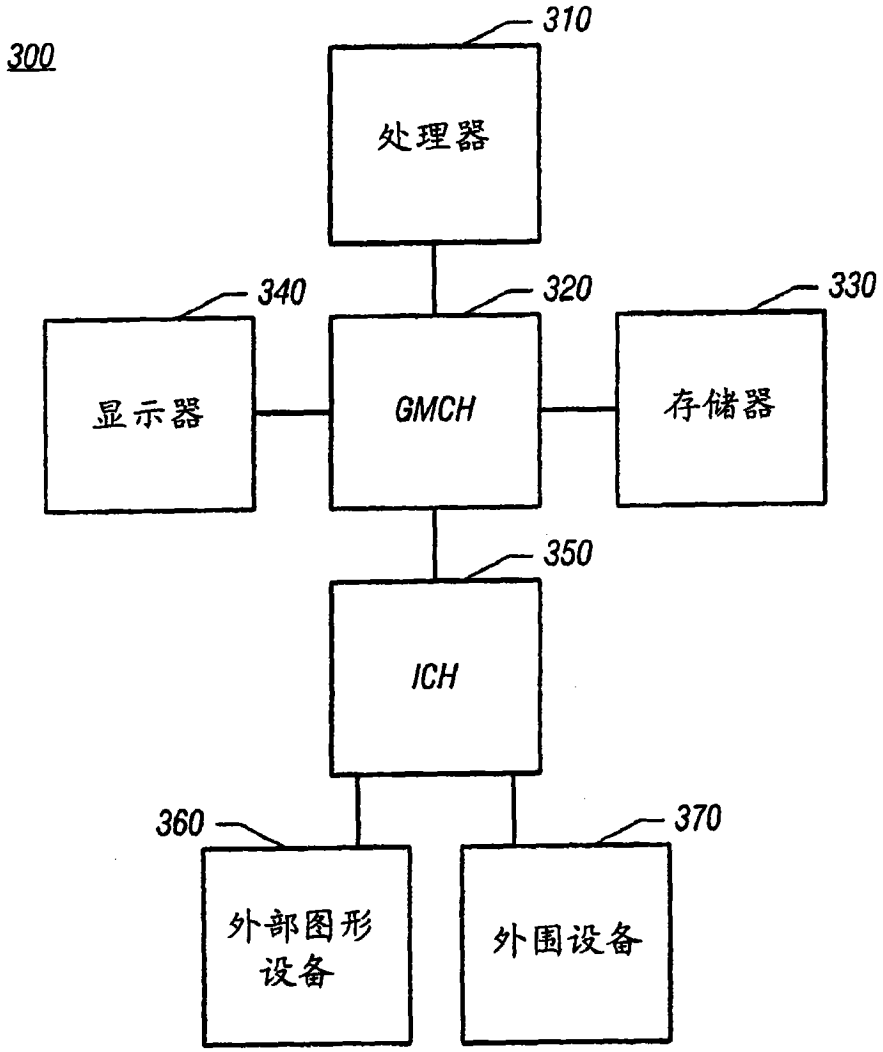

300
310
处理器
340
320
330
显示器
GMCH
存储器
350
ICH
360
370
外部图形设备
外围设备

图 4

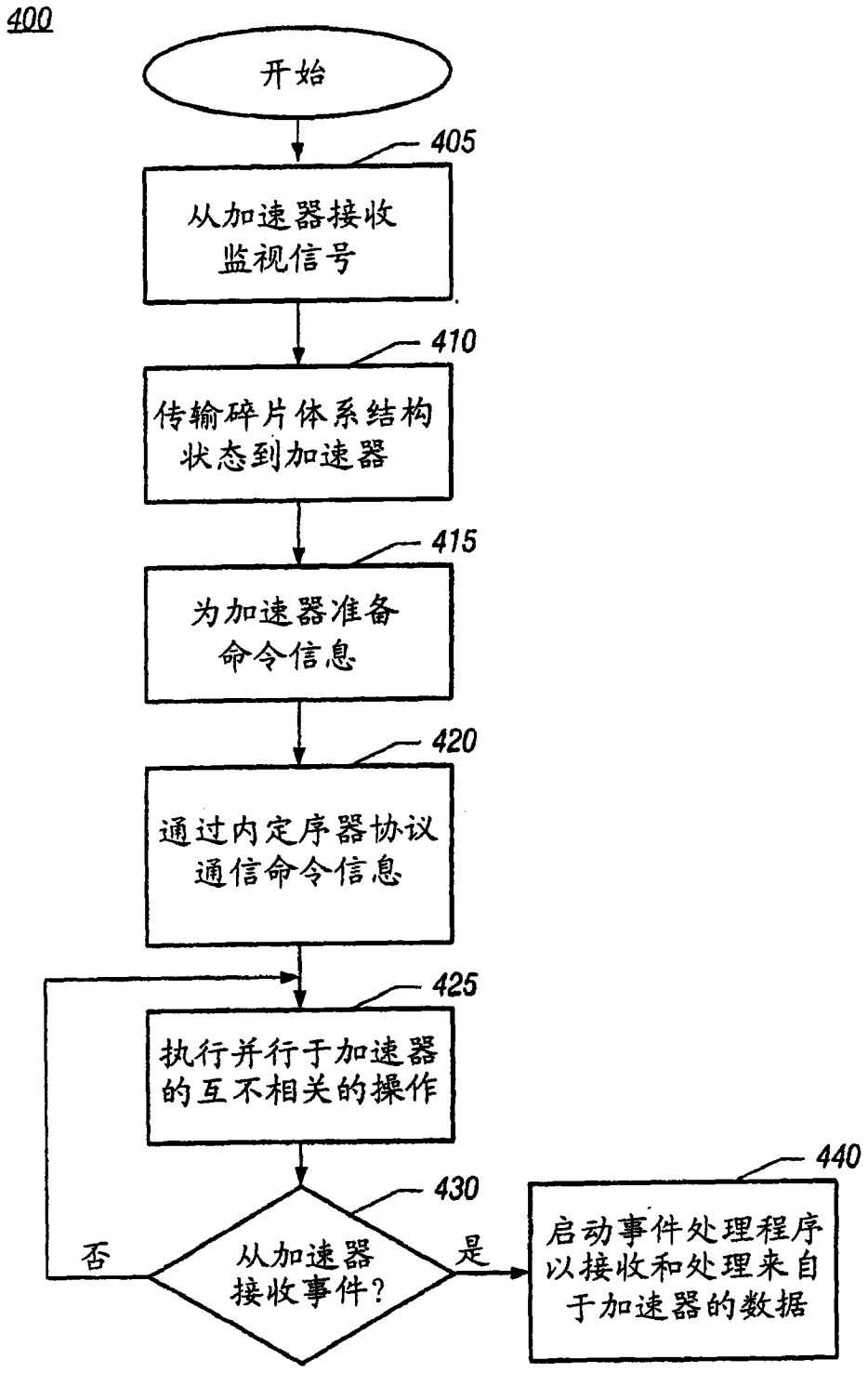

400
开始
405
从加速器接收监视信号
410
传输碎片体系结构状态到加速器
415
为加速器准备命令信息
420
通过内定序器协议通信命令信息
425
执行并行于加速器的互不相关的操作
430
从加速器接收事件?
否
是
440
启动事件处理程序以接收和处理来自于加速器的数据

图 5

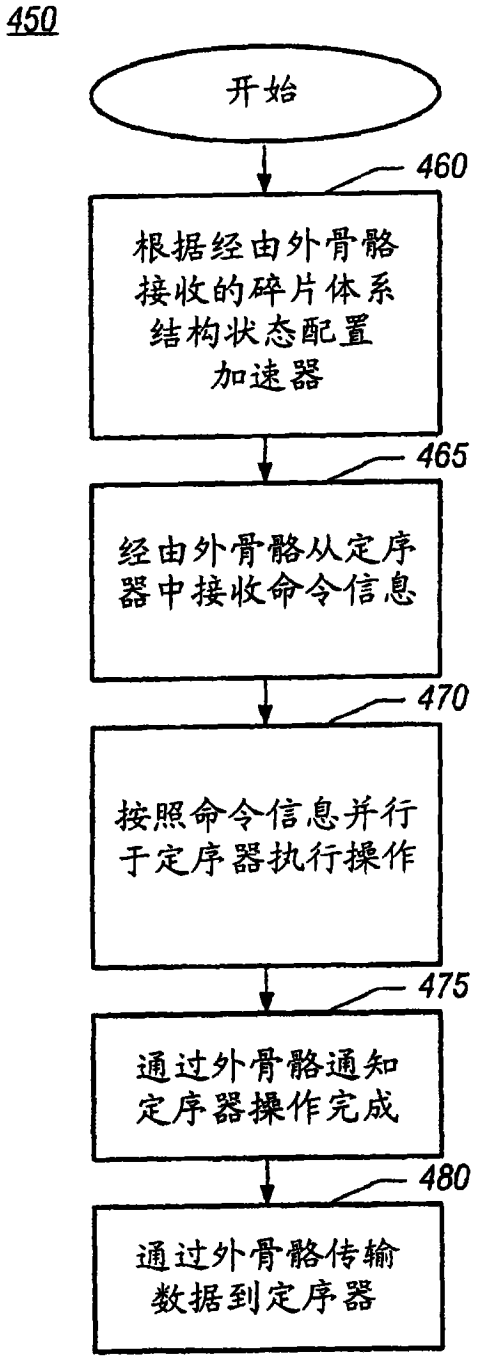

450
开始
460
根据经由外骨骼接收的碎片体系结构状态配置加速器
465
经由外骨骼从定序器中接收命令信息
470
按照命令信息并行于定序器执行操作
475
通过外骨骼通知定序器操作完成
480
通过外骨骼传输数据到定序器

图 6

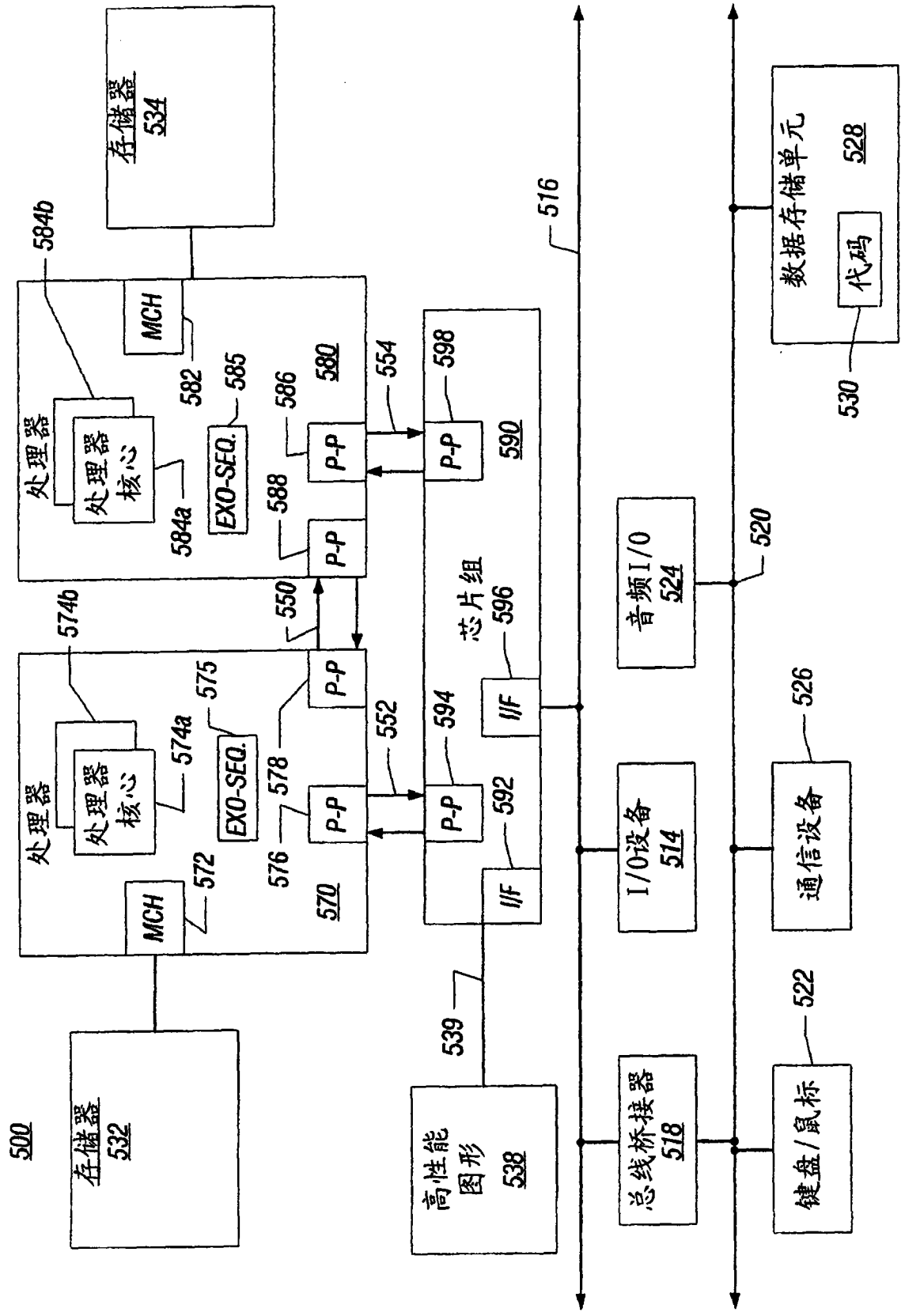

500
在储器 532
MCH
572
处理器
处理器核心
574a
574b
EXO-SEQ.
575
P-P
576
578
570
552
550
554
处理器
处理器核心
584a
584b
EXO-SEQ.
585
586
588
P-P
580
582
MCH
在储器 534
598
P-P
590
芯片组
596
I/F
594
P-P
592
I/F
539
高性能图形 538
516
音频I/O 524
520
I/O设备 514
526
通信设备
总线桥接器 518
522
键盘/鼠标
数据存储单元
代码
528
530

图 7