[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/46 (2006.01)
G06F 13/24 (2006.01)

[12] 发明专利说明书

专利号 ZL 200580023031.8

[45] 授权公告日 2009 年 12 月 30 日

[11] 授权公告号 CN 100576175C

[22] 申请日 2005.7.1
[21] 申请号 200580023031.8
[30] 优先权
[32] 2004. 7. 6 [33] US [31] 60/586,486
[32] 2005. 6.29 [33] US [31] 11/169,542
[86] 国际申请 PCT/US2005/023525 2005.7.1
[87] 国际公布 WO2006/014354 英 2006.2.9
[85] 进入国家阶段日期 2007.1.8
[73] 专利权人 茵姆拜迪欧有限公司
地址 美国加利福尼亚州
[72] 发明人 拉吉夫·S·德赛
辛格·拉吉帕特·贾斯温德尔
[56] 参考文献
US6631394B1 2003.10.7
CN1116881A 1996.2.14
CN1409209A 2003.4.9
US5903752A 1999.5.11
审查员 吉张媛
[74] 专利代理机构 北京三友知识产权代理有限公司
代理人 黄纶伟 迟 军

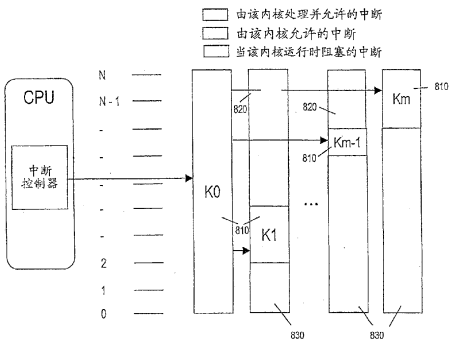
权利要求书 3 页 说明书 21 页 附图 12 页

[54] 发明名称

用于多个内核的并行执行的方法和系统

[57] 摘要

本发明公开了一种用于多个内核的并行执行的方法和系统。 提供了一种用于使用公共中断处理器和可选的公共调度器来并行地运行多个内核的方法。 还提供了在所述多个内核之间对执行进行切换的技术。 示出了使用中断屏蔽级的在所述多个内核之间的执行和中断占先。 还提供了用于在不同的内核上运行的任务之间共享资源的技术。



1、一种用于控制嵌入式系统的操作的方法，所述方法包括：

所述嵌入式系统的处理器启动主内核以控制所述嵌入式系统的操作；

所述处理器启动至少一个二级内核来辅助所述主内核控制所述嵌入式系统的操作，扩展所述嵌入式系统的功能，所述至少一个二级内核在所述主内核的至少部分控制之下；以及

所述处理器将一调度器操作为公共调度器，该公共调度器对所述主内核和所述至少一个二级内核的处理的执行进行调度。

2、根据权利要求1所述的方法，其中，所述主内核是通用操作系统。

3、根据权利要求1所述的方法，其中，所述至少一个二级内核中的至少一个是实时操作系统。

4、根据权利要求1所述的方法，其中，所述主内核比各个所述至少一个二级内核更有能力。

5、根据权利要求1所述的方法，该方法还包括：所述处理器将一中断处理器操作为公共中断处理器，所述公共中断处理器处理所述主内核和所述至少一个二级内核的中断。

6、根据权利要求1所述的方法，其中，该方法还包括：所述处理器在所述主内核的调度器和所述至少一个二级内核的一个或多个调度器之间选择一个更有能力的调度器来作为所述公共调度器。

7、根据权利要求5所述的方法，该方法还包括：所述处理器在所述主内核的中断处理器和所述至少一个二级内核的一个或多个中断处理器之间选择一个更有能力的中断处理器来作为所述公共中断处理器。

8、根据权利要求7所述的方法，其中所述选择步骤包括：所述处理器选择所述主内核的中断处理器作为所述公共中断处理器或选择所述主内核的调度器作为所述公共调度器。

9、根据权利要求1所述的方法，其中所述处理器在引导所述嵌入式系统时，在启动所述至少一个二级内核之前启动所述主内核。

10、根据权利要求1所述的方法，其中所述处理器启动所述至少一

个二级内核中的至少一个，作为所述主内核的运行时间动态模块。

11、根据权利要求 5 所述的方法，该方法还包括：

所述处理器将唯一的内核标识分配给所述主内核；以及

将至少一个中断屏蔽级分配给所述主内核，所述至少一个中断屏蔽级确定所述主内核所允许的中断。

12、根据权利要求 5 所述的方法，还包括：

所述处理器将唯一的内核标识分配给所述至少一个二级内核；以及

所述处理器将至少一个中断屏蔽级分配给所述至少一个二级内核，所述至少一个中断屏蔽级确定所述至少一个二级内核所允许的中断。

13、根据权利要求 5 所述的方法，还包括：所述处理器在所述公共调度器或所述公共中断处理器中安装用于所述至少一个二级内核中的至少一个的连接程序。

14、根据权利要求 1 所述的方法，该方法还包括：所述处理器执行将处理执行控制从当前有效内核切换到下一有效内核的任务，其中，所述下一有效内核是所述至少一个二级内核中的一个。

15、根据权利要求 14 所述的方法，其中，所述处理执行控制任务是周期任务，并且所述方法还包括：当操作所述主内核时，所述处理器通过根据二级内核轮询优先级方案进行轮询来将所述至少一个二级内核之一确定为所述下一有效内核，具有最高优先级的所述下一有效二级内核具有至少一个要执行的未决处理，并且在完成所述具有最高优先级的下一有效二级内核的所述至少一个要执行的未决处理的至少一部分之后，将处理执行控制转移回所述主内核。

16、根据权利要求 12 所述的方法，该方法还包括：所述处理器调用所述公共中断处理器，之后所述公共中断处理器执行至少一个内核独立中断处理功能，并随后将处理执行控制传递到与该中断相关联的目标内核的中断服务例程。

17、根据权利要求 16 所述的方法，其中，由所述至少一个二级内核的屏蔽级来确定所述目标内核。

18、根据权利要求 4 所述的方法，该方法还包括：

所述处理器调用所述公共调度器；

所述处理器确定哪个内核是当前正在执行的内核；

所述处理器将处理执行控制转移到所述当前正在执行的内核；以及

所述处理器执行当前执行的内核的至少一个内核特定调度功能。

19、根据权利要求 12 所述的方法，该方法还包括：

所述主内核将处理执行控制传递到所述至少一个二级内核中的一个，所述至少一个二级内核由此成为有效内核；以及

所述主内核将所述主内核的中断屏蔽级改变以反映接收处理执行控制的二级内核成为所述有效内核；以及

所述主内核将当前运行的内核标识代码改变，以将接收执行控制的所述二级内核标识为所述有效内核。

20、根据权利要求 1 所述的方法，该方法还包括：所述处理器安装用于在所述主内核与所述至少一个二级内核之间进行资源共享的应用程序接口。

用于多个内核的并行执行的方法和系统

技术领域

本发明一般涉及多任务操作系统。更具体地说，本发明涉及通过使用公共中断处理器和公共调度器以允许执行多个内核，从而在单个操作系统中支持多个内核的特征。

背景技术

通常基于使用操作系统的特殊应用，对操作系统进行设计，并对其操作进行优化。常常期望该操作系统具有一种类型的操作系统在另一操作系统中可用的特征。

例如，诸如 Linux 和 Windows 的通用计算机操作系统具有广泛的一组特征，诸如文件系统、设备驱动程序、应用程序、库等。这样的操作系统允许多个程序的并行执行，并试图优化与并行地执行程序的服务相关联的响应时间（也称作等待时间）和 CPU 使用或负荷。然而，不幸的是，这样的操作系统通常不适合于嵌入式实时应用；例如，诸如机器人、电信系统、机床、汽车系统等的控制。诸如这些以及许多其他的基于真实世界、事件和控制的应用需要所谓的硬实时性能。硬实时性能保证了最坏情况响应时间。通用操作系统（GPOS）通常折衷了针对应用程序的平均性能的程序执行时间的可预测性。包括 iTRON™ 等的几种已知的实时操作系统（RTOS）提供了硬实时特征。然而，遗憾的是，大多数 RTOS 不具有许多 GPOS 特征，并且，例如不提供对不同的文件系统、设备驱动程序、应用程序库等的支持。在许多应用中，期望具有 RTOS 的性能和通用操作系统的特征。

例如，Linux 是公知的通用操作系统，具有包括现代操作系统特征、大量开发工具、联网等的现代设备所期望的许多特征。然而，未将 Linux

设计为嵌入式操作系统。诸如而不限于机顶盒、移动电话以及汽车导航系统的许多现代设备，不仅需要诸如 Linux 的通用操作系统的特征，而且需要如实时性能的嵌入式操作系统的特征。

例如，iTRON 是普遍用在大量嵌入式设备中的成熟的实时嵌入式操作系统。iTRON 具有嵌入式设备所期望的许多特征，但是其缺乏诸如联网、对不同文件系统的支持等的 Linux 的特征。

对 GPOS 和 RTOS 二者都需要的一个示例是在汽车中使用的导航系统的控制器。该控制器从 GPS 传感器读取数据以计算该汽车的位置和方向。基于当前位置、目的地以及从导航数据 DVD 中提取的拓扑地图，该控制器计算最佳路径并在 LCD 屏上进行显示。可以用触摸板覆盖 LCD 屏，用来向导航系统输入参数。读传感器的任务、触摸板输入的任务需要硬实时；计算路径的任务、显示图形的任务、从 DVD 中读取的任务是标准编程任务，并使用通用操作系统的特征。可以通过使用诸如 iTRON 的 RTOS 内核来实现硬实时性能，而同时可以在 Linux 内核上运行通用任务。

对 GPOS 和 RTOS 二者都需要的另一示例是使用视频数据压缩硬件的固态数字视频摄像机的控制器。在这样的应用中，期望读取来自压缩硬件的数据流并执行图像处理功能，同时在 LCD 屏上显示并在可移动存储介质上存储数据。也可能例如必须使用同一控制系统来管理光学变焦机构和自动调焦机构。如果系统使用一些传统部件，则可能已存在针对特定 RTOS（例如，iTRON）可用的广泛的控制软件。可以通过硬实时操作系统（hRTOS）来很好地处理控制电机的任务、数据收集和存储的任务，而同时可以由通常在 GPOS 下的标准编程来更好地管理显示、图像处理及其他功能。此外，将在 iTRON 中可用的广泛的软件移植到另一 RTOS 的成本通常是非常高的，而且在 iTRON 中提供显示和文件系统支持等可能同样不简单。因此，在该示例中，对于该应用，将 RTOS 的特点与通用操作系统的特点相组合的系统将是最佳的。

对 GPOS 和 RTOS 二者都需要的另一示例是在为了对特定功能进行加速或附加特定功能性而需要使用专用硬件的系统中。例如，在许多多

媒体设备中，必须使用图形加速器芯片或用于音频或视频的 DSP 或 CODEC。在一些情况下，如果操作系统可以为一些任务提供有保证的性能，则可以去除对附加硬件的需要。例如，在支持流式音频的系统中，可能必须具有保证以确定的速率对压缩的和编码的音频进行解码以避免分组丢失并保持确定的输出质量的高性能任务。在一些情况下，包括 GPOS 和 RTOS 的系统可能能够去除对专业化硬件的需要，由此降低了产品成本。

可以将所有的真实世界系统分类为硬实时（HRT）系统、软实时（SRT）系统或非实时（NRT）系统。硬实时系统是这样的系统，即，在该系统中，一个或更多个活动必须永不错过最终期限或定时约束，否则认为该任务已失败。软实时系统是这样的系统，即，该系统具有计时要求，但是只要应用要求在总体上继续得到满足，则偶尔错过这些计时要求具有可忽略的效果。最后，非实时系统是即非硬实时又非软实时的系统。非实时任务不具有任何最终期限或定时约束。在许多现代应用中，必须支持各种实时系统性能。例如，考虑网络设备对安全应用的要求。网络设备可能必须在高速网络连接上对每个网络分组进行采样而不错过单个分组（硬实时任务）。硬实时任务将把这些分组存放在缓冲器中以随后对其进行处理。这可以使用 hRTOS 来实现。必须对该缓冲器中的这些分组采样进行处理和分类，但是偶尔的是，如果处理和分类慢下来，则只要该缓冲器未溢出，就将不存在问题（软实时任务）。这可以使用 hRTOS 中的任务与 GPOS 中的任务的组合来实现。在请求时，可以使用网络服务器来对已处理的和已分类的数据进行递送。通常不存在对该活动的定时约束（即，非实时任务）；因此，可以在 GPOS 中执行该任务。

鉴于以上所述，存在对于实现多内核环境的系统（例如，GPOS 和 RTOS）的需要，该系统有效地且方便地提供多内核的性能和特征并支持各种实时性能。

发明内容

为了实现上述和其他目的，并根据本发明的目的，为在多个内核之

间的并行执行和资源共享提供了多种技术。

描述了用于在多内核环境中并行执行多个内核的一种方法、系统、计算机代码和装置。在本发明的一个方法实施例中，配置了主内核和至少一个二级内核，至少一个二级内核处于所述主内核的至少部分控制之下，并且配置了可选的公共调度器，该公共调度器对在所述主内核和所述二级内核的至少一个二级内核中未决的处理的执行进行调度，并且配置了公共中断处理器，该公共中断处理器对所述主内核和所述二级内核中的至少一个二级内核中的中断和中断处理的执行进行处理。根据另一实施例，还提供了一种用于实现上述方法的装置。根据又一实施例，还提供了用于实现上述方法的计算机代码。

提供了本发明的另一方法实施例，用于在多内核环境中的多个内核之间共享系统资源，其中，配置了主内核和至少一个二级内核，所述至少一个二级内核在所述主内核的至少部分控制之下，并且配置了用于在内核之间进行系统资源共享的应用程序接口（API），调用内核设置有用于其他内核中的至少一些的合适的伪 API 调用。根据另一实施例，还提供了用于实现该方法的装置。根据又一实施例，还提供了用于实现该方法的计算机代码。

从应结合附图来阅读的以下详细描述中，本发明的其他特征、优点和目的将变得更加清楚和更加易于理解。

附图说明

在附图中，通过示例的方式而非通过限制的方式来解释本发明，并且在附图中，相似的标号表示相似的元件。在附图中：

图 1 例示了根据本发明实施例的使得能够在一个硬件平台上运行多个内核的示例性体系结构的图；

图 2 例示了根据本发明实施例的用于并行地运行多个内核的方法的流程图；

图 3 例示了根据本发明实施例的用于图 2 中所描述的主内核的选择的示例性方法的流程图；

图 4 例示了根据本发明实施例的用于图 2 中所描述的启动主内核的示例性方法的流程图；

图 5 例示了根据本发明实施例的用于图 2 中所描述的（多个）二级内核的选择和添加的示例性方法的流程图；

图 6 例示了根据本发明实施例的用于图 2 中所描述的启动二级内核的示例性方法的流程图；

图 7 例示了根据本发明实施例的用于多个内核的公共中断处理器和公共调度器的示例性体系结构的框图；

图 8 例示了根据本发明实施例的用于图 7 的情境中的多个内核的中断屏蔽级的示例性图表；

图 9 例示了根据本发明实施例的用于图 2 的框图和图 8 的条形图中所示的多个内核的中断屏蔽级的其他方面。

图 10 例示了根据本发明实施例的用于通过周期信号来切换内核的示例性方法的流程图；

图 11 例示了本发明实施例的示例性框图，其中，将公共系统调用用于多内核资源共享；以及

图 12 例示了典型计算机系统，在对该计算机系统进行适当的配置或设计时，其可以用作在其中可以具体实施本发明的计算机系统。

除非另外指明，否则不必按比例量度附图中的图示。

具体实施方式

通过参照详细附图以及这里阐述的说明来最佳地理解本发明。

下面参照附图详述本发明的实施例。然而，本领域中的技术人员将容易理解，这里给出的关于这些附图的详细说明是出于说明性目的的，而本发明的范围超出这些有限的实施例。

将在下面稍详细地描述的本发明的一个方面是要操作两个或更多个操作系统内核而同时保持这两个操作系统内核的特征和能力。

通常，可能存在开发多内核系统的许多动机。四个原因是：

1．一个内核的性能特性可能是另一内核中所期望的（例如，实时功

能性可能是通用操作系统中所期望的）

2．一个操作系统（或内核）的特征可能是另一操作系统（或内核）中所期望的（例如，文件系统、设备驱动程序、实时 API、库）

3．在一些情况下，通过使用多内核系统来去除对专业化硬件的需要，由此降低产品成本

4．可能存在对于可以支持各种实时性能的、包括 hRTOS 和 GPOS 的系统的需要。

图 1 例示了根据本发明实施例的使得能够在一个硬件平台上运行多个内核的示例性体系结构的图。如该图所示，在标有“CPU”的传统中央处理单元上执行标有内核 0、内核 1、内核 2、内核 n 的多个内核。应理解，尽管该图和以下公开内容示出并详述了单 CPU 系统的情境中的实施例和示例，但是按照本发明的教示并根据已知的技术，本发明并不限于单 CPU 实现，并且可以进行合适的配置以使用多 CPU 系统来适当地执行本发明的教示。这里，称内核 0 为主内核，内核 1、内核 2、...、内核 n 代表由内核 0 执行的一定数量的内核，内核的数量通常可能受到系统资源的限制。这些内核可以属于通用操作系统（GPOS）或实时操作系统（RTOS），并且各内核可以在其所提供的特征和能力中宽泛地变化。

接下来将稍详细地描述本发明的内核选择方面。图 2 例示了根据本发明实施例的用于并行地运行多个内核的方法的流程图。所示的本处理中的各步骤将在后续附图中更详细地单独地进行示出。该处理由选择在步骤 210 中选择的具有最大能力和最多特征的通用操作系统或操作系统的内核（例如，图 1 中的那些内核）作为主内核（内核 0）开始，并在步骤 220 中启动主内核（内核 0）。在本示例的情境中，启动该主内核包括硬件加电、装入引导装入器，该引导装入器适当地装入或执行内核 0。内核 0 在启动时启动中断处理器、调度器、任务管理器等，并通过安装合适的驱动程序来对系统硬件进行初始化。接着，在步骤 230 中，添加在该主内核中不具备的但期望具备的具有给定的目标应用所期望的特定特征的内核作为该主内核的动态模块（例如，内核 1、内核 2、... 、内核 n）。该处理在步骤 230 处循环，回到步骤 230，直至添加了所有期望的内核为

止。

各二级内核在激活时优选地被分配唯一的内核标识方式（ID），下面将稍详细地例示该标识的使用。优选地预分配这些内核 ID。在步骤 240 中，根据已预分配的中断屏蔽和内核 ID，由主内核对已添加的内核进行选择，然后在步骤 250 中，激活已添加的（或二级）内核内核 1、内核 1、...、内核 n 作为动态模块。

图 3 例示了根据本发明实施例的用于图 2 中所描述的主内核的选择的示例性方法的流程图。该处理在选择具有最期望的能力和特征的公共中断处理器的步骤 310 开始。接着在步骤 320 中，将该公共中断处理器所属的内核指定为主内核。在步骤 330 中，选择调度器作为公共调度器。该调度器可以是主内核或任何其他内核的调度器。取决于具体应用的需要，本发明的另选实施例可以实现或相反使得主内核和/或本系统能够使用任何合适的中断处理器或调度器。在本发明另一实施例中，可以不存在主控制内核，而是由公共任务调度器和公共中断处理器来控制多内核系统中的所有内核，这些内核并不直接地互相控制。在本发明另一实施例中，可以不使用主内核的中断处理器，而是在主内核之外实现另一中断处理器，在该情形下，可以根据已知的技术来实现中断处理器仿真器，并且可以另外实现本发明的其他新颖方面。应理解，在本实施例中，将其默认中断处理器用于公共中断处理器的内核自动地成为多内核系统的主内核。还应理解的是，在一些应用中，系统设计者可以选择去除主内核的默认中断处理器并用另一中断处理器来代替它，其中，任一情况下，将主内核可用的有效的中断处理器认为是针对本实施例的目的的内核的一部分。根据本发明的教示的公共中断处理器的其他合适的实现变化的多样性对于本领域中的技术人员将变得更清楚。

回到图 3，在步骤 340 和步骤 350 中，分别将唯一的 ID 和中断屏蔽级分配给主内核。下面将稍详细地描述中断屏蔽级。

图 4 例示了根据本发明实施例的用于图 2 中所描述的启动主内核的示例性方法的流程图。该处理在安装主内核的公共中断处理器的步骤 410 开始。接着在步骤 420 中安装公共调度器。在步骤 430 中安装公共应用

程序接口（API）用于资源共享。预先已知，是否将存在一个或更多个二级内核，或者，所选择的二级内核的何种特定资源将对资源共享是可用的。用于资源共享的公共 API 允许资源的无限共享，而无需预先知晓资源 API 的细节。在步骤 440 中，安装周期任务或处理，该周期任务或处理根据取决于具体应用的期望的切换方案而将执行切换到二级内核。在优选的实施例中，通过硬件计时器中断来触发内核之间的切换。取决于具体应用的需要，本领域中的技术人员可以按照本发明的教示来建议其他合适的可能是非周期的或是事件驱动的切换方案。通过示例的方式而非通过限制的方式，包括周期方案、非周期方案、基于事件的方案、基于优先级的方案的合适的内核切换方案可以用于内核之间的切换。

图 5 例示了根据本发明实施例的用于图 2 中所描述的（多个）二级内核的选择和添加的示例性方法的流程图。该处理在步骤 510 开始，在该步骤中，选择具有从在其中将安装多内核软件的系统的需要导出的一组期望的能力和特征的二级内核。分别在步骤 520 和步骤 530 中将唯一的 ID 和中断屏蔽级分配给二级内核。

图 6 例示了根据本发明实施例的用于图 2 中所描述的启动二级内核的示例性方法的流程图。该处理在步骤 610 开始，在该步骤中，将用于二级内核的所谓“连接程序（hook）”安装到主内核的公共中断处理器中。接着在步骤 620 中，将用于二级内核的连接程序安装到公共调度器中。在步骤 630 中，将用于二级内核的连接程序安装到公共应用编程接口（API）中。该连接程序使得实现了从主内核中的控制应用（例如，中断处理器或调度器或公共 API）到与该连接程序相关联的特定二级内核的控制路径。

接下来将稍详细地描述本发明的中断屏蔽和内核优先级方面。所有现代计算机系统都具有可以选择性地启用或禁用的中断。中断屏蔽级确定允许哪些中断以及不允许哪些中断对处理器进行中断。图 7 例示了根据本发明实施例的用于多个内核的公共中断处理器和公共调度器的示例性体系结构的框图。在该图中，内核 0 的屏蔽级是这样的以致允许所有的中断。在本发明的优选实施例中，为各二级内核分配仅当该内核运行

时才启用的中断范围。在本实施例中，这通过使用屏蔽级来实现。

大多数现代处理器支持中断屏蔽级。如上所指出的，内核屏蔽级确定内核允许哪些中断以及不允许哪些中断。然而应注意，即使中断可能是内核所允许的，但也可能不由该内核来处理它。因此，本实施例具有关于内核和中断的三个中断条件：（1）内核可以阻塞中断，（2）内核可以允许中断但不对该中断进行处理，以及（3）内核可以允许中断并且对该中断进行处理。假定将某个内核允许并处理的中断分配给该内核。此外，内核 0 允许并处理所有的中断。也可以将各中断唯一地分配给任一其他内核。因此，在本实施例的方法下，中断必须是内核 0 所允许并处理的，并且一个且仅一个其他内核可以允许并处理该中断。本发明的一些实施例还为中断提供了优先级，可以由 CPU 的设计来指定这些优先级，或者由本领域中的技术人员所知的其他方法来指定。

在本实施例的典型应用中，在多内核系统的设计处理中，优选地指定中断的优先级，以使得将最高优先级的中断分配给具有最高的执行优先级的内核。如图 7 所示，内核 1 具有高于内核 0 的优先级，内核 2 具有高于内核 1 的优先级，等等。内核 n 具有最高的优先级。因此，内核 0 可以被内核 1、内核 2、...、内核 n 占先。内核 1 可以被内核 2 至内核 n 占先。内核 n 不能被任何内核占先。按照本发明的教示，其他另选的和合适的中断优先化方案对于本领域中的技术人员将容易地变得清楚。

接下来将稍详细地描述本发明的中断处理方面。本发明的新颖方面在于首先对公共中断处理器进行选择。与该公共中断处理器相关联的内核称为主内核。在优选的实施例中，由内核 0 中断处理器对所有的中断进行处理。在接收中断（710）时，内核 0 执行非内核特定中断服务例程，并接着将控制传递到该特定中断所分配给的内核的中断处理器。再参照图 7，当分配给内核 n 的中断 N 发生时，首先由内核 0 处理器对其进行处理，然后调用内核 n 的中断服务例程，在该情况下内核 n 在此称为目标内核（720）。应注意，当调用中断处理器时，中断处理器执行内核独立中断处理功能；并且，将控制传递到目标内核的中断服务例程。使用中断屏蔽级来优选地识别目标内核。在这种方式中，内核 0 的中断处理

器用作多内核系统的公共中断处理器。

图 8 例示了根据本发明实施例的用于图 7 的情境中的多个内核的中断屏蔽级的示例性图表。该图示出本实施例中的中断屏蔽级如何用于为各中断确定目标内核。在该图表的左侧或轴上将中断号示出为升序号，N 为中断的总数量。

竖条的打点的（或大部分空白的）区域（810）示出相应内核所处理并允许的中断。阴影区域（820）示出相应内核所允许的中断。砖块纹理区域（830）示出当相应内核正运行时，即在 CPU 时间的控制之下，被阻塞的中断。

图 9 例示了根据本发明实施例的用于图 2 的框图和图 8 的条形图中所示的多个内核的中断屏蔽级的其他方面。该图中所示为屏蔽级如何确定内核将处理哪些中断、内核可以禁用哪些中断以及给定的内核可以启用哪些中断的示例。在该图表的左侧或轴上将中断号示出为升序号（910），N 为中断的总数量。

具体地说，在最右边条处的竖条中，将第 i 个内核表示为 Ki，“ai”（920）表示可以由内核 Ki 启用的中断，“bi”（930）表示可以由内核 Ki 禁用的中断，“ci”（940）表示可以由内核 Ki 处理的中断。

接下来将稍详细地描述本发明的调度方面。在大多数传统操作系统中，使用硬件计时器来周期性地调用调度器。通常将硬件计时器设置为触发周期中断以起动调度事件。多内核系统中的各内核可以取决于该操作系统的目的而具有用于调用调度器的不同的周期。非限制性地例如，在通用操作系统的情况下，10 毫秒周期对于期望的性能可能是足够的。然而，在实时内核的情况下，可能必须每 100 毫秒具有一次调度事件。

在本实施例中，为多内核系统选择公共调度器。所有的调度事件优选地首先由该公共调度器进行接收。在执行内核独立调度功能之后，该调度器优选地对当前运行的内核的调度器的控制（730）进行传递。为了该示例的目的，将当前运行的内核定义为当调度事件发生时运行的内核。

接下来将稍详细地描述本发明的多内核执行方面。本发明的另一新颖方面在于即使当较高优先级的内核正在执行时，系统也允许在机会出

现时（即，高优先级的内核中的任务不在运行状态—例如，等待、睡眠、休眠、...等）执行较低优先级的内核中的任务。

图 10 例示了根据本发明实施例的用于通过周期信号来切换内核的示例性方法的流程图。在该图所示的实施例中，通用内核（内核 0，未示出）运行周期处理，该周期处理通过在步骤 1005 中产生触发内核切换处理的周期信号来从一个内核切换到另一内核，由此该处理开始进行首先确定在另一内核中存在任何未决任务。这可以通过许多合适的方法来实现；该图中示出了一个合适的方法，其中，采用了一系列轮询方法来确定链中的内核是否具有任何要执行的未决任务。在所示的示例中，在步骤 1005 中产生周期信号时，内核 1 轮询任何要执行的未决任务。如果内核 1 具有一个或更多个要执行的未决任务（“是”路径），则例如通过将当前运行 id 改变为内核 1 的 id 以由此将 CPU 时间转换为执行该（多个）未决任务来实现内核 1 中的（多个）未决任务的执行。如果内核 1 不具有要执行的未决任务（“否”路径），则该处理继续轮询下一内核；例如步骤 1020 中的内核 2，针对链中的各后续内核以相同的方式继续该处理，直到达到最后内核（步骤 1030 中的内核 n）为止。一旦已执行了内核中的所有的未决任务或没有发现未决任务（即，通过步骤 1010 至 1030 的“否”路径），该处理就终止了，并且重新开始执行内核 0 任务。然而，在本发明的一些另选实施例中，不是在返回对主内核的控制之前必须完成所有的未决任务，而是可以根据已知的技术来实现另一轮询或切换方案（非限制性地例如，首先仅服务最高优先级的内核，并接着服务在随后的途径上的较低优先级内核等）以在将控制返回到主内核之前服务未决处理的至少一部分。该处理在步骤 1005 中产生下一周期信号时重新开始。取决于具体应用的需要，本领域中的技术人员将认识到按照本发明的教示的多种另选的和合适的切换方案。

接下来将稍详细地描述本发明的多内核资源共享方面。本发明的又一新颖方面在于可以在主内核与任何二级内核之间以及二级内核之间对资源进行共享。在许多应用中，通常期望从一个操作系统内核使用其他操作系统内核的特征和资源。在一些情况下，这可能是实现多内核系统

的主要促进因素。

图 11 例示了本发明实施例的示例性框图，其中，将公共系统 API 用于多内核资源共享。在本实施例中，通过为支持主内核与二级内核之间的资源共享（例如，文件系统、设备驱动程序、库等）的各内核定义伪 API 系统调用（例如，系统调用 1、系统调用 2、...、系统调用 n）来实现多内核资源共享。在优选的实施例中，主内核具有针对各内核的伪 API 调用，主内核支持主内核与二级内核之间的资源共享。当将二级内核激活为主内核的动态模块时，由真实（actual）API 调用代替伪 API 调用。

当从主内核执行真实 API 调用时，二级内核调用与该 API 相对应的特定函数调用，并且在二级内核下运行该函数。以这种方式，使得主内核可以利用二级内核的 API。因此，应用（用户和系统）可以得到二级内核的特征。当用户应用（1110）请求主内核（1120）执行内核 n 的 API 时，主内核（1120）使用用于资源共享的公共 API 系统调用 0（1130）来调用用于内核 n（1140）中的资源共享的公共 API 系统调用 n。用于内核 n 中的资源共享的公共 API 系统调用 n（1150）调用用户应用所请求的特定 API。

下面详述本发明的具体实施例，该实施例将该处理示例为应用于 Linux（GPOS）和 iTRON（RTOS）操作系统。应理解，本领域中的技术人员将容易地认识到如何对任何合适的 GPOS 和 RTOS 进行适当地配置以根据本发明的教示来进行操作。如同本发明认识到的，包括诸如 Linux 内核的 GPOS 和诸如 iTRON 内核的 RTOS 的混合系统将具有许多现代嵌入式设备所最期望的特征。

在上述教示的上下文中，在本实施例中，选择 Linux 内核作为通用操作内核（k0），并选择 iTRON 作为二级内核（k1）。分别选择 Linux 的调度器和中断处理器作为系统的公共调度器和公共中断处理器。在启动计算机时，首先启动 Linux 内核。插入 iTRON 内核作为 Linux 内核的运行时间动态模块。例如将唯一的内核 ID 0 和 1 分别分配给 Linux 和 iTRON。可以为 iTRON 内核 1 分配中断屏蔽级 11 至 15（例如适合于 Hitachi SH-4 实现中）。因此，例如若 iTRON 内核正在运行具有屏蔽级 1

至 10 的中断，则不允许中断。

因为将 Linux 调度器用作系统的公共调度器，所以系统使用硬件计时器来周期地调用 Linux 调度器。当触发调度事件时，调用该 Linux 调度器。该 Linux 调度器确定当调用该调度器时正在运行的内核的内核 id。如果正在运行的内核是 Linux，那么例如调用 linux_schedule()函数，如通过以下伪代码所例示的：

```
#define DUET_NUM_KERNELS    2    /* 假设两个内核（一个主内核比方说 Linux 和一个二级内核比方说 itron）*/
#define DUET_NUM_POINTERS    3
typedef int (*duetptr)(unsigned long);
typedef int (*duetptr2)(signed long, unsigned long, unsigned long *, unsigned long *);
int linux_do_IRQ(unsigned long);
int duet_running_kernel=0;
int linux_sys_call(signed long function_id, unsigned long argc, unsigned long * arg_type, unsigned long * arg_arr)
{
    return 0;
}
duetptr              duetptrarr[DUET_NUM_KERNELS][DUET_NUM_POINTERS]=
{
    {linux_schedule, linux_do_IRQ, 0},
    {0, 0, 0}
};
duetptr2             duetptr2arr[DUET_NUM_KERNELS][DUET_NUM_POINTERS]=
{
    {linux_sys_call, 0, 0},
```

```
    {0, 0, 0}
};
asmlinkage int sys_duet_sys_call(signed long kernel_id, signed long
function_id, unsigned long argc, unsigned long *arg_type, unsigned long
*arg_arr)
{
    if(duetptr2arr[kernel_id][2])
        return duetptr2arr[kernel_id][2](function_id, argc, arg_type,
        arg_arr);
    else
        return -200;    /* 无效内核 */
}
asmlinkage void schedule(void)
{
duetptrarr[duet_running_kernel][0](0);
}
```

可以取决于哪个内核正在运行而对某些中断进行屏蔽。例如，当iTRON内核正在运行时，对具有屏蔽级1至10的所有中断（例如，SH-4实现）进行屏蔽。如果具有屏蔽级11至15的中断发生，则调用Linux中断处理器。Linux中断处理器执行非iTRON专用代码，接着使用do_IRQ来执行iTRON中断处理器，如通过以下伪代码所例示的：

```
asmlinkage int do_IRQ(unsigned long r4, unsigned long r5,
                      unsigned long r6, unsigned long r7,
                      struct pt_regs regs)
{
    return duetptrarr[duet_running_kernel][1]((unsigned long)&regs);
}
```

在本实施例中，如果需要安装二级内核（诸如iTRON），那么主内核首先安装以在内核之间切换执行为目的的周期信号。该周期信号可以

由硬件计时器来触发。当该周期信号发生时，中断处理器确定在二级内核（iTRON）中是否存在任何未决执行任务，如果不存在，则将执行传递到 Linux 内核。这允许在二级内核空闲的同时在主内核中执行任务。

在本实施例中，当主内核（例如，Linux 内核）将执行传递到二级内核(例如,iTRON 内核)时,首先将中断屏蔽级改变为该二级内核(iTRON)的中断屏蔽级是优选的。非限制性地例如，当将执行传送到 iTRON 时，通过调用如下所示的 linux_2_itron()来设置中断屏蔽级。其将中断屏蔽设置为 0x000000A0。现将仅允许 11 与 15 之间的中断。如果具有中断级 0 至 10 的中断发生，则忽略该中断，如通过以下伪代码所例示的：

```
void linux_2_itron(void)
{
    /* 屏蔽 */
    duet_imasks=0x000000A0;
    /* 设置 iTRON 调度, IRQ */
    duet_running_kernel=1;
}
```

当将执行从 iTRON 传送回 Linux 时，将中断屏蔽设置为允许所有中断的 0x00000000。应注意，在对执行进行传送之前，还将内核 id 改变为要将该执行传递到的内核的 id。例如，当将执行从 Linux 传递到 iTRON 时，内核 id 从 0 改变为 1。当将执行返回 Linux 时，内核 id 从 1 改变为 0，如通过以下伪代码所例示的：

```
void itron_2_linux(void)
{
    /* 设置 linux 调度, IRQ */
    duet_running_kernel=0;
    /* 无屏蔽 */
    duet_imasks=0x00000000;
}
```

在 Hitachi SHx 系列处理器上实现以上系统。Hitachi SHx 处理器和许

多其他处理器都支持显式中断优先级。在硬件中不支持中断优先级的系统中，可以通过仿真或一些其他技术来在软件中实现中断优先级。

大多数传统实时嵌入式系统使用基于事件的编程；即，当特定事件发生时执行任务。在良好编程的嵌入式计算机系统中，系统 CPU 大部分时间处于休息空闲状态。可以将大多数（若非所有的）嵌入式应用视为包括三种类型的任务；这三种类型的任务是，硬实时（HRT）、软实时（SRT）和非实时或常规（NRT），在接下来将对其进行稍详细描述的本发明的实施例中将影响这些任务模型以及相应的中断模型。在该情境中，本发明的另一方面利用嵌入式系统中的该典型空闲时间来提高通用操作系统的性能和占空度。在本发明的影响上述任务模型和系统空闲时间的一个实施例中，将 HRT 任务实现为 RTOS 内核中的一个或多个任务，非限制性地例如使用 iTRON API 的 iTRON 内核；使用一个或多个 RTOS 内核来实现 SRT 任务，非限制性地例如 iTRON API 和/或一个或多个 GPOS 内核，非限制性地例如 Linux 库（系统调用和内核 API）；并且，使用一个或多个 GPOS 内核来实现 NRT 任务，非限制性地例如标准 Linux API。

本实施例适合于使用已知的或尚待开发的 RTOS 和 GPOS 系统的任何组合；然而，为了后续讨论清楚，将假设 RTOS 是 iTRON 且 GPOS 是 Linux。根据本实施例的方法，只要在 iTRON 内核中存在未决执行任务，Linux 处理就不会得到对其进行执行的机会。如果存在多于一个准备执行的任务，则首先执行具有最高优先级的任务，接着执行具有次最高优先级的任务等等，直到不再存在处于准备好的或未决状态的任务为止。

在 iTRON 系统中不存在未决执行任务的情况下，将执行控制传递到 Linux，其中，同样首先执行具有最高执行优先级的任务。为了保持等待时间合理地小，所有的 SRT 任务都具有比标准 Linux 处理（即，NRT 任务）更高的执行优先级。在优选的实施例中，使用 Linux“RT 优先级”来实现 Linux 中的 SRT 与 NRT 之间的优先级系统。因此，直到不存在 SRT 未决执行任务才执行 NRT 处理。在另选的实施例中，可以实现任何合适的公共域或专有优先级管理系统以管理 HRT、SRT 和 NRT 处理的优先级和调度。

如先前所提到的，本发明的另一新颖方面在于这样的处理，即，通过该处理多个内核可以共享诸如文件系统、设备驱动程序、库等的资源。在本发明的一个实施例中，通过为支持资源共享的各内核定义伪 API 调用来实现该资源共享处理。非限制性地例如，非常期望从 GPOS 内核（例如，Linux）中使用在 RTOS 内核（例如，iTRON）中可利用的特征。下面非限制性地通过伪代码示例表示针对 iTRON 内核的伪 API 调用：

```
#define ITRON_BAS_ERR     300
int itron_syscall(signed long function_id, unsigned long argc, unsigned
long * arg_type, unsigned long * arg_arr)
{         switch(function_id)
```

```
/************************************************/
/* 函数代码 */
/************************************************/
    /* 4.1 节 任务管理服务调用 */
    case TFN_CRE_TSK: /*(-0x05)*/
    case TFN_DEL_TSK: /*(-0x06)*/
    case TFN_ACT_TSK: /*(-0x07)*/
/* 4.4.1 节 信标（semaphore）服务调用 */
case TFN_CRE_SEM: /*(-0x21)*/
return (cre_sem((ID)arg_arr[0], (T_CSEM *)arg_arr[1]) - ITRON_BAS
_ERR);
case TFN_DEL_SEM: /*(-0x22)*/
return (del_sem((ID)arg_arr[0])-ITRON_BAS_ERR);
case TFN_SIG_SEM: /*(-0x23)*/
return (sig_sem((ID)arg_arr[0])-ITRON_BAS_ERR);
    default:
return INVALID_FUNCTION }
}
```

当首次激活（装入）二级内核作为动态运行时间模块时，将伪 API 调用连接到真实 API。当在 Linux 下激活 iTRON 作为动态模块时，由真实 API 调用代替伪 API 调用。以这种方式，使得主内核（例如，本示例中的 Linux）可以利用全部二级内核（例如，本示例中的 iTRON）API，如在以下伪代码中所非限制性地例示的：

```
asmlinkage int sys_duet_sys_call(signed long kernel_id, singed long function_id, unsigned long argc, unsigned long *arg_type, unsigned long *arg_arr)
{
if(duetptr2arr[kernel_id][2])
return duetptr2arr[kernel_id][2](function_id, argc, arg_type, arg_arr);
else
return INVALID_KERNEL_; /* 无效内核 */
}
duetptr2 duetptr2arr[DUET_NUM_KERNELS][DUET_NUM_POINTERS ]=
{
{linux_sys_call, 0, 0},
{0, 0, 0}
};
```

为了将本实施例用于 Linux 的运行时间动态模块，可以通过示例且非限制性的方式使用以下伪代码：

```
void duet_init_itron(void)
/* 设置 duet 进度, IRQ */
duetptrarr[1][0]=itron_schedule;
duetptrarr[1][1]=itron_IRQ;
duetptr2arr[1][2]=itron_syscall; /* 安装 itron_syscall */
duet_imaskc=0x000000D0;
```

```
duet_imasks=0x00000000;}
```

当移除 RTOS 模块（例如，iTRON）时，如在以下伪代码中所非限制性地例示地移除伪 API：

```
void duet_deinit_itron(void)
{
duetptr2arr[1][2]=0; /* 卸载 itron_syscall */
duet_imaskc=0x000000F0;
duet_imasks=0x00000000;
}
```

以这种或类似的方式，通过使用伪 API 调用，主内核可以执行专门使得主内核可以通过伪 API 来利用的二级内核函数。可以期望，该机制使得可以使用两个内核之间的复杂的相互作用，该相互作用包括但不限于数据共享、任务同步和通信功能（信标、事件标志、数据队列、信箱）。通过使用伪 API 和使用 GPOS（例如，Linux）系统调用，按照本发明的教示，本领域中的技术人员可以在实时嵌入式程序中开发可以利用 Linux 的丰富特性（例如，文件系统、驱动程序、网络等）的程序。本发明的一些实施例可能不包括上述公共调度器和/或公共伪 API，因为它们是可选的。即，可以使用本发明的公共中断处理器来运行多个内核，而无需公共调度器和/或公共伪 API。然而，在许多应用中，公共调度器提供提高的性能和更佳的误差处理。不需要多个内核之间的资源共享的应用可以不实现本发明的上述公共伪 API 方面。

图 12 例示了典型计算机系统，在对该计算机系统进行适当的配置和设计时，其可以用作在其中可以具体实施本发明的计算机系统。计算机系统 1200 包括任意数量的处理器 1202（也称作中央处理单元或 CPU），处理器 1202 连接到包括主存储器 1206（通常为随机存取存储器或 RAM）、主存储器 1204（通常为只读存储器或 ROM）的存储设备。CPU 1202 可以是包括微控制器和微处理器的各种类型，诸如可编程设备（例如，CPLD 和 FPGA）和诸如门阵列 ASIC 或通用微处理器的不可编程设备。如本领域中公知的，主存储器 1204 用作将数据和指令单向地传送到 CPU，并且

通常使用主存储器 1206 来以双向方式传送数据和指令。这两个主存储器都可以包括任何合适的诸如上述的那些计算机可读介质。也可以将海量存储设备 1208 双向地连接到 CPU 1202，并且海量存储设备 1208 提供附加的数据存储容量并可以包括任何上述的计算机可读介质。可将海量存储设备 1208 用于存储程序、数据等，且海量存储设备 1208 通常是诸如硬盘的二级存储介质。应理解，在适当的情况下，可以将海量存储设备 1208 内保持的信息以标准方式并入作为主存储器 1206 的一部分的虚拟存储器。诸如 CD-ROM 1214 的专用海量存储设备也可以将数据单向传递给 CPU。

也可以将 CPU 1202 连接到接口 1210，接口 1210 连接到一个或更多个输入/输出设备，诸如视频监视器、跟踪球、鼠标、键盘、麦克风、触控式显示器、传感器读卡器（transducer card reader）、磁带阅读器或纸带阅读器、图形输入板、指示笔、声音或手写识别器或显著成绩然地诸如其他计算机的其他公知的输入设备。最后，可选的是，可以使用 1212 处总体地示出的外部连接将 CPU 1202 连接到诸如数据库、计算机、电信网络或互联网络的外部设备。使用这样的连接，可以设想，在执行本发明的教示中所描述的方法步骤的处理中，CPU 可以从网络接收信息，或者可以将信息输出到网络。

本领域中的技术人员将容易地认识到，根据本发明的教示，可以将任何上述步骤和/或系统模块合适地进行替换、重排、去除，并且取决于具体应用的需要，可以插入附加的步骤和/或系统模块，并且可以使用多种合适的处理和系统模块中的任何处理和系统模块来实现本实施例的方法和系统，并不限于任何特定的计算机硬件、软件、RTOS、GPOS、固件、微代码等。

已充分描述了本发明的至少一个实施例，对于本领域中的技术人员来说，根据本发明的多个内核的并行执行以及在多个内核之间的资源共享的其他等同的或另选的方法将是清楚的。以上已通过例示的方式描述了本发明，所公开的具体实施例不旨在将本发明限制在所公开的具体形式。因此，本发明欲涵盖落入以下权利要求的精神和范围的所有的变型

例、等同物以及另选例。

相关申请的交叉引用

本 PCT 专利申请要求于 2004 年 7 月 6 日根据 35 U.S.C.119（e）而提交的专利号为 60/586486 的美国临时申请的权益。

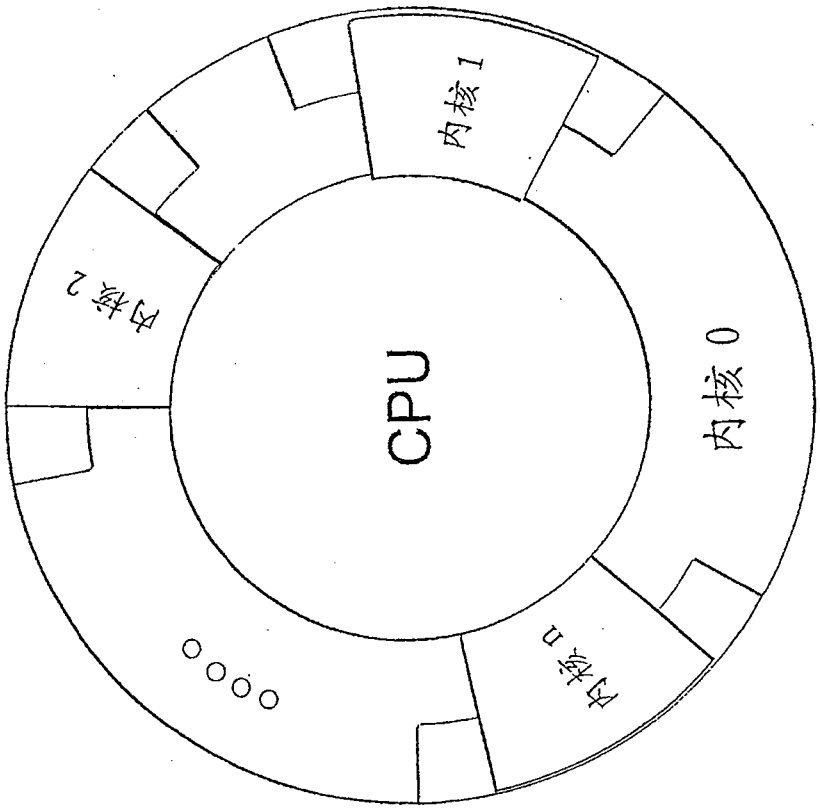


图 1

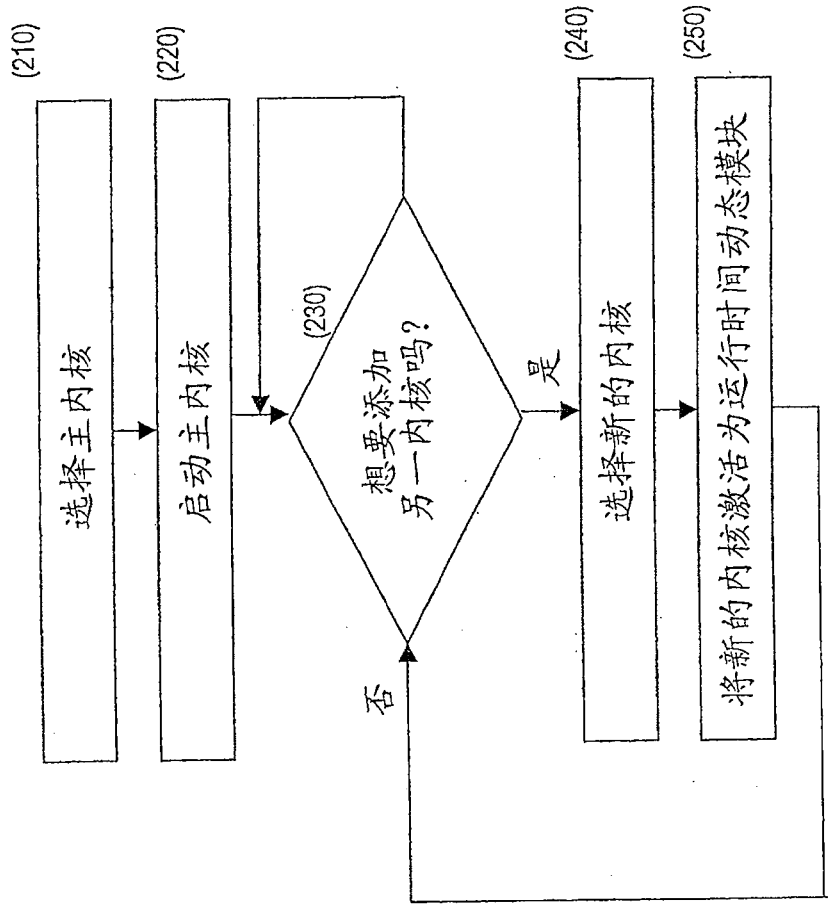

(210)
选择主内核
(220)
启动主内核
(230)
想要添加另一内核吗?
否
是
(240)
选择新的内核
(250)
将新的内核激活为运行时间动态模块

图 2

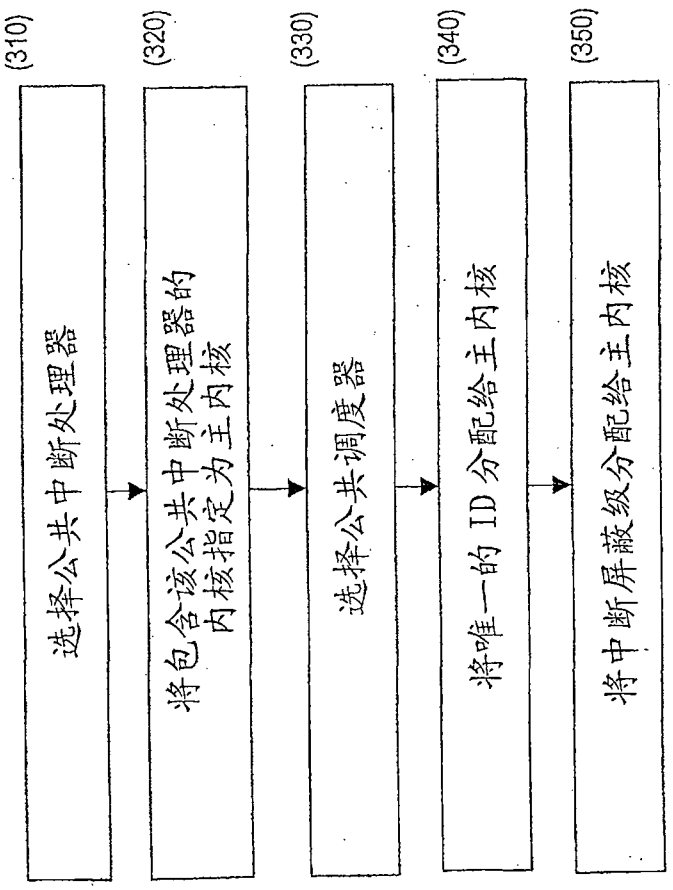

(310)
选择公共中断处理器
(320)
将包含该公共中断处理器的内核指定为主内核
(330)
选择公共调度器
(340)
将唯一的 ID 分配给主内核
(350)
将中断屏蔽级分配给主内核

图 3

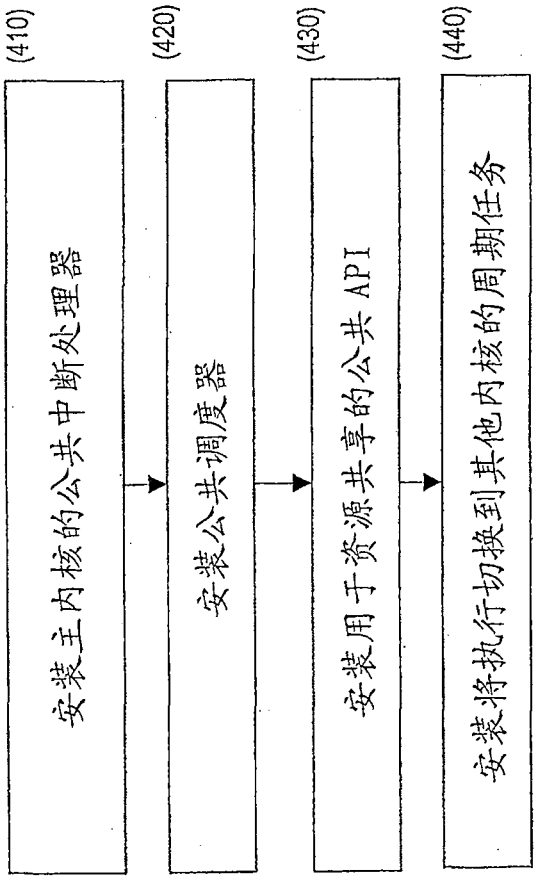

(410)
安装主内核的公共中断处理器
(420)
安装公共调度器
(430)
安装用于资源共享的公共 API
(440)
安装将执行切换到其他内核的周期任务

图 4

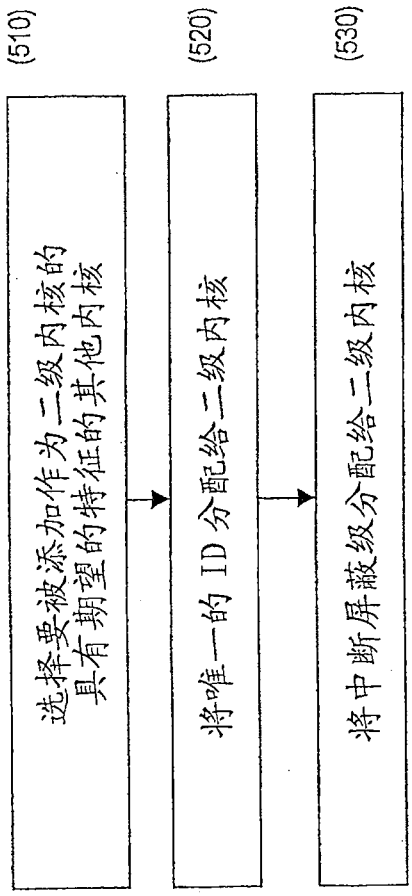

(510)
选择要被添加作为二级内核的具有期望的特征的其他内核
(520)
将唯一的 ID 分配给二级内核
(530)
将中断屏蔽级分配给二级内核

图 5

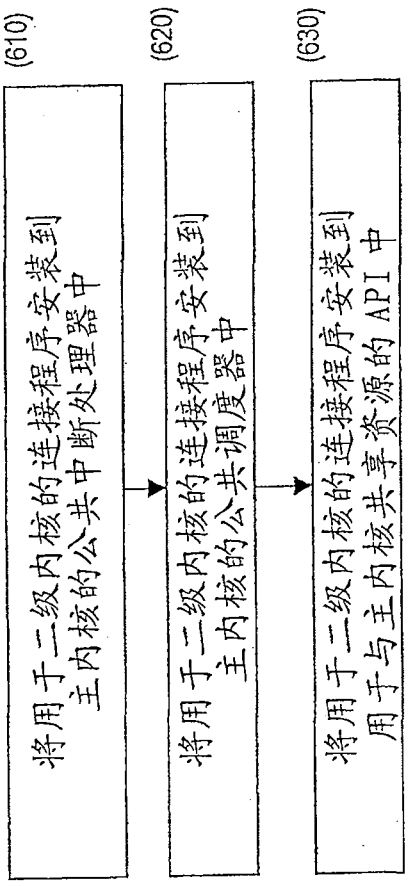

(610)
将用于二级内核的连接程序安装到主内核的公共中断处理器中
(620)
将用于二级内核的连接程序安装到主内核的公共调度器中
(630)
将用于二级内核的连接程序安装到用于与主内核共享资源的 API 中

图 6

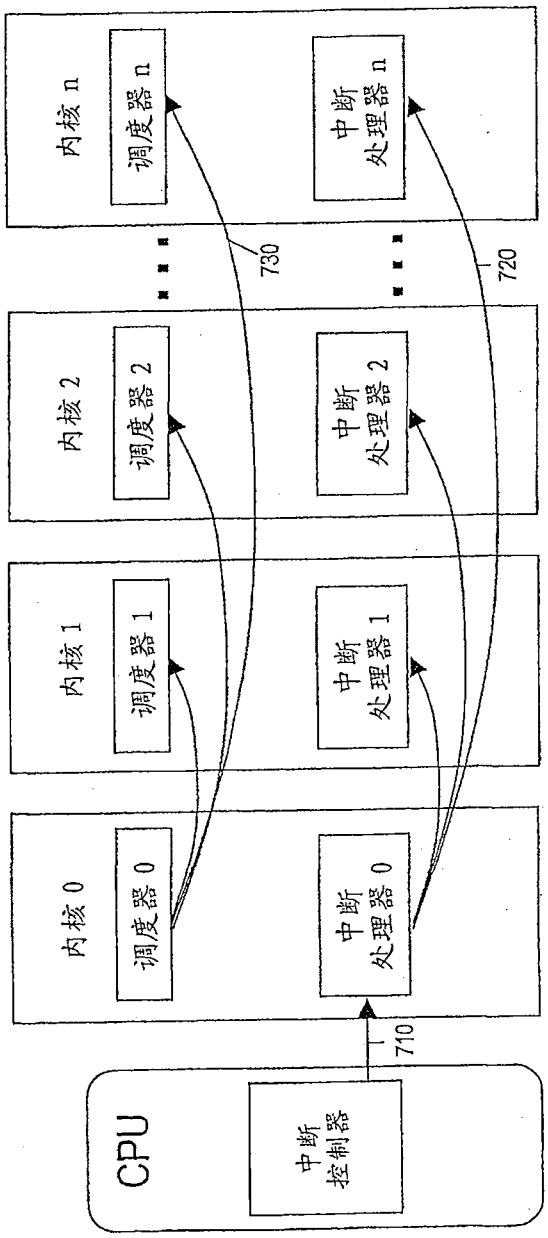

CPU
中断
控制器
710
内核 0
调度器 0
中断
处理器 0
内核 1
调度器 1
中断
处理器 1
内核 2
调度器 2
中断
处理器 2
内核 n
调度器 n
中断
处理器 n
730
720

图 7

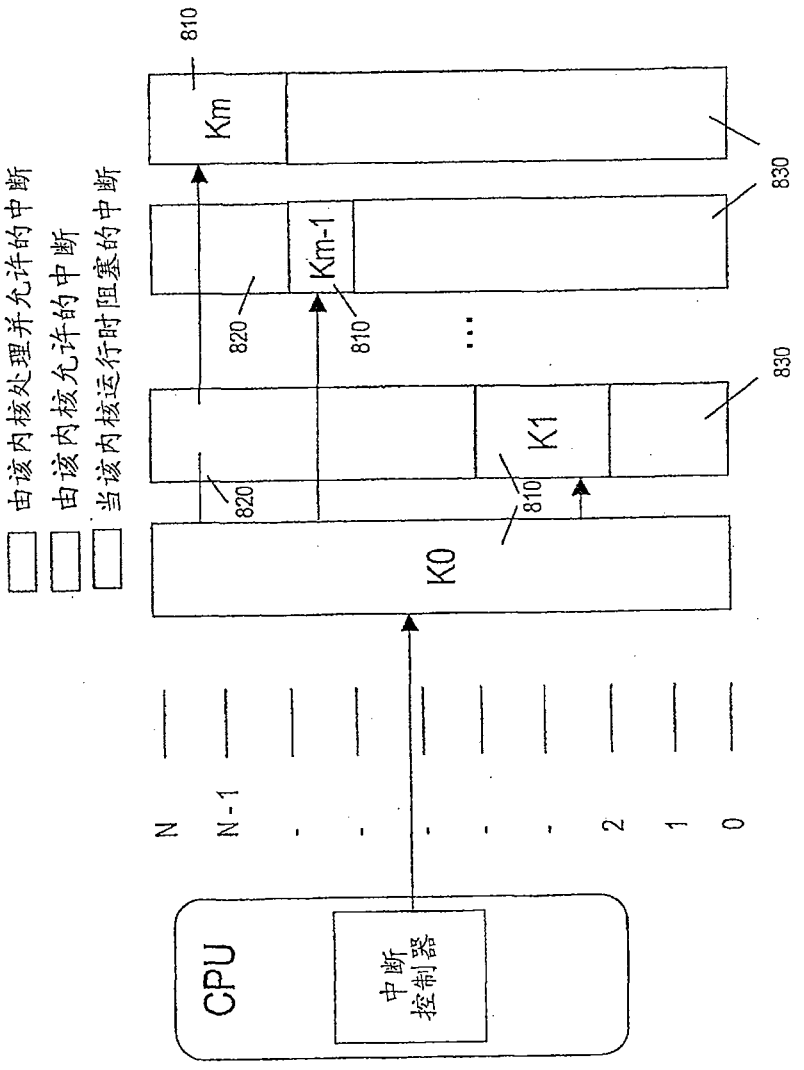

由该内核处理并允许的中断
由该内核允许的中断
当该内核运行时阻塞的中断
CPU
中断控制器
N
N-1
2
1
0
K0
K1
Km-1
Km
810
820
830

图 8

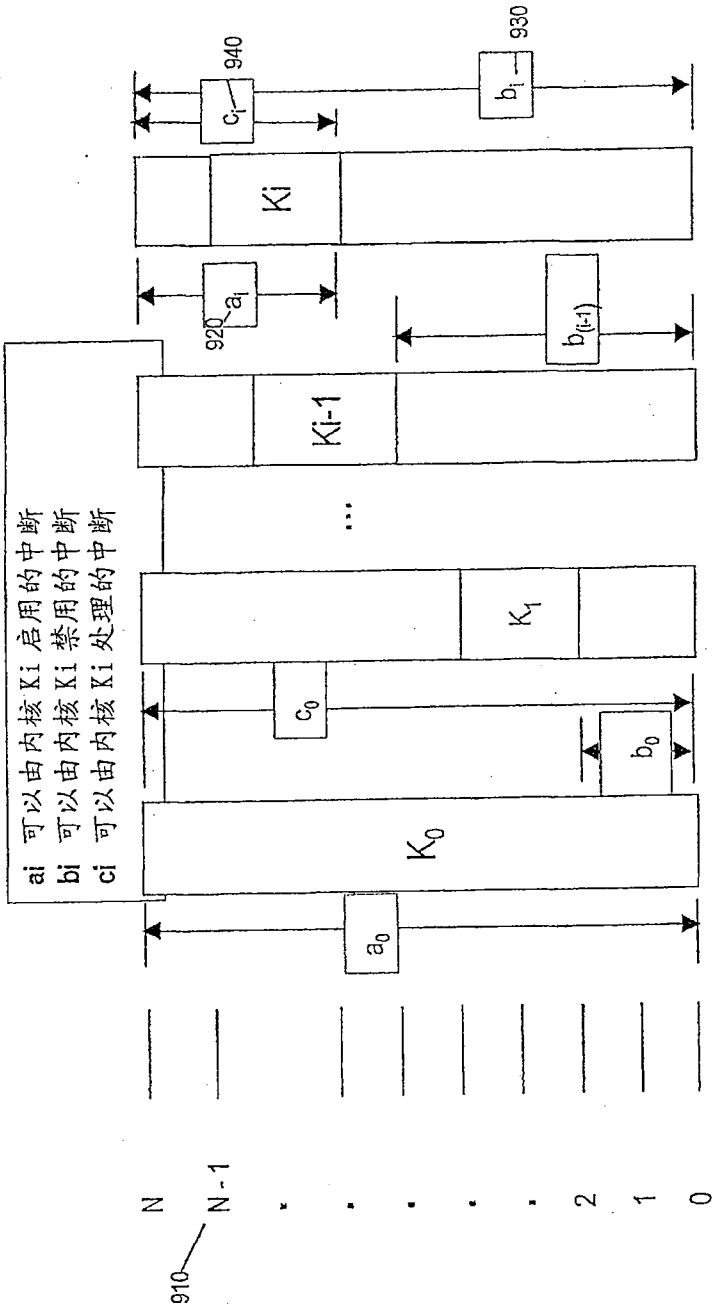

ai 可以由内核 Ki 启用的中断
bi 可以由内核 Ki 禁用的中断
ci 可以由内核 Ki 处理的中断
910
N
N - 1
2
1
0
K_0
K_1
Ki-1
Ki
a_0
b_0
c_0
a_1
b_1
c_1
$b_{(i-1)}$
920
930
940

图 9

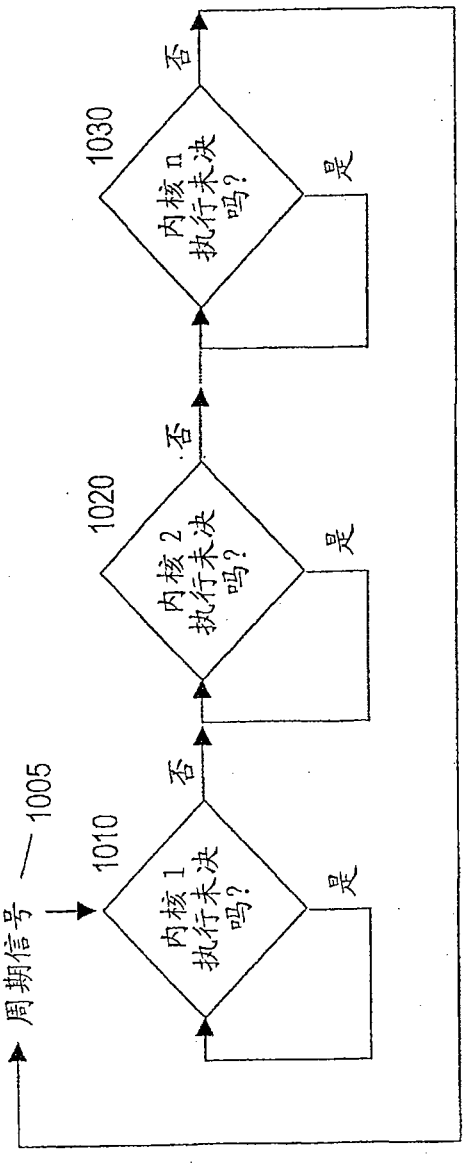

周期信号
1005
1010
内核 1 执行未决吗?
否
是
1020
内核 2 执行未决吗?
否
是
1030
内核 n 执行未决吗?
否
是

图 10

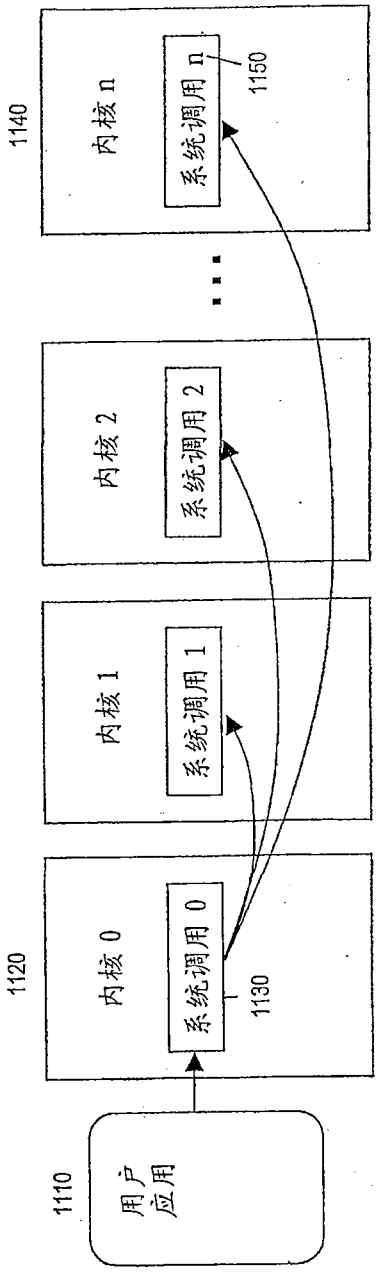

1110
用户应用
1120
内核 0
系统调用 0
1130
内核 1
系统调用 1
内核 2
系统调用 2
1140
内核 n
系统调用 n
1150

图 11

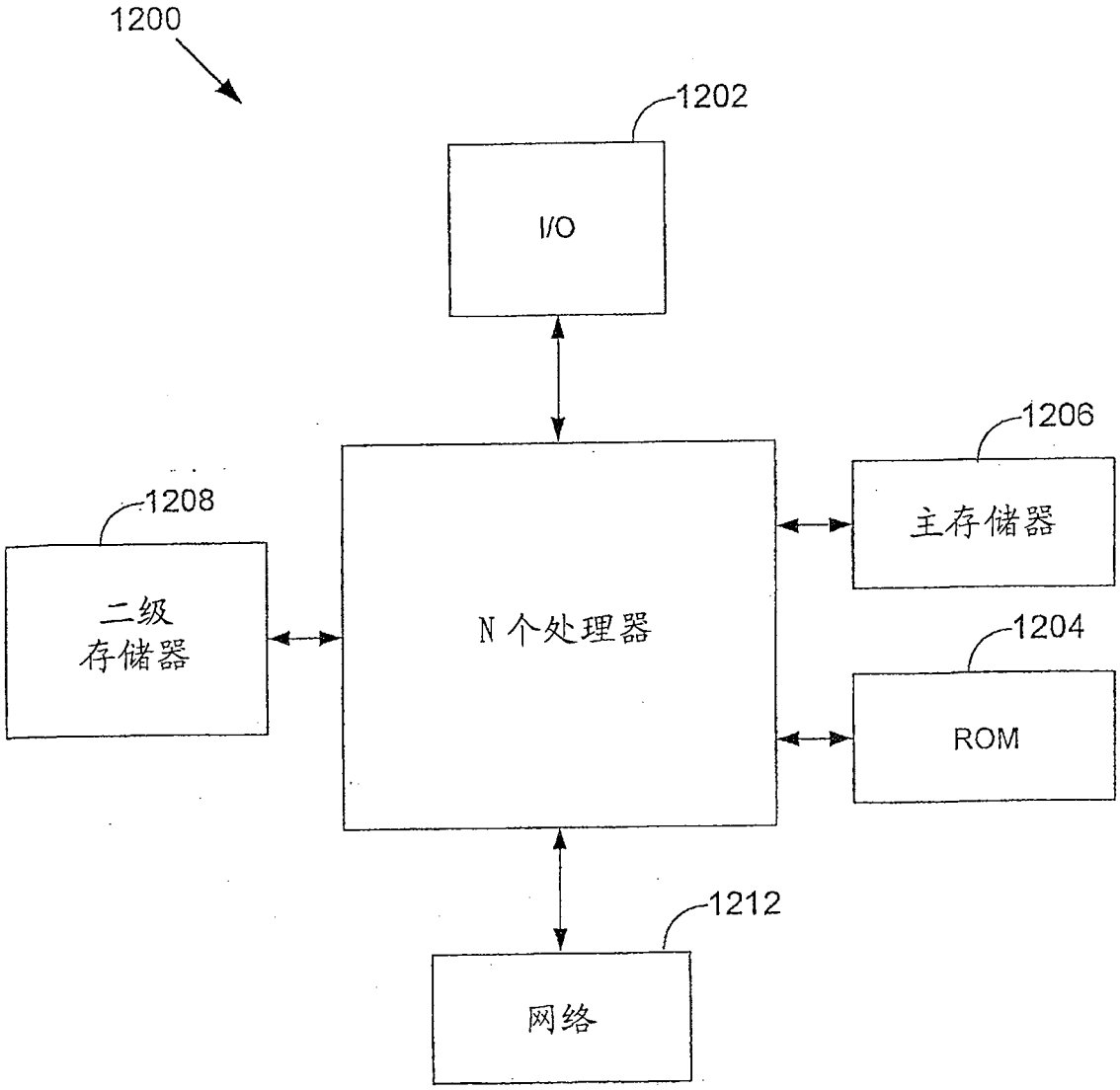


图 12