(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 104583959 A
(43)申请公布日 2015.04.29

(21)申请号 201380044102.7

(22)申请日 2013.06.29

(30)优先权数据
13/626,441 2012.09.25 US

(85)PCT国际申请进入国家阶段日
2015.02.17

(86)PCT国际申请的申请数据
PCT/US2013/048793 2013.06.29

(87)PCT国际申请的公布数据
WO2014/051818 EN 2014.04.03

(71)申请人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 V·尚伯格 S·J·鲁滨逊

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 张东梅

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

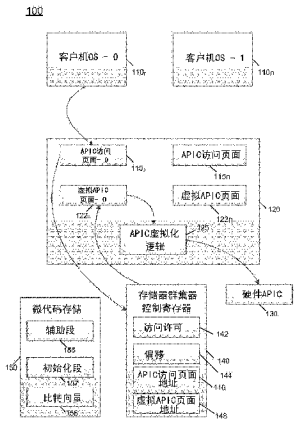
权利要求书2页 说明书11页 附图8页

(54)发明名称

允许处理器资源的虚拟化

(57)摘要

在一个实施例中，处理器包括访问逻辑，以判断来自虚拟机的访问请求是否是与处理器的设备相关联的设备访问页面，如果是，则至少部分地基于存储在处理器的控制寄存器中的信息，将访问请求重新映射到与VM相关联的系统存储器中的虚拟设备页面。描述并要求保护其他实施例。



CN 104583959 A

1.一种处理器,包括 :

用于执行指令的执行装置 ;

控制寄存器装置,包括第一指示符字段,以存储第一许可指示符以指出对系统存储器中的高级可编程中断控制器 (APIC) 访问页面的仿真的存储器访问是否被允许,第一地址字段,以存储标识所述 APIC 访问页面内的偏移的偏移值 ;以及

用于判断访问请求是否针对所述 APIC 访问页面的装置,如果是,则将所述访问请求重新映射到与所述访问请求的请求者相关联的所述系统存储器中的虚拟 APIC(vAPIC) 页面。

2.如权利要求 1 所述的处理器,其特征在于,所述第一许可指示符包括读取访问许可指示符,所述控制寄存器装置进一步包括第二指示符字段以存储指出对所述 APIC 访问页面的仿真的写入访问是否被允许的第二许可指示符。

3.如权利要求 1 所述的处理器,其特征在于,所述控制寄存器装置进一步包括第二地址字段以存储对应于所述系统存储器中的所述 vAPIC 页面的地址的 vAPIC 页面地址。

4.如权利要求 1 所述的处理器,其特征在于,进一步包括微代码存储装置,包括辅助段,以响应于所述访问请求不匹配所述偏移值,编程所述控制寄存器装置。

5.如权利要求 4 所述的处理器,其特征在于,所述微代码存储装置还包括初始化段,以在进入客户机时,将第一值存储在所述第一地址字段中,并且将第一指示符值存储在所述第一指示符字段,以指出对所述 APIC 访问页面的所述仿真的存储器访问被允许。

6.如权利要求 4 所述的处理器,其特征在于,进一步包括比特向量,该比特向量包括多个字段,每一个字段都指出所述 APIC 访问页面内的偏移是否有效。

7.如权利要求 6 所述的处理器,其特征在于,所述辅助段将允许所述处理器至少部分地基于所述比特向量,判断所述访问请求是否针对所述 APIC 访问页面内的有效偏移,如果所述访问请求不是针对有效偏移,则导致从虚拟机退出。

8.如权利要求 7 所述的处理器,其特征在于,所述辅助段将响应于所述访问请求针对所述有效偏移,而编程所述控制寄存器装置。

9.如权利要求 7 所述的处理器,其特征在于,所述辅助段将导致所述处理器从所述处理器的前端单元重新启动所述访问请求,所述前端单元包括指令缓存,其中所述指令缓存不是包括性的缓存。

10.如权利要求 1 所述的处理器,其特征在于,所述请求者包括虚拟机。

11.一种方法,包括 :

接收对处理器的访问逻辑中的系统存储器的地址的存储器访问请求,并判断所述地址是否在所述处理器的设备的设备访问页面内 ;

如果是,则判断所述存储器访问请求是否是针对具有存储在所述处理器的控制寄存器的许可字段中的有效许可指示符的类型,所述地址的一部分对应于存储在所述控制寄存器的第一地址字段中的仿真的偏移值 ;以及

如果是,则将所述地址重新映射到与所述存储器访问请求的请求者相关联的所述系统存储器的虚拟设备页面。

12.如权利要求 11 所述的方法,其特征在于,进一步包括使用从所述控制寄存器的第二地址字段中获取的所述虚拟设备页面的地址,重新映射所述地址。

13.如权利要求 12 所述的方法,其特征在于,进一步包括从所述请求者的虚拟机控制

结构获取所述虚拟设备页面地址，并将所述虚拟设备页面地址存储在所述第二地址字段中。

14. 如权利要求 13 所述的方法，其特征在于，进一步包括使用所述处理器的微代码的虚拟化初始化段，获取并存储所述虚拟设备页面地址。

15. 如权利要求 11 所述的方法，其特征在于，进一步包括，如果所述存储器访问请求不是针对具有有效许可指示符的类型的或所述地址的一部分不对应于仿真的偏移值，则调用所述处理器的微代码的虚拟化辅助段，以判断所述存储器地址访问请求是否将针对所述设备访问页面内的仿真的偏移，如果是，则利用所述许可指示符和所述仿真的偏移值，编程所述控制寄存器，并在所述处理器的前端单元中重新启动所述存储器访问请求。

16. 如权利要求 15 所述的方法，其特征在于，进一步包括，如果所述存储器访问请求不针对所述设备访问页面内的所述仿真的偏移，则导致从客户机退出到在所述处理器上执行的虚拟机监视器。

17. 一种系统，包括 ：

处理器，包括用于判断来自虚拟机 (VM) 的访问请求是否针对与所述处理器的设备相关联的设备访问页面的访问装置，如果是，则至少部分地基于存储在所述处理器的控制寄存器中的信息，将所述访问请求重新映射到与所述 VM 相关联的系统存储器中的虚拟设备页面，存储在所述处理器的控制寄存器中的信息包括第一指示符字段，以存储第一许可指示符，以指出所述访问请求是否被允许，以及第一地址字段，以存储标识所述设备访问页面内的有效偏移的偏移值 ；以及

所述系统存储器包括耦合到所述处理器的动态随机存取存储器 (DRAM)。

18. 如权利要求 17 所述的系统，其特征在于，所述控制寄存器进一步包括第二地址字段，以存储对应于所述系统存储器中的所述虚拟设备页面的地址的虚拟设备页面地址。

19. 如权利要求 17 所述的系统，其特征在于，进一步包括微代码存储装置，包括辅助段，以响应于所述访问请求的地址的一部分不匹配所述偏移值，编程所述控制寄存器。

20. 如权利要求 19 所述的系统，其特征在于，所述微代码存储装置进一步包括 ：

初始化段，用于在进入所述 VM 时，将第一值存储在所述第一地址字段中，将第一指示值存储在所述第一指示符字段，以指出对所述设备访问页面的第一类型的访问请求被允许 ；以及包括多个字段的比特向量，每一个字段都指出来自 VM 的到所述设备访问页面内的偏移的访问请求是否被允许，其中，所述辅助段将允许所述处理器至少部分地基于所述比特向量，判断对所述设备访问页面内的第一偏移的所述访问请求是否被允许，如果所述访问请求不被允许，则导致从所述虚拟机退出。

21. 被配置为执行如权利要求 11 到 16 中任一项所述的方法的通信设备。

22. 至少一个机器可读介质，所述至少一个机器可读介质包括多个指令，响应于在计算设备上被执行，所述指令导致所述计算设备执行如权利要求 11 到 16 中任一项所述的方法。

23. 一种用于处理指令的设备，被配置成执行如权利要求 11 到 16 中任一权利要求所述的方法。

24. 一种设备，包括用于执行根据权利要求 11 到 16 中的任一项所述的方法的装置。

允许处理器资源的虚拟化

背景技术

[0001] 常规虚拟机监视器 (VMM) 在计算系统上执行，并向其他软件呈现一个或多个虚拟机 (VM) 的抽象。每一个虚拟机都可以充当自包含平台，运行由 VMM 主控的其自己的客户机操作系统 (OS) 及其他软件，统称为客户机软件。客户机软件预期好像它在专用计算机上运行而并非在虚拟机上运行。即，客户机软件预期控制各种事件并有权访问硬件资源。硬件资源可以包括处理器驻留资源，驻留在存储器中的资源，以及驻留在底层硬件平台上的资源。事件可以包括中断、异常、平台事件、某些指令的执行等等。

[0002] 在虚拟机环境中，VMM 应该能够对这些事件和硬件资源具有最终控制，以提供在虚拟机上运行的客户机软件的正确操作以及来自虚拟机上运行的客户机软件的保护及虚拟机之间的保护。为实现此，当客户机软件访问受保护资源时或当发生其他事件（诸如中断或异常）时，VMM 通常接收控制。例如，当由 VMM 支持的虚拟机中的操作导致系统设备生成中断时，当前正在运行的虚拟机被中断，对处理器的控制被传递到 VMM。然后，VMM 接收中断，并处理中断本身，或将中断提供到合适的虚拟机。然而，在客户机和 VMM 之间的此切换是对处理器周期的低效率使用。

[0003] 附图简述

[0004] 图 1 是根据本发明的一实施例的系统的一部分的框图。

[0005] 图 2 是根据本发明的一个实施例的方法的流程图。

[0006] 图 3 是根据本发明的一个实施例的用于执行 APIC 虚拟化微代码操作的方法的流程图。

[0007] 图 4 是根据本发明的一个实施例的虚拟机环境的一个实施例的框图。

[0008] 图 5 是根据本发明的一个实施例的处理器核的框图。

[0009] 图 6 是根据本发明的一个实施例的处理器的框图。

[0010] 图 7 是包括多个核的处理器的一个实施例的框图。

[0011] 图 8 是根据本发明的一个实施例的系统的框图。

具体实施方式

[0012] 在各实施例中，诸如多核处理器之类的处理器的一个或多个资源可以使用如此处所描述的控制结构和逻辑来虚拟化。更具体而言，各实施例涉及诸如高级可编程中断控制器 (APIC) 之类的存储器映射的处理器资源的虚拟化。一般而言，APIC 是处理器的硬件资源，诸如微控制器或从各种代理（处理器内部和外部的）接收传入的中断，并按优先级处理以及选择用于处理中断的合适的资源（诸如处理器的一个或多个核）的逻辑。

[0013] 通过使用本发明的各实施例，此 APIC 可以被虚拟化，以便多个虚拟代理，例如，在处理器上执行的单个客户机，可以以时间切片方式访问 APIC。

[0014] 如下面所进一步描述的，为实现 APIC 虚拟化，多个地址可以被存储到虚拟机的控制结构中，此处被称为虚拟机控制结构 (VMCS)，它可以被提供用于在处理器上运行的每一个虚拟机，并可以存储在系统存储器中。即，这些地址可以包括物理访问页面地址和虚拟访

问页面地址。物理访问页面地址是客户机 OS 向其执行输入 / 输出 (IO) 操作的物理地址。在虚拟化 APIC 的示例中，这是客户机向其执行存储器映射的 IO 操作的地址。虚拟访问页面地址是存储器中的存储了虚拟化设备的状态的页面的地址。在虚拟化 APIC 的示例中，此页面存储虚拟 APIC 的状态。具体而言，在虚拟化 APIC 的示例中，物理访问页面可以对应于 APIC 访问页面，虚拟访问页面可以对应于虚拟 APIC 页面。然而，更一般而言，这些地址可以是用于设备访问页面和虚拟设备页面的。对此 APIC 访问页面内的有效位置的客户机访问可以通过将这样的访问重新映射到虚拟 APIC 页面来仿真。如此，写入访问将数据存储在虚拟 APIC 页面内的对应的偏移处，加载访问将来自虚拟 APIC 页面中的对应的偏移处的数据返回到请求者。对 APIC 访问页面内的无效位置的访问会导致从虚拟机退出，以便虚拟机监视器或其他主管软件可以处理该状况。

[0015] 为有效率地执行这样的操作，各实施例可以提供对处理器的组合的硬件控制的和微代码控制的机制。更具体而言，当检测到对 APIC 访问页面的访问时，在某些情况下，可以调用此微代码。一般而言，此微代码可以验证访问的各种属性是否有效，如果是，则在对硬件编程以将 APIC 访问页面重新映射到虚拟 APIC 页面内的偏移之后，重新启动指令执行，以便在接下来对 APIC 访问页面访问时进行有效率的访问。

[0016] 另外，各实施例还进一步提供可以代替微代码辅助使用的硬件，以维护正确操作，甚至在指令缓存中不包括指令的情况下，其中，可能会由于从重新映射后变化的指令缓存获取指令而发生不正确的操作。如此，各实施例可以组合虚拟化硬件和微代码辅助机制，以甚至在指令缓存中不保证包括指令的情况下确保正确操作。

[0017] 在各实施例中，处理器的存储器集群内的访问逻辑可以使用来自根据本发明的一个实施例的控制寄存器的信息，提供 APIC 访问页面上的可编程偏移的匹配。当客户机被进入时（例如，通过 VMLAUNCH 或 VMRESUME 指令），微代码可以利用偏移将此控制寄存器的页面地址字段中的第一地址字段（例如，EMULATED_REG_OFFSET 字段）编程到 APIC 访问页面内的预定的偏移值（例如，0x80H）。另外，作为默认，控制寄存器可以被编程为通过存储在许可字段（例如，EMULATE_READ 和 EMULATE_WRITE 字段）中的第一和第二许可指示符提供读和写访问的仿真。在 VM 条目上，APIC 访问页面地址和 vAPIC 页面地址被从 VMCS 读出，并写入到这些控制寄存器中。

[0018] 然后，在检测到对 APIC 访问页面的访问时，此访问请求的地址可以被重新映射到 vAPIC 页面中的对应的偏移，例如，对应于存储在控制寄存器中的地址（即，可以在进入中客户机时，从 VMCS 中获取的 vAPIC 页面地址）。

[0019] 然后，存储器集群按如下方式针对在控制寄存器中编程的那些验证访问的许可：如果访问是加载，并且访问地址的比特 11:0 匹配控制寄存器内的 EMULATED_REG_OFFSET 值，EMULATE_READ 指示符被编程为置位（例如，逻辑 1），则访问被允许；而如果访问是存储，并且访问地址的比特 11:0 匹配 EMULATES_REG_OFFSET 值，EMULATE_WRITE 指示符被编程为 1，则访问被允许。当访问被允许时，存储器集群重新映射加载或存储的地址，以便它对虚拟 APIC 页面被执行。

[0020] 如果不满足这些条件（访问不指向由微代码编程的偏移，访问是加载，EMULATE_READ 是 0，或访问是存储，EMULATE_WRITE 是 0），则硬件可以调用微代码辅助。硬件还向微代码辅助提供访问的细节，例如，偏移、大小和类型（读 / 写）。

[0021] 在此微代码辅助中，可以评估访问的偏移、大小和类型（加载或存储）以判断访问是否将指向有效位置。如果是，则微代码可以将合适的值（例如，关于许可和 APIC 访问页面偏移）编程到控制寄存器并重新启动指令。

[0022] 如果重新启动的指令与最初导致辅助的指令相同，则访问现在将匹配存储器集群，并将被仿真并重定向到 vAPIC 页面。匹配为仿真允许的类型的此偏移的任何后续指令都将继续被仿真，并重新映射，无需微代码辅助。此行为允许重新映射在硬件中发生，无需对于最常访问的偏移的微代码辅助。

[0023] 现在参考图 1，所示是根据本发明的实施例的系统的一部分的框图。图 1 所示出的系统的一部分详述了软件和硬件的各层之间的交互。具体而言，一个或多个客户机操作系统 (OS) 可以在对应的虚拟机 110_0-110_n（一般地，VM 或客户机 110）内操作。当然，额外的软件也可以在每一个 VM（诸如各个客户机软件）内执行，例如，一个或多个用户级别的应用程序。

[0024] 与这里的讨论相关，在客户机 110_0 中执行的客户机 OS 可以表面上向存储器结构发出访问请求。然而，由于虚拟化，代替直接与诸如硬件 APIC130 之类的底层硬件进行通信，相反可以对 APIC 访问页面 1150 进行访问，APIC 访问页面 115_0 是虚拟机监视器 (VMM) 120 内的被映射到客户机地址空间的有效物理地址。此访问页面可以是与给定客户机相关联的物理存储器中的页面。注意，可以有每个客户机地提供的 APIC 访问页面。

[0025] 此对 APIC 访问页面的访问 115 又可以导致对 VMM 120 所使用的系统存储器内的虚拟 APIC(vAPIC) 页面 1220 的访问。即，取决于在此页面内访问指向的偏移，VMM 的访问逻辑 125（此处也被称为 APIC 虚拟化逻辑）可以判断是否允许此访问流向底层硬件 APIC 130（可以是处理器的存储器映射的硬件结构）。

[0026] 为允许这样的访问发生，首先，可以判断来自客户机的访问是否被允许。为此，可以提供额外的硬件，即，控制寄存器 140（其可被实现为处理器的存储器集群的一个或多个控制寄存器），以存储关于正在执行的特定客户机的访问许可、地址信息等等的信息。在所示出的实施例中，控制寄存器 140 可以包括访问许可存储 142。在各实施例中，此存储可以包括一个或多个字段，以存储指出是否允许对 APIC 访问页面的特定类型的访问的指示符。在一个实施例中，此许可存储可包括读取许可指示符和写入许可指示符。另外，还可以存在偏移存储 144。偏移存储可以存储到要仿真的 APIC 访问页面内的一个位置的偏移。还可以提供 vAPIC 存储 146。此存储可以被用来存储对应的客户机的 vAPIC 页面的基本地址。此外，还可以提供 vAPIC 存储。此存储可以被用来存储仿真的访问将被重新定向到的 vAPIC 页面的基本地址。

[0027] 如此，在虚拟化环境内的对硬件 APIC 130 的访问被准许之前，可以至少部分地基于存储在控制寄存器 140 中的信息，判断访问是否被允许。另外，如下文进一步描述的，处理器可以包括微代码存储 150，该微代码存储 150 包括要由处理器的底层硬件执行的各种微代码。作为此微代码的一部分，可以提供辅助微代码段 155，该辅助微代码段 155 可用于帮助对于如此处所描述的 APIC 流的仿真操作。也用于仿真操作，可以进行对微代码存储器内的比特向量 156 的访问，下面进一步描述了其细节。微代码也可以包括初始化段 157，以对于处理器执行初始化操作，包括为控制寄存器编程默认值，并在进入 VM 时执行进一步的操作。还要注意，尽管此处所描述的实施例中的硬件设备仿真是针对 APIC 的，但是，应理

解，本发明的范围在这方面不受限制，在其他实施例中，可以仿真其他类型的硬件设备，如网络控制器。尽管在图 1 的实施例以这样高级别地表示，然而要理解本发明的范围不限于此方面。

[0028] 现在参考图 2，所示是根据本发明的一个实施例的方法的流程图。如图 2 所示，根据本发明的一个实施例，方法 200 可以由处理器的存储器集群内的逻辑执行，该逻辑可以处理传入的存储器请求并判断是否允许访问。如图 2 所示，方法 200 可以从判断在存储器集群中接收到的访问请求是否命中 APIC 访问页面开始。如上文所描述的，APIC 访问页面可以是 VMM 物理地址空间中的被映射到客户机的物理地址空间的页面。如果访问不是针对这样的页面，则控制进入框 215，在那里，可以执行用于处理访问的常规操作，例如，加载或存储访问。

[0029] 如果相反判断访问请求命中 APIC 访问页面，则控制进入菱形 220，以判断与从其执行访问的 VM 的此 APIC 访问页面相对应的虚拟 APIC 页面是否有效。在一个实施例中，此判断可以基于控制寄存器中的有效比特。作为示例，除 vAPIC 页面地址之外，图 1 所示出的控制寄存器 140 可以包括指出存储在控制寄存器的此部分中的地址是否有效的有效比特（也可以存储在控制寄存器中）。如果是，则控制接下来进入菱形 230，在那里，可以判断访问是否指向仿真的寄存器。虽然本发明的范围在这方面不受限制，但是，此判断可以基于各种构成判断。具体而言，在一个实施例中，这些判断可以包括：访问是否是针对数据读取或写入的（不是指令获取）；访问大小至多是 32 比特；访问包含在天然地校准的 16 字节区域的低 4 字节内；访问地址的比特 11:0 匹配 EMULATED_REG_OFFSET（例如，存储在偏移字段 144）；访问是数据读取，EMULATED_READ 是 1，或访问是数据写入，EMULATED_WRITE 是 1（例如，两者都存储在访问许可字段 142 中）。

[0030] 如此，如果菱形 230 中的判断肯定访问将指向仿真的寄存器，则控制进入框 240，在那里，访问请求的地址，例如，加载 / 存储访问操作可以被重新映射到 vAPIC 页面中的偏移。如此，对于读取操作，可以从虚拟 APIC 页面内的对应的偏移中读取请求的信息，并报告回客户机。或者，如果它是存储请求，则写入请求的对应的数据可以存储在 vAPIC 页面中的偏移处，即，vAPIC 页面中的对应于 APIC 访问页面中的偏移的偏移处，换言之，由访问的比特 11:0 所指定的偏移。

[0031] 仍参考图 2，从菱形 220 和菱形 230，如果这些点中的任何一个的判断是否定的，则控制进入框 225，在那里，可以调用 APIC 虚拟化微代码辅助。下面进一步描述此辅助的进一步的细节。

[0032] 现在参考图 3，所示是根据本发明的一个实施例的用于执行 APIC 虚拟化微代码操作的方法的流程图。如图 3 所示，方法 250 可以从判断被请求的访问是否指向仿真的偏移开始。此判断，与图 3 所示出的所有其他操作相同，可以通过微代码来执行。更具体而言，给定处理器可以包括微代码段，此处被称为微代码辅助段，该段包括用于在处理器的硬件上执行以有效率地执行所示出的操作的微代码。在一个实施例中，对仿真的偏移的访问的此判断可以基于 APIC 访问页面内的有效位置的比特向量。在一个实施例中，此比特向量可以存储在非易失性存储器中，例如，作为处理器的微代码的一部分。在另一个实施例中，此比特向量可以由 VMM 通过 VMCS 来提供，并存储在处理器控制寄存器中，供此微代码随后使用。虽然本发明的范围在这方面不受限制，但是，在一个实施例中，可以提供多个有效偏移，

用于读取仿真，并可以提供多个有效偏移，用于写入仿真。在一个特定实施例中，可以提供大致 40 个有效偏移，用于读取仿真，提供大致 20 个有效偏移，用于写入仿真，虽然本发明的范围在这方面不受限制。

[0033] 如果访问不指向这些仿真的偏移中的一个，例如，基于比特向量确定的，控制进入框 290，在那里，可以执行从虚拟机环境到 VMM 的退出。更具体而言，可以发生 VM 退出，如此，将控制传递到 VMM，带有错误是由于 APIC 访问失败造成的指示。响应于这样的指示，VMM 可以采取合适的动作，诸如结束客户机或执行仿真的设备特定的动作，诸如记录错误等等。

[0034] 仍参考图 3，如果相反访问指向有效仿真的偏移，则控制进入框 270，在那里，各种信息可以被编程为允许对虚拟 APIC 页面的访问发生（如上文参考存储器集群逻辑所描述的）。在一个实施例中，此编程可以是针对上文所描述的控制寄存器的。更具体而言，此控制寄存器可以被利用关于仿真的读和写的许可以及仿真的寄存器偏移编程，如此，指出对此仿真的偏移的访问将被允许。

[0035] 控制接下来进入框 280，在那里，可以重新启动辅助指令。更具体而言，微代码可以向处理器的前端单元发出指令，以导致访问请求指令（例如，加载 / 存储）被重新获取和重新执行。注意，有可能出现这样的情况 :此指令仍可能存在于处理器的缓存结构中，诸如指令缓存。如果是，则可以以最小的延迟获取指令，并传递，以供执行。否则，如果由于种种原因，诸如容量清除，指令被清除，则可以再次从存储器层次结构中获取指令。在任何情况下，方法 250 都结束，且控制回到此访问指令的进一步的操作。例如，可以对于重新获取的指令，再次执行图 2 的操作。然而，此时，（假设获取了相同指令），菱形 220 和 230 中的判断应该是肯定的，如此，可以在框 240 执行重新映射。尽管在图 3 的实施例以这样高级别地表示，然而要理解本发明的范围不限于此方面。

[0036] 图 4 示出了虚拟机环境 300 的一个实施例。在此实施例中，裸平台硬件 310 是计算平台，诸如给定服务器、台式机、膝上型计算机、Ultrabook™、平板电脑、智能电话等等，以执行 OS 或诸如 VMM 325 之类的 VMM。平台硬件 310 包括至少一个处理器 312、存储器 320 和可能其他平台硬件（例如，输入输出设备），未示出。

[0037] 处理器 312 可以是能够执行软件的任何类型的处理器，诸如微处理器、数字信号处理器、微控制器，等等。处理器 312 可以包括根据本发明的各实施例的用于执行处理器组件的虚拟化的微代码、可编程逻辑或硬编码的逻辑。为此，可以存在访问逻辑 330，以控制对诸如 APIC（为便于说明，未示出）之类的虚拟化的资源的访问。

[0038] 存储器 320 可以是硬盘、随机存取存储器 (RAM)、只读存储器 (ROM)、闪存，或其他非易失性存储器，或上述设备的任何组合，或可由处理器 312 读取的任何其他类型的非瞬时的机器可读的存储介质。存储器 320 可以存储用于执行本发明的方法实施例的执行的指令或数据。可以看出，存储器 320 包括一个或多个 VMCS 322，VMCS 322 包括一个或多个设备访问页面，以及对应的虚拟设备页面的地址，如此处所描述的。

[0039] 每一个 VMM 325，尽管通常是以软件实现的，可以仿真并输出到较高级别的软件的裸机接口。这样的较高级别的软件可以是标准或实时 OS。VMM 325 可以，例如，以硬件、软件、固件，或通过各种技术的组合来实现。

[0040] 当运行时，每一个 VMM 325 都向客户机软件 (VMM 325 的软件之外的软件）呈现一个或多个 VM 的抽象。VMM 325 可以向各种客户机提供相同或不同的抽象。在每一个 VM 上

运行的客户机软件都可以包括客户机 OS(例如,客户机 OS 334、344 或 354)和各种客户机软件应用(例如,应用 336、346 和 356)。共同地,客户机 OS 和软件应用此处被称为客户机软件 303、305 和 315。

[0041] 客户机软件 303、305 和 315 预期访问 VM 332、342 和 352 内的客户机软件在其上面正在运行的物理资源(例如,处理器寄存器、存储器和 I/O 设备)。VMM 325 促进对客户机软件所希望的资源的访问而同时保持对平台硬件 310 内的资源的最后控制。另外,客户机软件 303、305 和 315 还预期处理各种事件,诸如异常、中断和平台事件。

[0042] 各实施例可以以许多不同的系统来实现。例如,实施例可在诸如多核处理器之类的处理器中实现。现在参照图 5,图 5 示出了根据本发明一个实施例的处理器核的框图。如图 5 所示,处理器核 400 可以是多核处理器中的一个核,并且被示为多级流水线化的无序处理器。在图 5 中利用相对简化的视图示出处理器核 400,以示出与根据本发明的实施例的处理器资源的虚拟化结合使用的各种特征。

[0043] 如图 5 所示,核 400 包括前端单元 410,前端单元 410 可用于取得将被执行的指令并将这些指令准备好以供以后在处理器中使用。例如,前端单元 410 可包括获取单元 401、指令缓存 403 和指令解码器 405。在某些实施例中,指令缓存可以维护包括性,保留条目,直到对应的指令被隐退。在其他实施例中,指令缓存可以是非包括性的缓存,以便对应于核中的挂起的指令的条目可以在指令的隐退之前被清除。在某些实现中,前端单元 410 可进一步包括跟踪缓存、微码存储以及微操作存储。获取单元 401 可(例如,从存储器或指令缓存 403)获取宏指令并将它们馈送至指令解码器 405 以将它们解码为原语,即用于通过处理器执行的微操作。还存在微代码存储 407 以存储根据本发明的一个实施例的辅助代码。

[0044] 无序 (OOO) 引擎 415 耦合在前端单元 410 与执行单元 420 之间,无序引擎 415 可用于接收微指令并将它们准备好以供执行。更具体地,OOO 引擎 415 可包括多个缓冲器,用于对微指令流重新排序并分配执行所需的各种资源,以及例如通过使用该引擎的重命名逻辑来提供逻辑寄存器到各个寄存器组(诸如寄存器组 430 和扩展寄存器组 435)内的存储位置上的重命名。寄存器组 430 可包括用于整数和浮点操作的单独的寄存器组。扩展寄存器组 435 可提供向量尺寸单元的存储,例如,每寄存器 256 或 512 位。可以存在一组控制寄存器 436,包括存储与诸如 APIC 之类的处理器资源的可编程虚拟化相关联的信息的寄存器 438,如此处所描述的。

[0045] 在执行单元 420 中可存在多种资源,包括例如多种整数、浮点和单指令多数据 (SIMD) 逻辑单元等其它专门硬件。例如,此类执行单元可包括一个或多个算术逻辑单元 (ALU)422。当然,还可存在诸如乘法累加单元之类的其他执行单元等等。结果可以被提供给隐退逻辑,该隐退逻辑可以实现在处理器的存储器子系统 460 内。例如包括执行单元和前端逻辑的各个处理器结构可以耦合到存储器子系统 460。该存储器子系统可提供处理器结构与存储器层次的其他部分(例如芯片上或芯片外的缓存和系统存储器)之间的接口。如所看见那样,各子系统具有各个组件,包括存储器排序缓冲器 (MOB)440。更具体地,MOB 440 可包括各种阵列和逻辑以接收与被执行的指令相关联的信息。然后,通过 MOB 440 检查该信息以确定指令是否可以有效隐退并且结果数据是否被提交至处理器的架构状态,或阻止指令的正常隐退的一个或多个异常是否发生。当然,MOB 440 可处理与隐退相关联的其他操作。

[0046] 如图5所示，MOB 440耦合至缓存450，在一个实施例中，该缓存450可以是低级缓存（例如L1缓存）。存储器子系统460还可包括集成存储器控制器470以提供与系统存储器（在图5中为了说明方便而未示出）的通信。存储器子系统460还可包括存储器执行单元475，该存储器执行单元475处理各种操作，以发起存储器请求并且处理数据从存储器的返回。例如，如图5的实施例所示，MEU 475可以包括页漏失处理程序476，以当发生转换后援缓冲器漏失时获取从逻辑地址到物理地址的转换。MEU的访问逻辑477可以执行访问检查，以判断访问请求是否将指向仿真的区域，如果是，则将请求重新映射到重定向区域（例如，APIC访问页面到vAPIC页面重新映射）。还可以存在各种存储器请求之间仲裁的仲裁器479。进一步，虽然未示出，但应理解MEU中可存在诸如缓冲器、调度器等等其他结构。

[0047] 从存储器子系统460，可发生与更高级缓存、系统存储器等等的数据通信。虽然在图5的实施例中用高级框图示出，但应理解本发明的范围不限于此方面。例如，虽然图5的实现方式是参照无序机、比如所谓的x86指令集架构(ISA)的无序机的，但本发明的范围在此方面不受限制。即，其他实施例可在以下处理器中实现：有序处理器；诸如基于ARM的处理器之类的精简指令集计算(RISC)处理器；或具有另一类型ISA的处理器，该另一类型的ISA可经由仿真引擎和相关联的逻辑电路来仿真不同ISA的指令和操作。

[0048] 即，在其他实施例中，处理器架构可包括仿真特征，使得处理器可执行称为源ISA的第一ISA的指令，其中该架构是根据第二ISA的，该第二ISA被称为目标ISA。一般而言，包括OS和应用程序的软件被编译至源ISA，而硬件实现针对给定硬件实现方式而特别设计的具有特殊性能和/或能源效率特征的目标ISA。

[0049] 现在参考图6，所示是根据本发明的实施例的处理器的框图。如图6所示，处理器500可以是包括核域510中的多个核510a-510n的多核处理器。在一个实施例中，每个这样的核都可以是独立功率域并可被配置成在独立电压和/或频率下工作，并当存在可用净空时进入超频模式，或可以作为单独的域，均匀地控制核。各种核都可以通过互连515而耦合到包括各种组件的系统代理或非核520。如所见那样，非核520可包括共享的缓存存储器530，它可以是最末级缓存。另外，非核还可以包括集成的存储器控制器540、各种接口550和功率控制单元555，以控制处理器的组件的功率消耗。可以如此处所描述的核的存储器集群的访问逻辑，在核上执行的多个VM之间虚拟化APIC 545。

[0050] 进一步参见图6，处理器500可经由例如存储器总线与系统存储器560通信。另外，通过接口550可对诸如外围设备、海量存储器等多种芯片外组件作出连接。虽然在图6的实施例中示出具有该特定实现，但本发明的范围不限于此方面。

[0051] 参考图7，示出了包括多个核的处理器的实施例。处理器1100包括任何处理器或处理器件，诸如微处理器、嵌入式处理器、数字信号处理器(DSP)、网络处理器、手持式处理器、应用处理器、协同处理器、片上系统(SOC)、或用于执行代码的其它器件。在一个实施例中，处理器1100包括至少两个核——核1101和1102，它们可包括非对称核或对称核（所示实施例）。然而，处理器1100可包括可以是对称的或非对称的任何数量的处理元件。

[0052] 在一个实施例中，处理元件指的是用于支持软件线程的硬件或逻辑。硬件处理元件的示例包括：线程单元、线程槽、线程、进程单元、上下文、上下文单元、逻辑处理器、硬件线程、核、和/或能保持处理器的诸如执行状态或架构状态之类的状态的任何其它元件。换言之，在一个实施例中，处理元件指的是能够与诸如软件线程、操作系统、应用、或其它代码

之类的代码独立地相关联的任何硬件。物理处理器通常指的是集成电路，其可能包括任意数量的诸如核或硬件线程之类的其它处理元件。

[0053] 核通常指的是位于集成电路上的能够维持独立架构状态的逻辑，其中每个独立维持的架构状态与至少某些专用执行资源相关联。与核相反，硬件线程通常指的是位于集成电路上的能维持独立架构状态的任何逻辑，其中独立维持的架构状态共享对执行资源的访问。可以看出，当某些资源是共享的而其它资源是架构状态专用时，硬件线程与核的术语之间的界线交叠。核和硬件线程常常被操作系统视为单个逻辑处理器，其中，操作系统能够各个地在每一个逻辑处理器上调度操作。

[0054] 如图 7 所示的物理处理器 1100 包括两个核——核 1101 和 1102。在此，核 1101 和 1102 被视为对称核，即这些核具有相同的配置、功能单元和 / 或逻辑。在另一个实施例中，核 1101 包括无序处理器核，而核 1102 包括有序处理器核。然而，核 1101 和 1102 可从任何类型的核中单独地选择，诸如原生核、受软件管理的核、适于执行原生指令集架构 (ISA) 的核、适于执行转换 ISA 的核、协同设计的核或其它已知核。不过，为进一步讨论，以下将进一步详细描述在核 1101 中示出的功能单元，因为核 1102 中的单元以类似方式操作。

[0055] 如所描绘的，核 1101 包括两个硬件线程 1101a 以及 1101b，它们也可以被称为硬件线程槽 1101a 以及 1101b。因此，在一个实施例中，诸如操作系统之类的软件实体潜在地将处理器 1100 视为四个独立的处理器，即能够并发地执行四个软件线程的四个逻辑处理器或处理元件。如上所述，第一线程与架构状态寄存器 1101a 相关联，第二线程与架构状态寄存器 1101b 相关联，第三线程可与架构状态寄存器 1102a 相关联，并且第四线程可与架构状态寄存器 1102b 相关联。在此，架构状态寄存器 (1101a、1101b、1102a 和 1102b) 中的每一个可被称为处理元件、线程槽或线程单元，如上所述。如图所示，架构状态寄存器 1101a 在架构状态寄存器 1101b 中复制，如此，单个架构状态 / 上下文能够为逻辑处理器 1101a 和逻辑处理器 1101b 存储。在核 1101 中，其他较小资源，诸如分配器和重新命名块 1130 中的指令指针和重新命名逻辑也可以为线程 1101a 和 1101b 复制。诸如重排序 / 隐退单元 1135 中的重排序缓冲器、ILTB1120、加载 / 存储缓冲器，以及队列之类的某些资源可以通过分区来共享。诸如通用内部寄存器、页表基本寄存器、低级数据缓存和数据 TLB 1115、执行单元 1140、以及无序单元 1135 的部分之类的其它资源潜在地被完全共享。

[0056] 处理器 1100 通常包括其它资源，这些其它资源可被完全共享、通过分区共享、或由处理元件专用 / 专用于处理元件。在图 7 中，示出了具有处理器的说明性的逻辑单元 / 资源的纯示例性处理器的实施例。注意，处理器可包括或省略这些功能单元中的任一个，并包括未描绘出的任何其它已知的功能单元、逻辑或固件。如图所示，核 1101 包括简化的、代表性的无序 (000) 处理器核。但是，在不同实施例中可利用有序处理器。000 核包括用于预测要被执行 / 进行的分支的分支目标缓冲器 1120 以及用于存储指令的地址转换条目的指令转换缓冲器 (I-TLB) 1120。

[0057] 核 1101 进一步包括耦合至获取单元 1120 以用于解码所获取的元素的解码模块 1125。在一个实施例中，获取逻辑包括分别与线程槽 1101a、1101b 相关联的单独定序器。通常，核 1101 与第一 ISA 相关联，该第一 ISA 定义 / 指定能在处理器 1100 上执行的指令。是第一 ISA 的一部分的机器代码指令常常包括引用 / 指定要被执行的指令或操作的指令的一部分 (被称为操作码)。解码逻辑 1125 包括由这些指令的操作码来识别这些指令并在流

水线上传递所解码的指令以进行如第一 ISA 所定义的处理的电路。例如，在一个实施例中，解码器 1125 包括被设计成或适于识别诸如事务性指令之类的特定指令的逻辑。作为解码器 1125 识别的结果，架构或核 1101 采取特定的、预定的动作以执行与适当指令相关联的任务。重要的是应注意，本申请中描述的任务、块、操作和方法中的任一个可响应于单个或多个指令来执行；它们中的一些可以是新的或旧的指令。

[0058] 在一个示例中，分配器和重命名器块 1130 包括用于预留资源的分配器，诸如用于存储指令处理结果的寄存器组。然而，线程 1101a 和 1101b 潜在地能够进行无序执行，其中分配器和重命名器块 1130 还保留其它资源（诸如用于跟踪指令结果的重排序缓冲器）。单元 1130 还可包括寄存器重命名器，用于将程序 / 指令引用寄存器重命名为处理器 1100 内部的其它寄存器。重排序 / 隐退单元 1135 包括诸如上述的重排序缓冲器、加载缓冲器和存储缓冲器之类的组件，以支持无序执行的指令的无序执行和稍后的有序隐退。

[0059] 在一个实施例中，调度器和执行单元块 1140 包括调度器单元，用于在执行单元上调度指令 / 操作。例如，在具有可用的浮点执行单元的执行单元的端口上调度浮点指令。还包括与执行单元相关联的寄存器组，用于存储信息指令处理结果。示例性的执行单元包括浮点执行单元、整数执行单元、跳转执行单元、加载执行单元、存储执行单元以及其它已知的执行单元。

[0060] 较低级的数据缓存和数据转换缓冲器 (D-TLB) 1150 耦合至执行单元 1140。数据缓存用于存储最近使用 / 操作的元素（诸如数据操作数），这些元素在存储器一致性状态下潜在地被保持。D-TLB 用于存储最近的虚拟 / 线性至物理地址转换。作为特定示例，处理器可包括页表结构，用于将物理存储器分解成多个虚拟页。

[0061] 在此，核 1101 和 1102 共享对较高级或进一步远离的缓存 1110 的访问，较高级或进一步远离的缓存用于缓存最近获取的元素。注意，较高级或进一步指的是缓存级增加或进一步远离执行单元。在一个实施例中，较高级缓存 1110 是最后级数据缓存——处理器 1100 上的存储器层次中的最后级缓存，诸如第二或第三级数据缓存。然而，较高级缓存 1110 不限于此，因为它可与指令缓存相关联或包括指令缓存。替代地，跟踪缓存——一种类型的指令缓存——可耦合在解码器 1125 之后，用于存储最近解码的跟踪。

[0062] 在所描绘的配置中，根据本发明的一个实施例，处理器 1100 还包括总线接口模块 1105 和可以执行功率共享控制的功率控制器 1160。历史上，控制器 1170 被包括在处理器 1100 外部的计算系统中。在该场景中，总线接口 1105 与处理器 1100 外部的设备通信，处理器 100 外部的设备诸如系统存储器 1175、芯片组（通常包括存储器控制器中枢以连接到存储器 1175 以及 I/O 控制器中枢以连接到外围设备）、存储器控制器中枢、北桥、或其它集成电路。并且在该场景中，总线 1105 可包括任何已知的互连，诸如多点总线、点对点互连、串行互连、并行总线、一致性（例如缓存一致性）总线、分层协议架构、差分总线以及 GTL 总线。

[0063] 存储器 1175 可专属于处理器 1100 或与系统中的其它器件共享。存储器 1175 的类型的常见示例包括 DRAM、SRAM、非易失性存储器 (NV 存储器）以及其它已知的存储设备。注意，器件 1180 可包括耦合到存储器控制器中枢的图形加速器、处理器或卡，耦合到 I/O 控制器中枢的数据存储，无线收发器，闪存器件，音频控制器，网络控制器，或其它已知器件。

[0064] 然而，注意，在所描绘的实施例中，控制器 1170 被示为处理器 1100 的一部分。最

近，随着更多的逻辑和器件被集成在单个管芯上（如 SOC），这些器件中的每一个可被合并在处理器 1100 上。例如，在一个实施例中，存储器控制器中枢 1170 与处理器 1100 处于同一封装和 / 或管芯上。在此，核的一部分（核上部分，on-core portion）包括与诸如存储器 1175 和 / 或图形器件 1180 之类的其它器件进行接口的一个或多个控制器 1170。包括用于与此类器件进行接口的控制器和互连的该配置通常被称为核上（或非核（un-core）配置）。作为示例，总线接口 1105 包括环形互连，环形互连具有用于与存储器 1175 进行接口的存储器控制器以及用于与图形处理器 1180 进行接口的图形控制器。然而，在 SOC 环境中，诸如网络接口、协同处理器、存储器 1175、图形处理器 1180 以及任何其它已知计算机器件 / 接口之类的甚至更多的器件可被集成到单个管芯或集成电路上，以提供具有高功能性和低功耗的小外形规格。

[0065] 实施例可在许多不同的系统类型中实现。现在参照图 8，其中示出了根据本发明一实施例的系统的框图。如图 8 所示，多处理器系统 600 是点对点互连系统，并包括通过点对点互连 650 而耦合的第一处理器 670 和第二处理器 680。如图 8 所示，处理器 670 和 680 中的每一个都可以是多核处理器，包括第一和第二处理器核（即，处理器核 674a 和 674b 以及处理器核 684a 和 684b），虽然潜在地更多核可以存在于处理器中。处理器中的每一个都可包括虚拟化逻辑以允许多个客户机访问存储器映射的处理器资源，诸如 APIC，如此处所描述的。

[0066] 仍参考图 8，第一处理器 670 还包括存储器控制器中枢 (MCH)672 和点对点 (P-P) 接口 676 和 678。类似地，第二处理器 680 包括 MCH 682 和 P-P 接口 686 和 688。如图 8 所示，MCH 672 和 682 将处理器耦合到相应的存储器，即，存储器 632 和存储器 634，它们可以是本地连接到相应的处理器的系统存储器（例如，DRAM）的一部分。第一处理器 670 和第二处理器 680 可以分别通过 P-P 互连 652 和 654 来耦合到芯片集 690。如图 8 所示，芯片集 690 包括 P-P 接口 694 和 698。

[0067] 此外，芯片集 690 还包括接口 692，接口 692 通过 P-P 互连 639 而将芯片集 690 与高性能图形引擎 638 耦合。芯片集 690 又可以通过接口 696 耦合到第一总线 616。如图 8 所示，各种输入 / 输出 (I/O) 设备 614 以及总线桥接器 618 可以耦合到第一总线 616，总线桥接器 618 将第一总线 616 耦合到第二总线 620。在一个实施例中，各种设备可以耦合到第二总线 620，包括，例如，键盘 / 鼠标 622、通信设备 626 和数据存储单元 628，数据存储单元 628 诸如磁盘驱动器或可以包括代码 630 的其他大容量存储设备。进一步地，音频 I/O 624 可以耦合到第二总线 620。各实施例可以被包括到其他类型的系统中，包括诸如智能蜂窝电话、Ultrabook™、平板电脑、上网本等等之类的移动设备。

[0068] 各实施例可以以代码来实现，并可以存储在在其上存储了指令的非瞬时的存储介质上，指令可以被用来对系统进行编程以执行指令。存储介质可以包括，但不仅限于，任何类型的磁盘，包括软盘、光盘、固态驱动器 (SSD)、光盘只读存储器 (CD-ROM)、光盘可重写 (CD-RW)，以及磁光盘、诸如只读存储器 (ROM) 之类的半导体器件、诸如动态随机存取存储器 (DRAM)、和静态随机存取存储器 (SRAM) 之类的随机访问存储器 (RAM)、可擦除编程只读存储器 (EPROM)、闪存、电可擦除编程只读存储器 (EEPROM)、磁卡或光卡，或适于存储电子指令的任何其他类型的介质。

[0069] 尽管是参考数量有限的实施例来描述本发明的，但是，那些精通本技术的人将从

其中理解很多修改和变体。所附权利要求书涵盖所有这样的修改和变体都将在本发明的真正的精神和范围内。

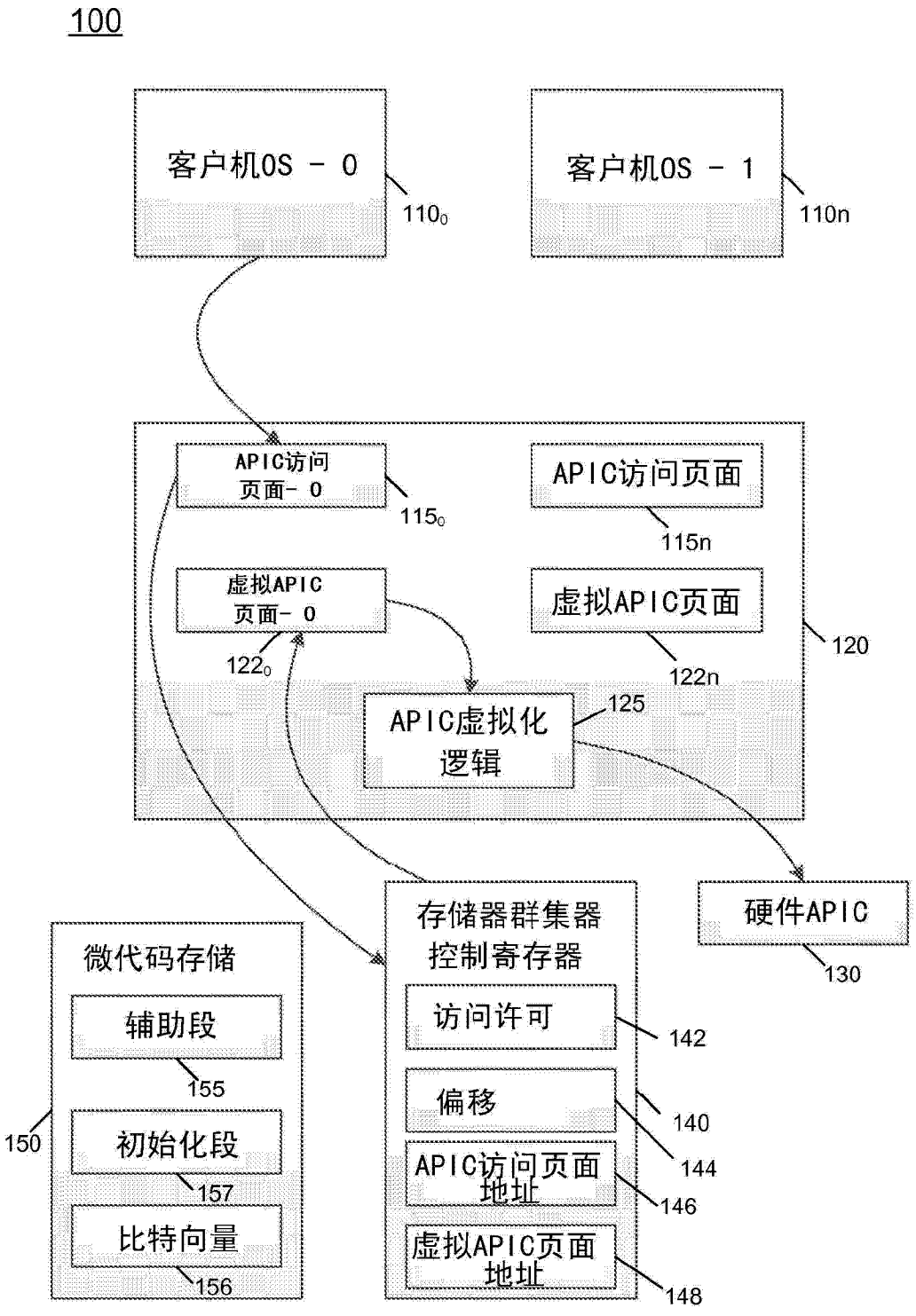

100
客户机OS - 0
1100
客户机OS - 1
110n
APIC访问
页面- 0
1150
APIC访问页面
115n
虚拟APIC
页面- 0
1220
虚拟APIC页面
122n
120
APIC虚拟化
逻辑
125
硬件APIC
130
存储器群集器
控制寄存器
访问许可
142
偏移
140
144
APIC访问页面
地址
146
虚拟APIC页面
地址
148
微代码存储
辅助段
155
150
初始化段
157
比特向量
156

图 1

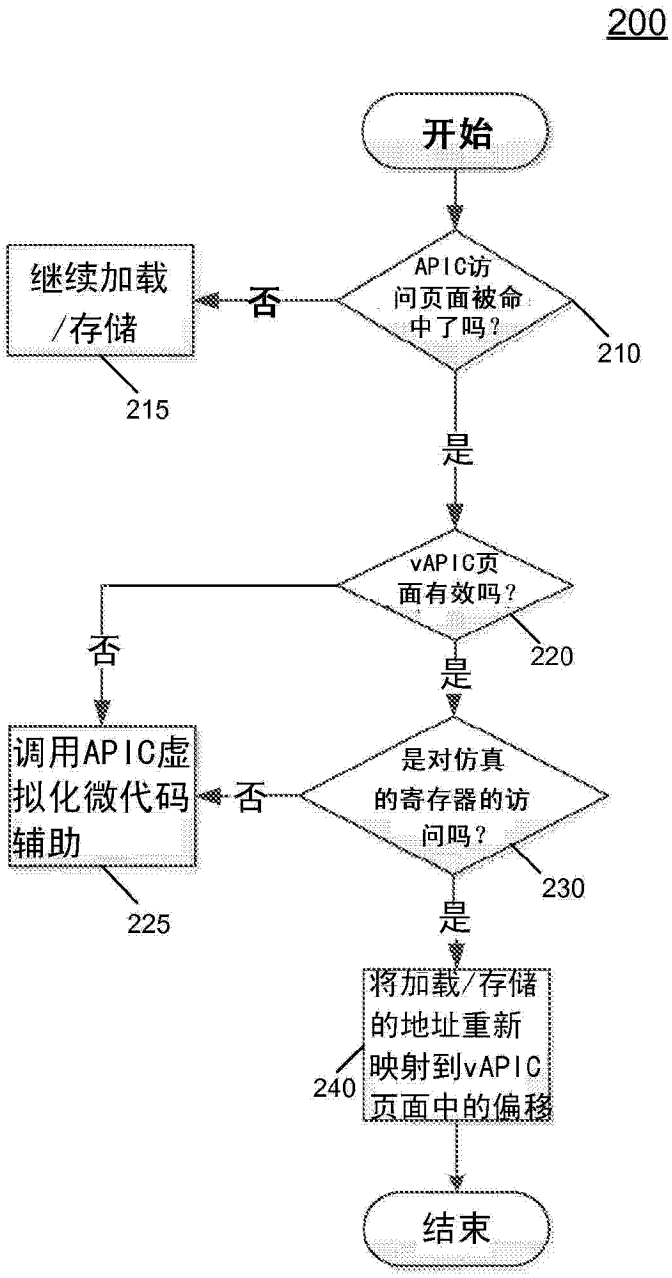

200
开始
APIC访问页面被命中了吗？
210
否
继续加载/存储
215
是
vAPIC页面有效吗？
220
否
是
调用APIC虚拟化微代码辅助
225
否
是对仿真的寄存器的访问吗？
230
是
将加载/存储的地址重新映射到vAPIC页面中的偏移
240
结束

图 2

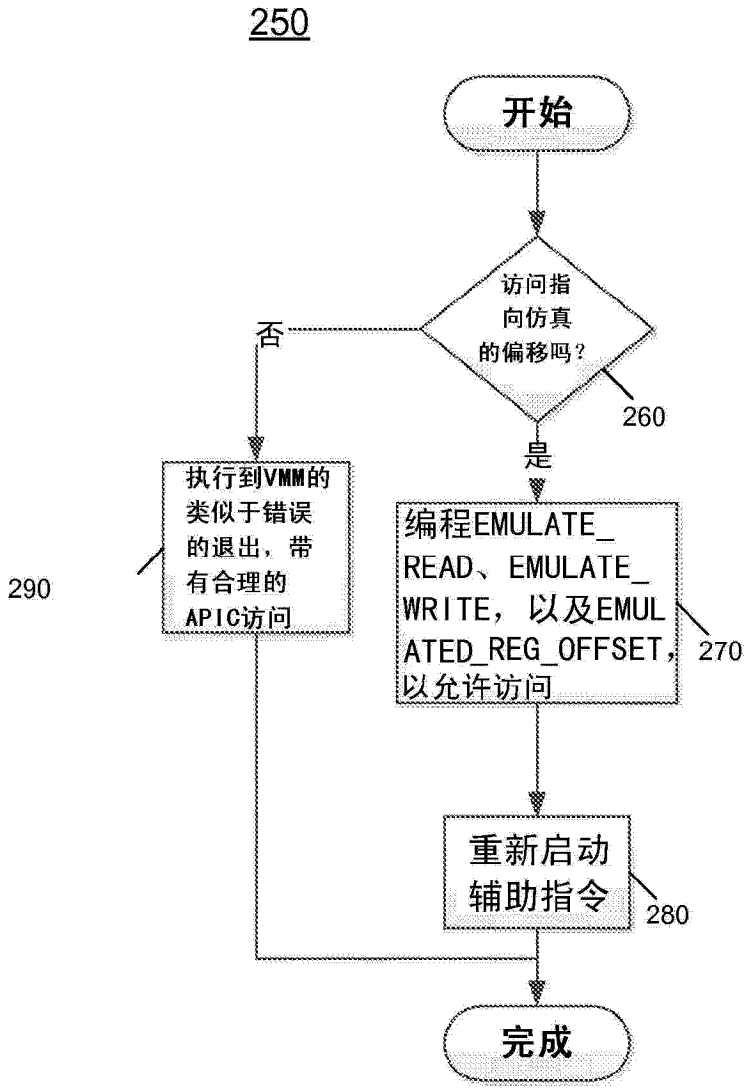

250
开始
访问指向仿真的偏移吗？
260
否
是
执行到VMM的类似于错误的退出，带有合理的APIC访问
290
编程EMULATE_READ、EMULATE_WRITE，以及EMULATED_REG_OFFSET，以允许访问
270
重新启动辅助指令
280
完成

图 3

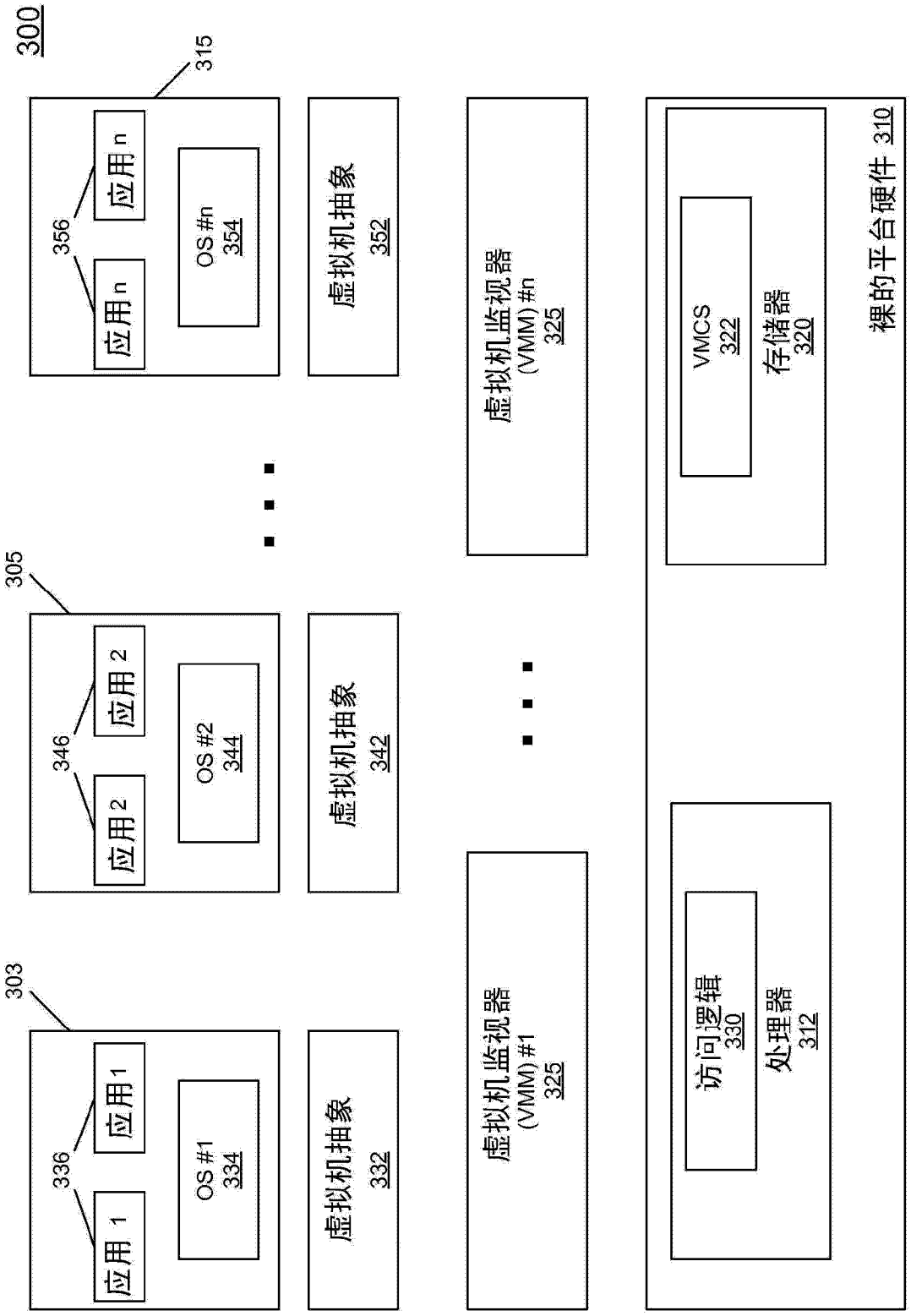

300
315
356
应用 n
应用 n
OS #n
354
虚拟机抽象
352
虚拟机监视器
(VMM) #n
325
305
346
应用 2
应用 2
OS #2
344
虚拟机抽象
342
303
336
应用 1
应用 1
OS #1
334
虚拟机抽象
332
虚拟机监视器
(VMM) #1
325
VMCS
322
存储器
320
访问逻辑
330
处理器
312
裸的平台硬件 310

图 4

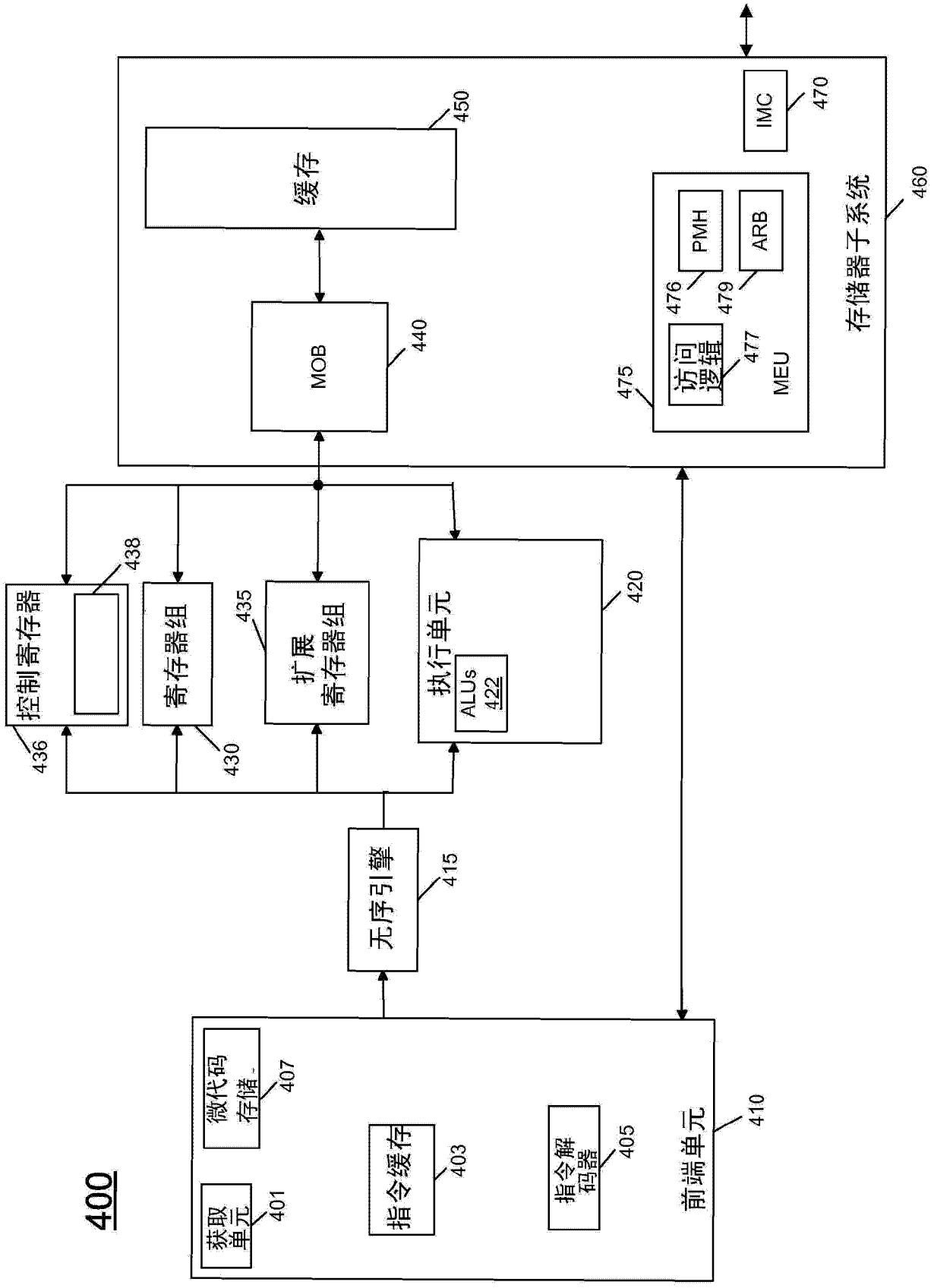

400
前端单元
410
获取单元
401
微代码存储
407
指令缓存
403
指令解码器
405
无序引擎
415
控制寄存器
438
436
寄存器组
430
扩展寄存器组
435
执行单元
420
ALUs
422
MOB
440
缓存
450
存储器子系统
460
IMC
470
PMH
476
ARB
479
访问逻辑
477
MEU
475

图 5

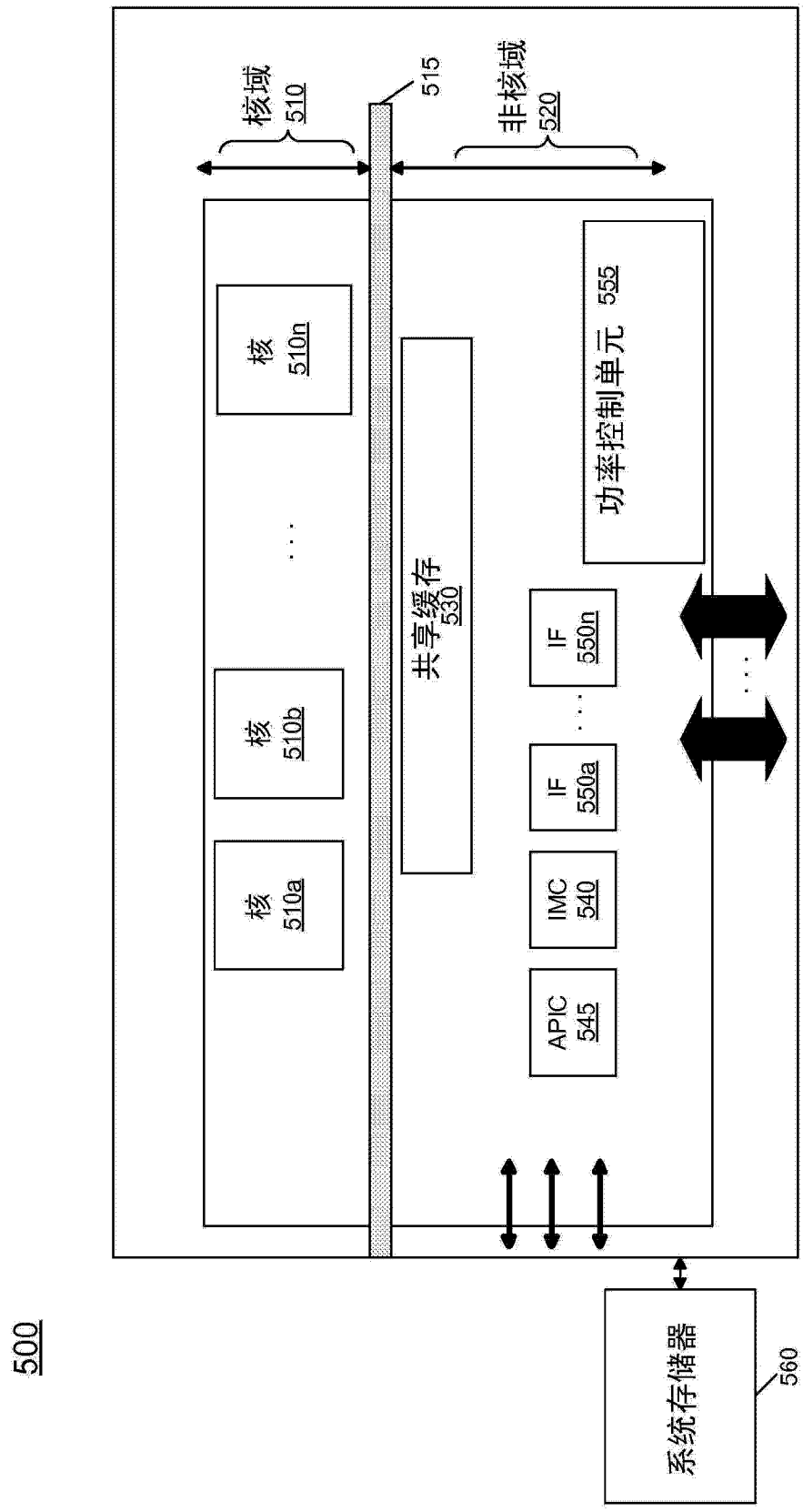

500
核域
510
515
非核域
520
核
510n
核
510b
核
510a
共享缓存
530
功率控制单元
555
IF
550n
IF
550a
IMC
540
APIC
545
系统存储器
560

图 6

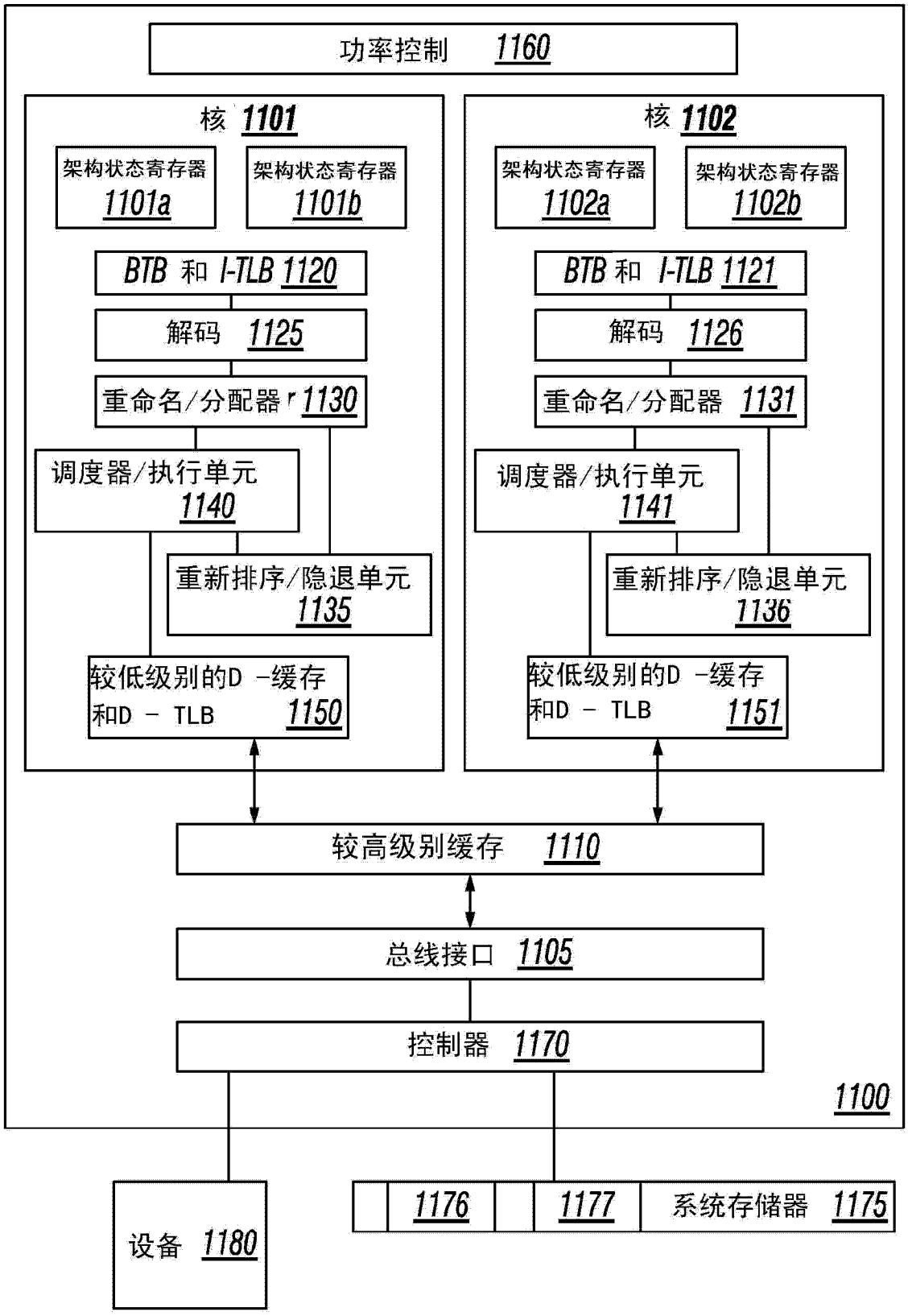

功率控制 1160
核 1101
架构状态寄存器 1101a
架构状态寄存器 1101b
BTB 和 I-TLB 1120
解码 1125
重命名/分配器 1130
调度器/执行单元 1140
重新排序/隐退单元 1135
较低级别的D -缓存和D - TLB 1150
核 1102
架构状态寄存器 1102a
架构状态寄存器 1102b
BTB 和 I-TLB 1121
解码 1126
重命名/分配器 1131
调度器/执行单元 1141
重新排序/隐退单元 1136
较低级别的D -缓存和D - TLB 1151
较高级别缓存 1110
总线接口 1105
控制器 1170
1100
设备 1180
1176
1177
系统存储器 1175

图 7

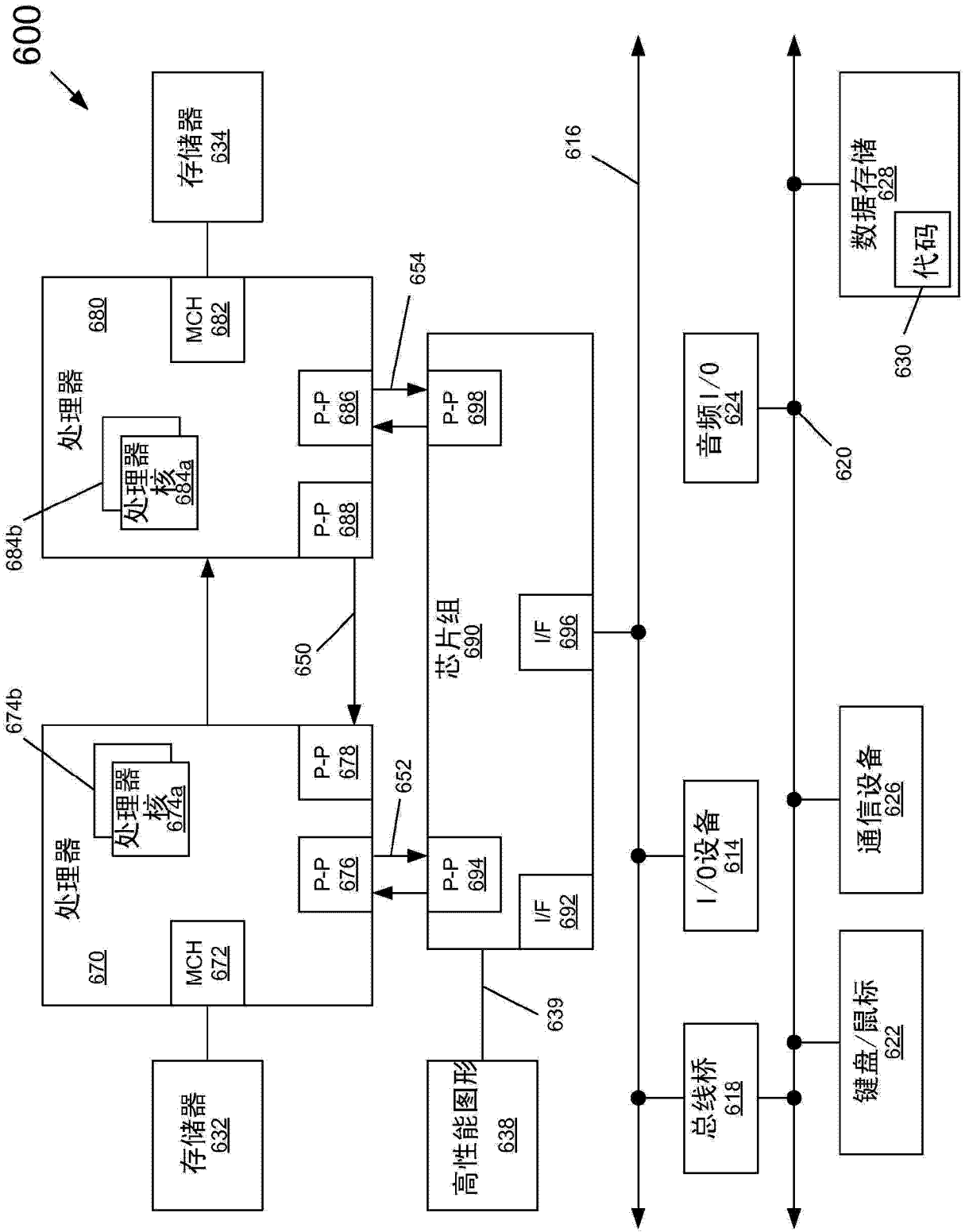

600
存储器 634
MCH 682
680
处理器
处理器核 684a
684b
P-P 686
P-P 688
654
650
P-P 698
芯片组 690
I/F 696
616
数据存储 628
代码
630
音频I/O 624
620
674b
处理器
处理器核 674a
670
P-P 678
P-P 676
652
P-P 694
I/F 692
MCH 672
I/O设备 614
通信设备 626
639
存储器 632
高性能图形 638
总线桥 618
键盘/鼠标 622

图 8