



(12) 发明专利

(10) 授权公告号 CN 106909444 B

(45) 授权公告日 2021.01.12

(21) 申请号 201710146866.7

(22) 申请日 2011.12.22

(65) 同一申请的已公布的文献号
申请公布号 CN 106909444 A

(43) 申请公布日 2017.06.30

(62) 分案原申请数据
201180075795.7 2011.12.22

(73) 专利权人 英特尔公司
地址 美国加利福尼亚州

(72) 发明人 A·赛敏

(74) 专利代理机构 上海专利商标事务所有限公
司 31100

代理人 张欣

(51) Int.Cl.

G06F 9/455 (2006.01)

G06F 16/93 (2019.01)

(56) 对比文件

CN 101821712 A, 2010.09.01

CN 1518691 A, 2004.08.04

CN 101689124 A, 2010.03.31

CN 1540508 A, 2004.10.27

CN 1308746 A, 2001.08.15

US 2008082841 A1, 2008.04.03

US 2011022833 A1, 2011.01.27

审查员 张剑峰

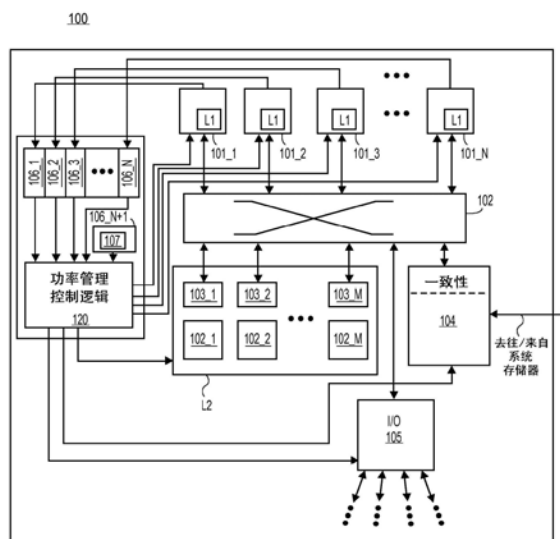
权利要求书1页 说明书8页 附图6页

(54) 发明名称

用于指定应用线程性能状态的指令的指令
处理装置及相关方法

(57) 摘要

描述了一种包括处理器的装置。该处理器具有执行指定应用线程的性能状态的指令的处理核。该指令属于应用线程。该处理器包括存储性能状态的寄存器。该处理器包括耦合到寄存器以根据该性能状态设置处理核的性能状态的功率管理控制逻辑。



1. 一种处理器,包括:

多个精简指令集 (RISC) 多线程核,所述多线程核中的至少一个多线程核同时执行多个线程;

所述多线程核中的至少一个多线程核包括:

指令获取电路,用于获取所述多个线程的指令;

一级 (L1) 高速缓存,用于在执行之前存储所述指令;

解码器,用于解码所述指令;

执行资源,用于执行所述线程的所述指令;

分配电路,用于基于对第一线程与第二线程的性能要求将相对较大百分比的所述执行资源分配给所述第一线程而非分配给所述第二线程;

功率管理电路,用于至少部分地响应于在所述第一线程和/或第二线程中执行的功率模式指令,基于对所述第一线程和/或第二线程的所述性能要求选择所述多线程核的当前功率状态;

其中,至少部分地通过执行频率/电压转换与时钟门控来实现所述功率状态;

其中,至少一个功率状态与超频率相关联。

2. 如权利要求1中所述的处理器,其特征在于,进一步基于存储在寄存器中的功率模式值来选择所述当前功率状态。

3. 如权利要求1中所述的处理器,进一步包括:

所述功率管理电路将第三线程标识为与低优先级工作相关联并响应地降低所述第三线程的性能。

4. 如权利要求1中所述的处理器,进一步包括:

共享高速缓存,用于存储多核的指令。

5. 如权利要求4中所述的处理器,进一步包括:

一致性电路,所述一致性电路耦合到所述L1高速缓存以及所述共享高速缓存。

6. 如权利要求1中所述的处理器,进一步包括:

输入/输出 (IO) 接口,所述输入/输出 (IO) 接口耦合到所述多个多线程核。

用于指定应用线程性能状态的指令的指令处理装置及相关方法

[0001] 本申请是国际申请日为2011/12/22,国际申请号为PCT/US2011/067025,进入中国国家阶段的申请号为201180075795.7,题为“用于指定应用线程性能状态的指令的指令处理装置及相关方法”的发明专利申请的分案申请。

技术领域

[0002] 本申请通常涉及计算科学,且尤其涉及指定应用线程性能状态的指令。

背景技术

[0003] 随着多年来处理器性能的增加,它们的功率消耗也已经增加。无论是“绿色能源”意识、尝试最小化电池消耗还是简单地尝试最小化计算系统的用户的物业账单,计算系统的用户日益聚焦于他们的内部处理器的功率管理能力。因而,大多数现代处理器具有内建的功率管理电路。内建的功率管理电路通常被设计为在硬件中动态地进行细粒度的功率管理调整,以及/或者,从软件支持粗粒度的功率管理调整/指导。

[0004] 通常,可以通过三种主要技术中的任意技术来控制电子电路的功率消耗:1) 频率缩放;2) 时钟门控;以及3) 电压缩放。这些技术中的每一种都考虑电子电路的功率消耗的基本原理。大体上,电子电路操作得越快,其性能和功率消耗将越大。下面提供对这些技术中的每一种的回顾。

[0005] 频率缩放调整逻辑电路的模块的工作频率。在这里,在以增加其功率消耗为代价对逻辑块期望较高性能时,提高逻辑块的工作频率。同样地,在以较低性能为代价期望较低功率消耗时,降低逻辑块的工作频率。频繁缩放的不同逻辑块的大小和功能可以取决于设计师期望的粒度而改变。

[0006] 时钟门控可以被看作是频率缩放的极端形式。在时钟门控的情况中,取消去往逻辑块的时钟信号以便将该模块的性能和功率消耗两者减少到零。在要使用该逻辑块时,时钟信号再度出现,使得该逻辑块复活。时钟门控因此具有启用/禁用逻辑块的效果。

[0007] 除了降低/提高电源电压以便降低/减少逻辑块的性能和功率消耗之外,电压缩放类似于频率缩放。值得注意的是,电子电路接收到的电源电压越高,可以施加到逻辑块的最大时钟频率就越高。

[0008] 处理器核迄今为止已经被设计为带有这样的硬件控制逻辑电路:响应于所检测到的处理器芯片的使用,可以快速地和动态地调整对处理器芯片的小的、中等的和/或大的逻辑块的频率缩放、时钟门控和/或电压缩放设置。例如,取决于指令队列中是否存在任何浮点指令,可以经由时钟门控禁用/启用流水线中的浮点执行单元。在这里,因为硬件控制可以快速地调整小尺寸或中等尺寸的逻辑块的功率管理设置,应理解,硬件功率管理控制能够进行“细粒度”控制。

[0009] 话虽如此,应理解,频率和电压缩放具有与它们各自的状态转换相关联的不期望的等待时间。即是说,即使硬件频率和/或电压缩放控制逻辑可以快速做出需要改变频率

和/或电压的决定,但实现改变本身要花费时间,这是因为,通常在没有数据恶化的风险的前提下不能对操作逻辑快速地改变电子逻辑电路的工作频率和/或供电电压。当在启用/禁用状态之间切换时,中等尺寸到大尺寸的逻辑块的时钟门控也倾向于具有相似的不期望的等待时间。举例来说,如果禁用了整个处理器核,通常不能在“下一个”时钟周期复活。

[0010] 在这一方面,值得注意的是,硬件功率管理控制是反应性的,这是因为它只能对它所观察到的处理器使用反应。在所观察的处理器使用显著改变时,至少部分由于在上面讨论的功率管理状态变化之间的等待时间,硬件控制的反应性本质引起性能冲击或工作量失衡。举例来说,如果大的多核处理器处于低功率状态,且已经由于低利用率禁用了多个核,并且突然给处理器呈现大量的线程以供执行,则新近呈现的线程中的许多必须不期望地“等待”在核心可以执行之前启用核心。

[0011] 然而,一些运行时编译器(例如,OpenMP和MPI)和操作系统(OS)调度程序可以向硬件提供即将到来的处理器使用的提示。借助于这样的提示,硬件可以为即将到来的使用改变提前做好准备,且在这样做时,通过在实际上呈现使用改变之前开始改变性能状态,理想地避免了性能冲击或工作量失衡。

[0012] 相对于软件功率管理控制,应理解,软件功率管理控制主要是(如果不是全部是的话)“粗粒度的”。即是说,软件功率管理控制通常影响中等尺寸到大尺寸的逻辑块,并且,此外,不能快速改变软件控制的功率管理设置(相反,它们持续扩展的时间周期)。因而,在处理器硬件提供软件可写控制字段或其他“钩子(hooks)”以便辅助软件功率管理的意义上,这样的字段/钩子不直接地实现任何细粒度控制。

[0013] 另外,许多现有软件功率管理控制机制依赖于或有效地仅仅监督功率管理的特定硬件实现,例如特定处理器的P状态。因而,真正的软件功率管理技术跨越不同硬件实现的可移植性较差。

[0014] 这种观点的最广为人知的可能的例外是PAUSE指令,PAUSE指令是引起执行将被置于睡眠状态的指令的线程的指令。

附图说明

[0015] 作为示例而非限制在附图的各图中阐释本发明,附图中,类似的标号指示相似的元素,且附图中:

[0016] 图1示出处理器;

[0017] 图2A和图2B涉及所述处理器的逻辑块性能状态;

[0018] 图3示出利用每个线程的TCMHINT指令来控制所述应用的功率消耗的多线程应用的结构流;

[0019] 图4示出利用每线程TCMHINT指令来控制所述第二应用的功率消耗的第二多线程应用的结构流;

[0020] 图5示出指令执行流水线的实施例;

具体实施方式

[0021] 在此描述的是专用指令TCMHINT,可以由应用软件开发施加该指令以便通过软件以细粒度方式控制他们的应用的功率管理。具体地,在此描述的所述指令和被设计为支

持它的处理器架构准许应用软件开发用多个性能状态中的一个来标记他们的应用的每一线程。由底层处理器使用每个线程性能状态标签来适当地将处理器的性能/功率消耗特性设置为与应用软件开发期望一致。同时,该指令不依赖于功率管理技术或硬件中的处理器架构的任何特定实现,这使得这一指令使用可移植到其他架构,并允许用于任何将来的可能处理器和系统架构的最佳指令实现。

[0022] 此外,可以作为在对程序流结构/行为的即将到来的改变之前的提示而嵌入TCMHINT指令,以便在改变之前“设置”处理器的性能/功率消耗特性。因而,可以避免由处理器从第一性能/功率状态转换到另一性能/功率状态所需要的时间引起的性能或工作量失衡。(由于处理器消耗的功率与其性能能力相关,在下文中,术语“性能状态”将用来指代影响处理器的性能能力或功率消耗的任何功率管理设置。)

[0023] 在这里,应用软件开发容易理解他们的应用程序的流或“结构”、他们的应用程序的性能和功率消耗含义、以及整个应用或应用的不同部分应该落在高性能与低功率消耗设计点的关系的哪一侧。根据一种实施例,可以根据三种不同性能状态中的一种标记各线程:1) 高性能(高功率消耗)——例如,其中一个或多个所标记的线程正要求最高可能性能的状态,例如在应用的串行部分(例如为IO事务提供服务):硬件实现可以允许向单个线程提供最高可能性能,从而减少线程执行的“等待时间”;2) 中等性能(中等功率消耗)——例如,其中通过多个线程的并发执行实现最好总体性能的状态,且因而可以减少每一线程的性能(相对于高性能状态),同时基于由多个线程提供的较高的计算吞吐量改善总体应用性能;3) 低性能(低功率消耗)——例如,其中可以由硬件优化线程执行以便即使付出进一步减少线程所提供的性能的代价也改善总体功率消耗的状态。

[0024] 为了进一步说明,软件开发通常知道他/她的线程化应用的特性,且可以说出应当借助于串行算法还是高度可并行算法来实现该应用的特定部分。一些算法本质上是串行的(例如排序存储在复杂结构中的链表,或仲裁IO通信),且因而开发者可以标出这种单个进程需要处理器资源(包括其功率预算)中的很多/大部分,以便为线程的指令流提供最好性能和最低计算时间(等待时间)。处理器将认识到,线程将以低并发模式运行,并采取适当的动作来重新调整其资源,以便为多个线程提供较好的性能(例如在进行到更高电平之前开始加速频率)。

[0025] 相比之下,应用的其他部分可以被称为充分并行(例如处理存储器中的规则结构化数据的大数组,或者在对完全独立数据集操作的多个线程中进行工作)且因而可以使用争用处理器资源的多个并发线程来实现。这可以由开发者(或者也由编译器或由线程化运行时)来标记,且硬件可以采取适当的动作来跨越多个核分配资源(例如功率预算),并采取其他必要的动作来并发地执行多个线程。因而,可以得到资源在处理器核和其他块之间的更均匀分布。

[0026] 更进一步,线程可以进行一些低优先级的工作(例如,杀毒例程扫描在后台运行的线程),使得如果系统被偏向最低功率则可以更进一步牺牲其性能。

[0027] 此外,借助于对处理器的哪些资源专用于单个线程且处理器的哪些资源被设计为支持多个线程的理解,设计该处理器硬件。在一个实施例中,将被设计为支持单个线程的处理器的那些部分置于由该线程执行的最近的TCMHINT指令所指示的性能状态。相比之下,被设计为支持多线程的处理器的那些部分考虑其支持的所有线程的性能状态,并且基于这一

观察将自身置于适当的性能状态。

[0028] 在更进一步的实施例中,通过可以实现以上所描述的两种设置(单线程专用资源和多线程专用资源)的软件将“系统告示”功率管理指示输入到处理器。

[0029] 图1示出基本示例。图1叙述具有多个处理核101_1到101_N(101_1、101_2、101_3、…、101_N)和L1高速缓存的多核处理器100。交叉开关(crossbar switch)102将处理核101_1到101_N与由分离的L2高速缓存片102_1到102_M(102_1、102_2、…、102_M)组成的L2高速缓存和相应的一致性协议代理103_1到103_M(103_1、103_2、…、103_M)互连起来。多处理器核还包括带有关联的一致性协议代理逻辑104和I/O中枢105的存储器控制器。可以使用放置在相同的半导体芯片上的逻辑电路来实现这些逻辑块中的每一个。在一个基本实施例,每一处理核专用于单个线程,然而,只要两个或更多个处理核活动,图1中所观察到的所有其他逻辑块就专用于多个线程。

[0030] 如图1可见,处理器100的寄存器空间被用来保存由它们各自的应用软件程序提供的个体线程106_1到106_N(106_1、106_2、106_3、…、106_N)的性能状态以及前述系统告示功率管理指示106_N+1。在一个实施例中,处理器的不同的逻辑块均能够具有四种不同的功率状态:1)低等待时间(LL);2)高吞吐量(HT);3)有效吞吐量(ET);以及4)优化的功率(P0)。下面更完整地描述这些功率状态的可能实施例的更多透彻的描述。

[0031] 目前,通常可以将它们看作是关于特定的性能和功率消耗设置的四种粒度。在这里,LL设置是最高性能状态,HT状态是下一种最高性能状态,ET状态是第二最低性能状态,P0是最低性能状态。在这里,对于任何逻辑块,较高性能状态通常对应于由功率管理控制逻辑电路120施加的工作频率、时钟门控和/或供电电压设置的总组合,这些设置对应于该逻辑块的比其较低性能状态选项中的任何更高的性能和功率消耗。注意,功率管理控制逻辑电路120可以如1图可见的集中、分布,或两种方法的某种组合。

[0032] 在一个实施例中,每一线程时常执行其自己的TCMHINT指令,这建立该线程的性能状态,直到由该线程执行的下一TCMHINT指令。因而,寄存器空间106_1到106_N分别耦合到处理器100中的N个流水线中的每一个。当流水线作为其线程执行的一部分执行TCMHINT指令时,将由TCMHINT指令指定的线程的性能状态写入到流水线的各自的寄存器中。

[0033] 在一个实施例中,每一流水线具有在寄存器空间中为它保留的与它可以并发地执行线程的数量一样多的寄存器。例如,如果流水线可以并发地执行四个线程,则存在寄存器空间中的四个不同的寄存器位置以便为其(最高达)四个并发线程中的每一个保存来自最近的TCMHINT指令的功率设置。注意,一些处理核可以具有多于一个的流水线。例如,具有两个流水线的处理核将在寄存器空间中具有八个各自的寄存器位置。一些处理器或流水线可具有给它指派的、大于它可以并发地执行的线程的数量的线程池。例如可以并发地执行四个线程的流水线可以具有给它指派的八个或十六个线程的池(以便允许处理器实时使得多个线程/应用多任务化)。在这种情况下,该流水线在激活和执行线程以及将其放回到该池中以便从该池激活另一线程之间来回切换。在这种情况下,可以扩展寄存器空间,以便不仅包括其并发执行的线程的功率状态而且也包括其池中的那些。在这种情况下,流水线可以被看作是支持其池中的所有线程的共享逻辑块。下面结合图2B进一步更详细地描述共享逻辑块和它们的适当的功率状态。

[0034] 如图2A和图2B可见,寄存器106_N+1中的系统告示功率管理指示的值107影响为专

用于单个线程的那些块(图2A)以及专用于多个线程那些块(图2B)选择这些状态中的哪些。系统告示指示107本质上是规定应当将处理器整体更加偏向较高性能(例如在服务器或高性能工作站中)还是较低功率消耗(例如在靠电池运行的移动平台中)的全局设置。

[0035] 具体地,图2A示出单个线程专用的资源(逻辑块)(例如简单的前述示例中的处理核)的适用设置。如图2A可见,在系统告示指示指定较高性能时:1) 如果通过TCMHINT指令声明逻辑块的线程处于高性能状态,则将逻辑块置于LL状态;2) 如果通过TCMHINT指令声明逻辑块的线程处于中等性能状态,则将逻辑块置于HT状态;以及3) 如果通过TCMHINT指令声明逻辑块的线程处于低性能状态,则将逻辑块置于ET状态。

[0036] 相比之下,如果系统告示指令指定较低功率:1) 如果通过TCMHINT指令声明逻辑块的线程处于高性能状态,则将逻辑块置于HT状态;2) 如果通过TCMHINT指令声明逻辑块的线程处于中等性能状态,则将逻辑块置于ET状态;以及3) 如果通过TCMHINT指令声明逻辑块的线程处于低性能状态,则将逻辑块置于P0状态。

[0037] 因而,不考虑系统告示设置,专用于单个线程的逻辑块的性能状态将与为该线程置执行的最近的TCMHINT指令相称地缩放。然而,对于相同的TCMHINT指令设置,如果系统告示值指示期望较低功率消耗,则逻辑块的性能状态缩放到更低。相反,对于相同的TCMHINT指令设置,如果系统告示值指示即使付出处理器的较高的总体功率消耗的代价也期望较高性能,则逻辑块的性能状态缩放到更高。

[0038] 图2B示出专用于支持多个线程的资源/逻辑块的适用的设置。如图2B可见,如果系统告示指示指定较高性能/功率,则共享块在逻辑块支持的所有线程当中标识具有最高功率TCMHINT指令设置210的线程,并且,接受该线程作为逻辑块支持的所有线程的模型线程。然后,共享逻辑块进入根据图2A中略述的策略211的性能状态。即是说,如果在步骤210标识的模型线程是高性能线程,则共享逻辑块进入LL状态等等。

[0039] 相比之下,如果系统告示指示指定较低性能和较低功率,则共享块标识共享逻辑块支持的线程的最低功率分级212并接受该线程作为各线程的模型线程。然后,共享逻辑块进入根据图2A中略述的策略213的性能状态。即是说,如果在步骤212标识的模型线程是高性能线程,则共享逻辑块进入HT状态等等。

[0040] 上面的方案可以应用到任何处理器架构,而不考虑处理器的什么特定块专用于特定线程,也不考虑哪些块专用于服务多个线程。注意,共享逻辑块所支持的一组线程可以随在处理器内的逻辑块不同而改变。例如,考虑类似于图1的处理器架构的处理器架构,但每一处理核是其中每一流水线专用于单个线程的“双流水线”核。在这种情况下,每一处理器是专用于两个线程的共享块,对于每一处理器核,每一线程对不同。相比之下,交叉开关、L2高速缓存、存储器控制器和I/O控制器是为所有处理核的一组全部线程提供服务的共享逻辑块。

[0041] 因而,例如,如果系统告示值规定高性能,如果处理器核101_1具有两个高性能线程,则它将处于LL状态,然而,同时,如果处理器核101_2具有两个低性能线程,则它将处于ET状态。

[0042] 然后,借助于对可以被实现为利用TCMHINT指令和系统告示值的硬件设计的实施例的理解,值得讨论使用TCMHINT指令来影响应用的功率管理。在这里,再次,可以想象到的是,应用软件——而不是编译器或OS调度程序——通过TCMHINT指令对硬件直接强加提示

或功率管理指示。可以由相同的应用软件或者诸如操作系统或系统功率管理软件等的系统级软件提供系统告示107。

[0043] 图3示出第一应用流结构,其最初由串行的单个线程组成,然后扇出(fan out)至包括多个并行线程,然后,跌回到单个线程。图3的应用流结构对具有主控制线程301_1的“分批”计算应用来说是典型的,主控制线程301_1最初组织“工程”310以便由多个“工作”线程301_2到301_N(301_2、301_3、...、301_N)并行处理。尽管主线程301_1组织工程310,但主线程301_1处于高性能状态(已经由主线程前期执行的TCMHINT指令置于高性能状态)。同时,该时刻没有任何工作要做的工作线程301_2到301_N处于低性能状态(也已经由它们中的每一个的前期TCMHINT指令置于低性能状态)。

[0044] 当主控制线程301_1识别工作线程301_2到301_N的工作切换接近即将发生时,通过线程间通信进程,它引起工作线程执行为它们中的每一个指定高性能状态的它们自己各自的TCMHINT指令。在这里,比由工作线程301_2到301_N执行的实际工作的开始提前得足够多地做出线程间通信,以便考虑改变支持工作线程301_2到301_N的硬件资源的频率、时钟门控和/或电压设置所需要的任何转换时间。为了更加精确,在对工作线程执行TCMHINT指令时,对于每一线程,将新的值写到寄存器空间。根据上面提供的讨论,线程升级到较高性能状态可以触发功率管理电路改变一个或多个流水线、处理器或其他逻辑块的功率状态。

[0045] 利用为工作线程要进行的工作提前准备好的硬件,将“工程”从主控制线程301_1分派给工作线程301_2到301_N,并且各工作线程对工程开始它们的工作。与工作线程对工程开始它们的工作相称,主控制线程301_1可以执行TCMHINT指令以便将自身置于低功率状态。硬件可以据此做出反应。

[0046] 随着工作线程即将完成该工程(例如,因为多个工作线程已经表明接近完成它们的工程部分),主控制线程301_1执行另一TCMHINT指令以便将自身置于高性能状态。再次,比由工作线程301_2到301_N实际完成工程充分提前地执行TCMHINT指令,以使得如果环境允许(例如,根据图2A和图2B的状态改变算法)则功率管理控制逻辑120可以提高支持主控制线程301_1的任何逻辑块的性能状态。

[0047] 在工作线程301_2到301_N已经完成它们的工作时,如果合适的话,主控制线程301_1借助于其已经处于较高性能状态的底层硬件处理结果。然后,工作线程301_2到301_N中的每一个执行另一TCMHINT指令,以便将它们的性能状态复位到低性能状态。如果合适/根据需要,它们的底层支持逻辑块重新调整它们自身。

[0048] 在上面的示例中,将工作线程301_2到301_N置于高性能状态以执行工程。回忆起在一个实施例中,可以给线程指派三种性能状态中的一种:1)高;2)中;3)低。工作线程301_2到301_N的大尺度并行操作实际上是其中可以使用中等线程功率状态的良好示例。具体地,如上面的示例中所讨论的,如果将所有工作线程301_2到301_N都置于高性能状态以便执行工程,尽管可以在最小时间量内完成工程,但处理器的功率消耗可能超出应用开发者的希望。

[0049] 因而,在备选的实施例中,代替使用TCMHINT指令来将工作线程301_2到301_N中的每一个置于高性能状态,使用TCMHINT指令来将工作线程301_2到301_N中的每一个置于中等性能状态。在这里,由于该工程是潜在的大规模并行操作,应用开发者可以选择消耗更多时间来完成工程,以便在处理工程的同时保持处理器的功率消耗受到限制。该选择由开发

者取决于应用的情况而进行。

[0050] 例如,“关键任务”工程可以利用高性能工作线程设置,而“较不重要”工程可以利用中等性能工作线程设置。此外,如图2A和图2B可见,系统告示设置准许可以由应用开发者或系统管理员管理的性能与功率消耗的折衷的更进一步的程度。

[0051] 注意,尽管可能执行在其自己的处理核或流水线上执行各线程中的每一个,但不存在这样做的要求。如果在相同的处理核或相同的流水线上执行多个工作线程,则由于工作线程同等地且一致地调整它们的性能状态,处理核或流水线的性能状态将仍然类似地/同等地受到影响。如果在主控制线程在其上执行的相同的处理核或流水线上执行一个或多个工作线程,则如果系统告示规定高性能则处理核或流水线将保持在高性能状态,或者如果系统告示规定低性能则保持在低性能状态,这是因为主控制线程和工作线程切换到彼此的反状态。

[0052] 在另一示例中,如图4中所观察的,观察到另一多线程应用。再次,线程401_1可以被看作是控制线程,且再次,线程401_2到401_N(401_2、401_3、...、401_N)可以被看作是工作线程。然而,从该应用在其上执行的系统所耦合到的网络接收由工作线程401_2到401_N处理的“工程”。控制线程401_1是处理来自网络的传入流量的线程。因而,在工作线程可以处理从网络接收到的工程之前,首先由控制线程401_1从网络接收“工程”。

[0053] 在这种情况下,控制线程401_1可以(例如,由在线程的开始执行的单个TCMHINT指令)永久置于高性能状态。因而,一旦该工程到达,控制线程401_1就可以尽可能快地接收和处理工程。一旦工程从网络到达,一部分处理就可以包括使用线程间通信来“唤醒”工作线程301_2到301_N,引起它们执行将它们的当前低功率空闲设置改变成较高功率设置的TCMHINT指令。再次取决于总体环境(工程的重要程度、功率预算等等),较高功率设置可以是中等性能状态或高性能状态。

[0054] 注意,从控制线程到工作线程的线程间通信无非就是对工作线程的函数调用,其中所调用的函数包括带有所期望的线程功率状态改变的TCMHINT指令。

[0055] 图5示出通用处理核500,相信,该通用处理核500描述多种不同类型的处理核架构,例如复杂指令集(CISC)、精简指令集(RISC)和超长指令字(VLIW)。图5的通用处理核500包括具有以下的指令执行流水线:1)(例如,从高速缓存或存储器)获取指令的获取单元503;2)解码指令的解码单元504;3)确定向执行单元506的指令发布的时序和/或次序的调度单元505(值得注意的是该调度程序是可选的);4)执行指令的执行单元506;5)表示指令的成功完成的引退单元(retirement unit)507。值得注意的是,处理核可以包括或不包括部分地或全部地控制执行单元506的微操作的微代码508。可以借助于在执行单元506内的一个或多个执行单元实现前述讨论中提及的指令执行资源/逻辑。

[0056] 在这里,具体地,执行在此描述的TCMHINT指令的(多个)执行单元可以耦合到由功率管理控制逻辑读取的寄存器空间,以便影响在包括图5的处理核的处理器内的各种逻辑块的性能状态设置。在执行TCMHINT指令时,将该指令中包括的性能状态信息(例如,形式“TCMHINT X”的指令格式中的“X”)写入到这样的寄存器空间。

[0057] 以上讨论教导的进程可利用程序代码执行,诸如机器可执行指令,其导致机器执行这些指令以实现某些功能。在这种背景下,“机器”可以是将中间形式(或抽象)指令转换成处理器专用指令(例如,抽象执行环境,诸如“虚拟机”(例如,Java虚拟机)解释器、公共语

言运行时、高级语言虚拟机等)的机器和/或设置在被设计成执行指令的半导体芯片中的电子电路(例如以晶体管实现的“逻辑电路”),诸如通用处理器和/或专用处理器。以上讨论示教的进程还可通过设计成执行进程(或其一部分)的电子电路来执行(替换机器或与机器组合),而不执行程序代码。

[0058] 相信以上讨论示教的进程也可按各种软件部署框架(例如微软公司的.NET、Mono、Java,甲骨文公司的Fusion等)支持的各种面向对象或非面向对象的计算机编程语言(例如,Java、C#、VB、Python、C、C++、J#、APL、Cobol、Fortran、Pascal、Perl等)在源级程序代码中描述。源级程序代码可被转换成中间形式的程序代码(诸如Java字节代码、微软中间语言等),可将其理解为抽象执行环境(例如,Java虚拟机、公共语言运行时、高级语言虚拟机、解释器等),或者可直接编译成对象代码。

[0059] 根据各种方法,通过1)编译中间形式的程序代码(例如在运行时(例如JIT编译器)),2)解释中间形式的程序代码或3)在运行时编译中间形式的程序代码和解释中间形式的程序代码的组合,抽象执行环境可将中间形式的程序代码转换成处理器专用代码。抽象执行环境可运行在各种操作系统上(诸如UNIX、LINUX、包括Windows族的微软操作系统、包括MacOS X的苹果计算机操作系统、Sun/Solaris、OS/2、Novell等)。

[0060] 制品可用于存储程序代码。存储程序代码的制品可体现为但不限于一个或多个存储器(例如,一个或多个闪存、随机存取存储器(静态、动态或其它))、光盘、CD-ROM、DVD ROM、EPROM、EEPROM、磁或光卡或适于存储电子指令的其它类型的机器可读介质。还可从远程计算机(例如,服务器)将程序代码作为体现在传播介质中的数据信号(例如,经由通信链路(例如,网络连接))下载至请求计算机(例如,客户机)。

[0061] 在上述说明书中,已参考本发明具体示例实施例描述了本发明。然而,显然可对这些实施例作出各种修改和改变,而不背离如所附权利要求所述的本发明的更宽泛精神和范围。

100

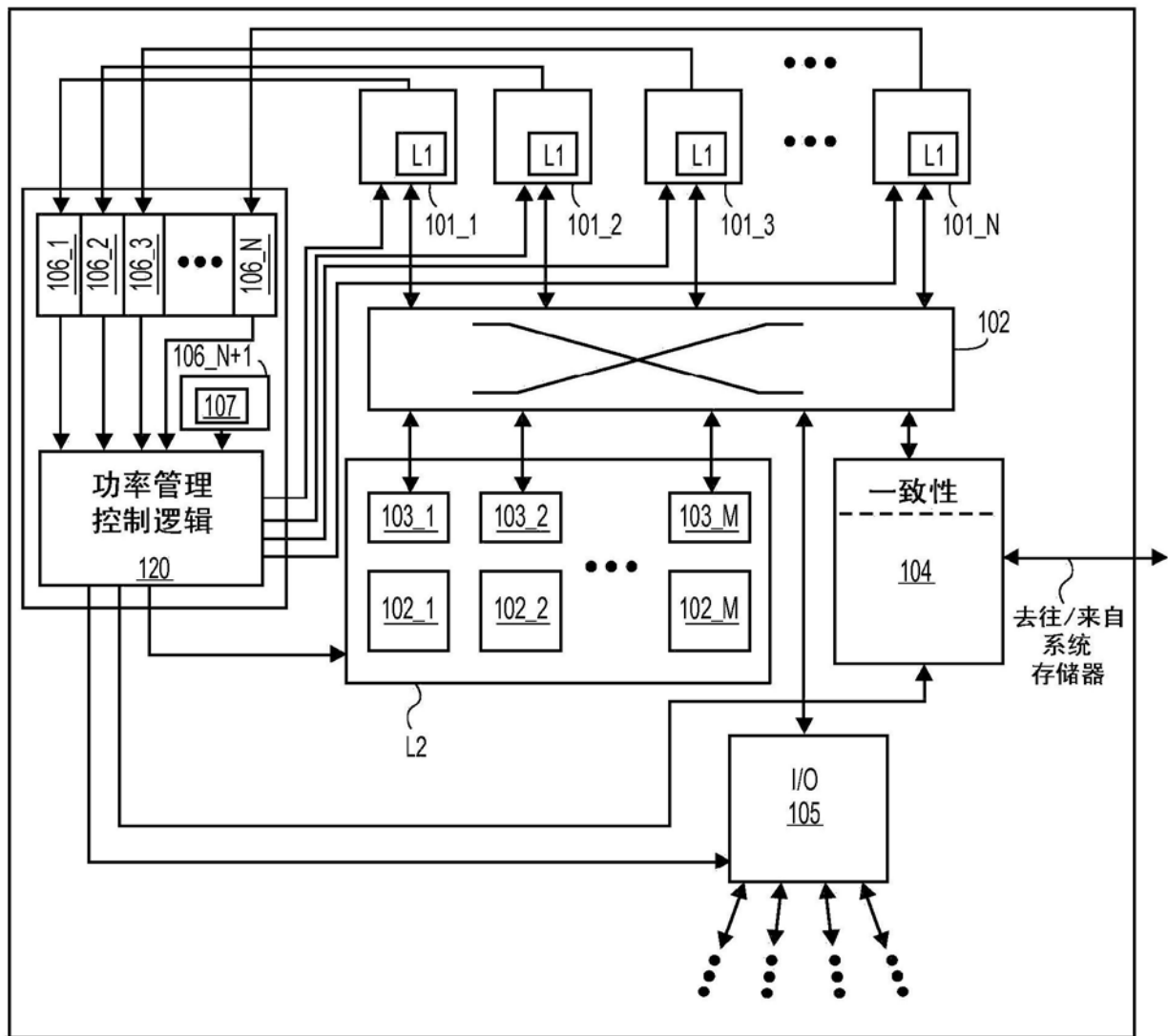


图1

专用于单个线程的逻辑块

线程性能状态	系统告示设置	
	高功率性能	低功率性能
高	LL	HT
中	HT	ET
低	ET	PO

211

213

图2A

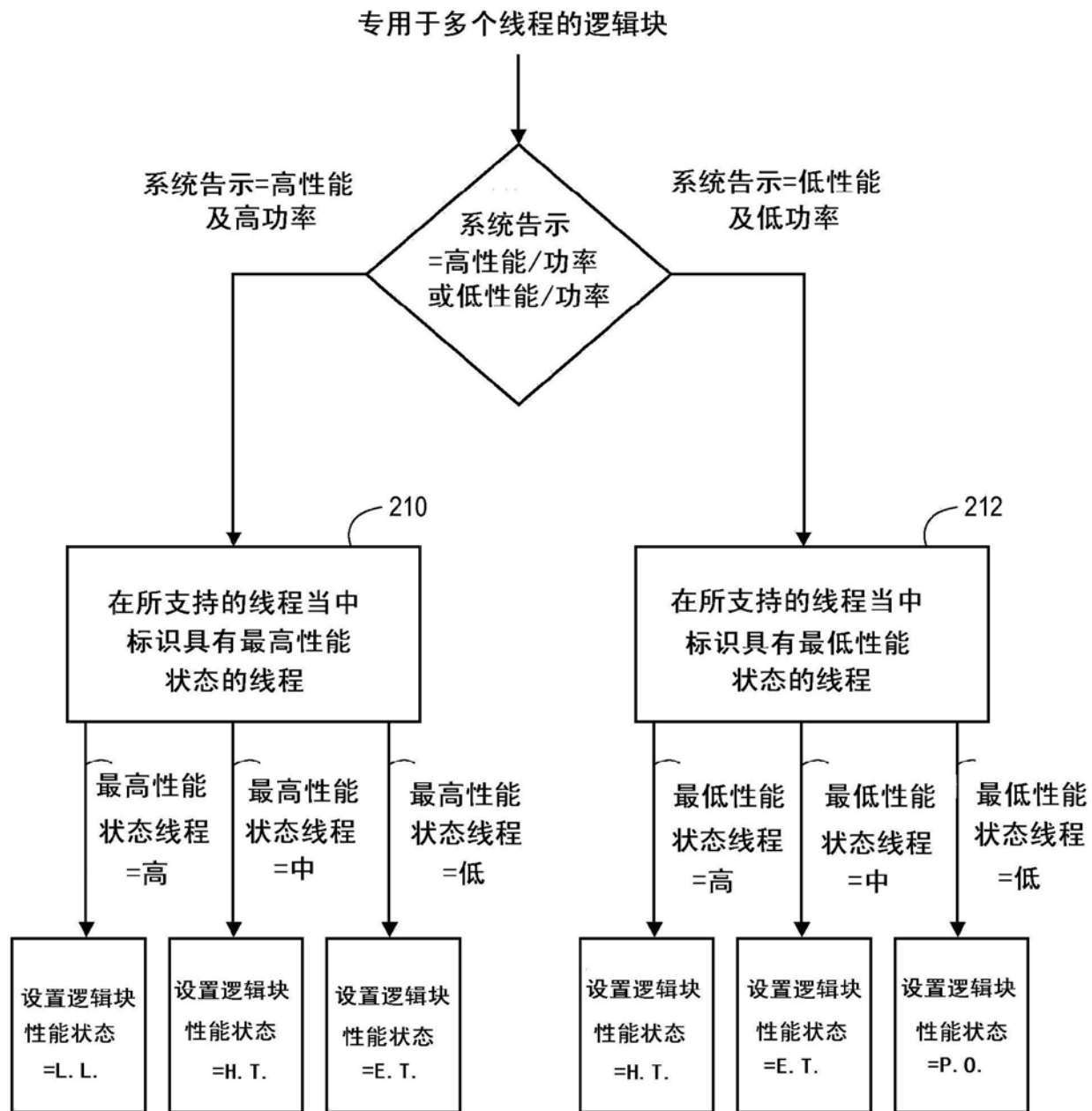


图2B

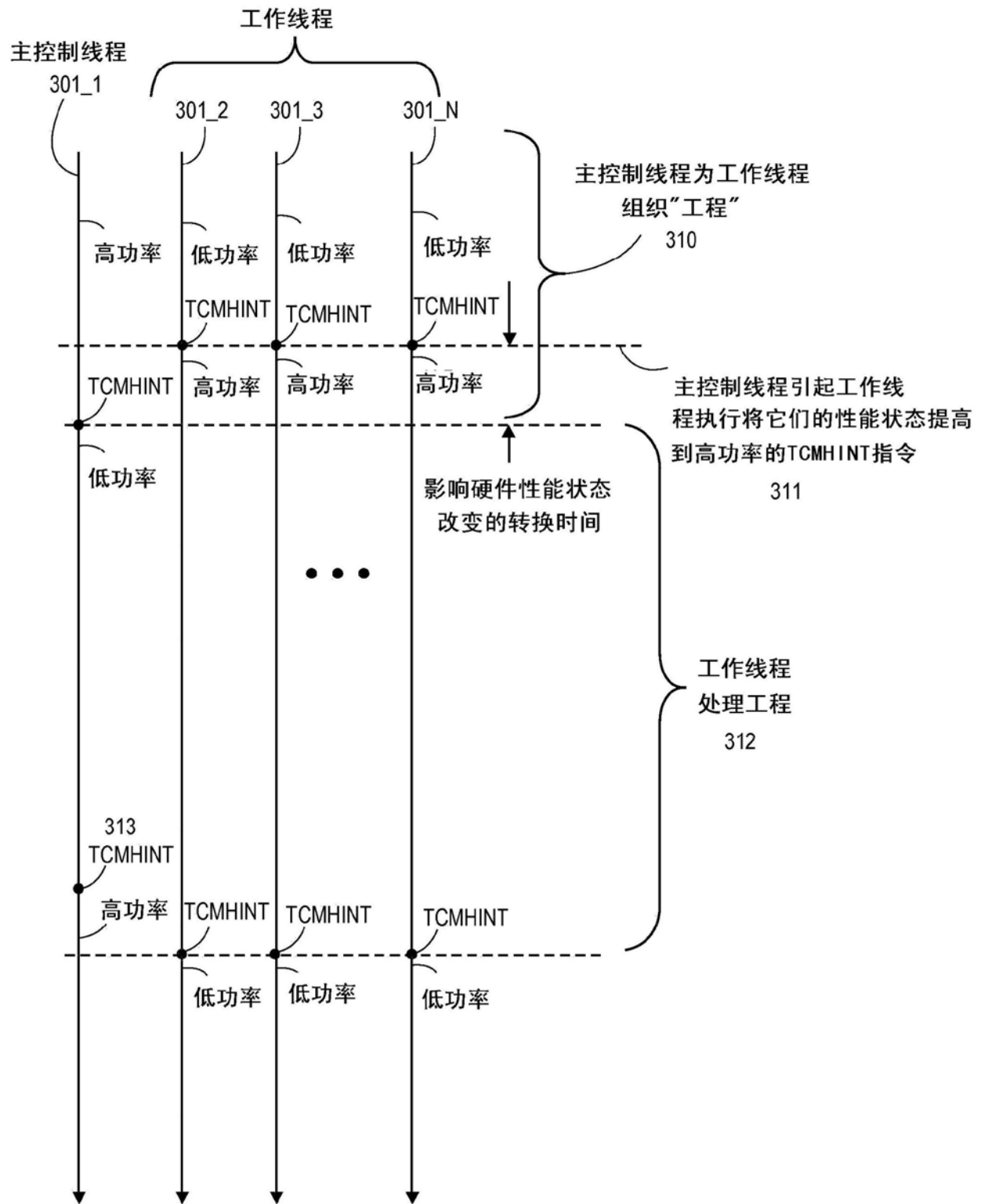


图3

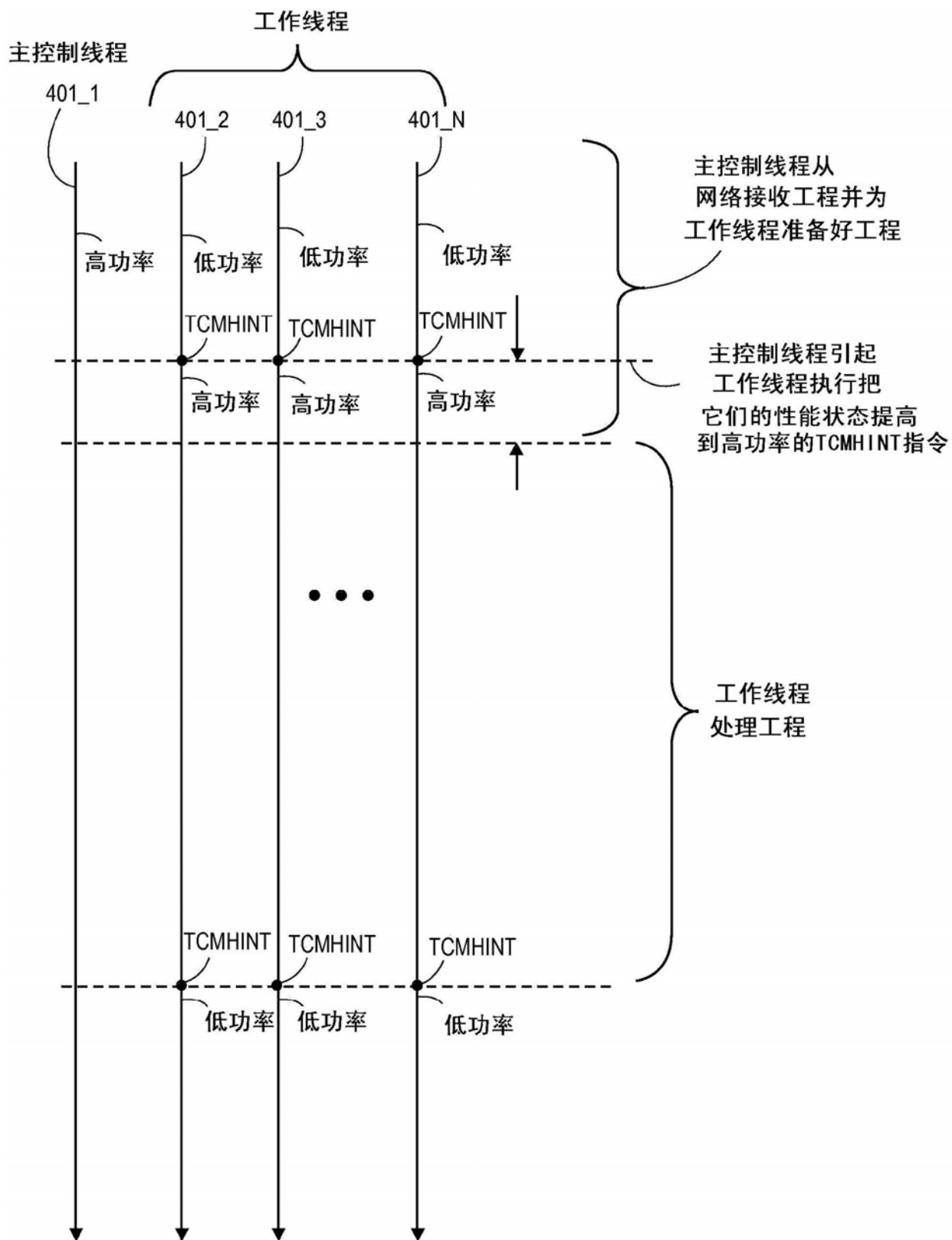


图4

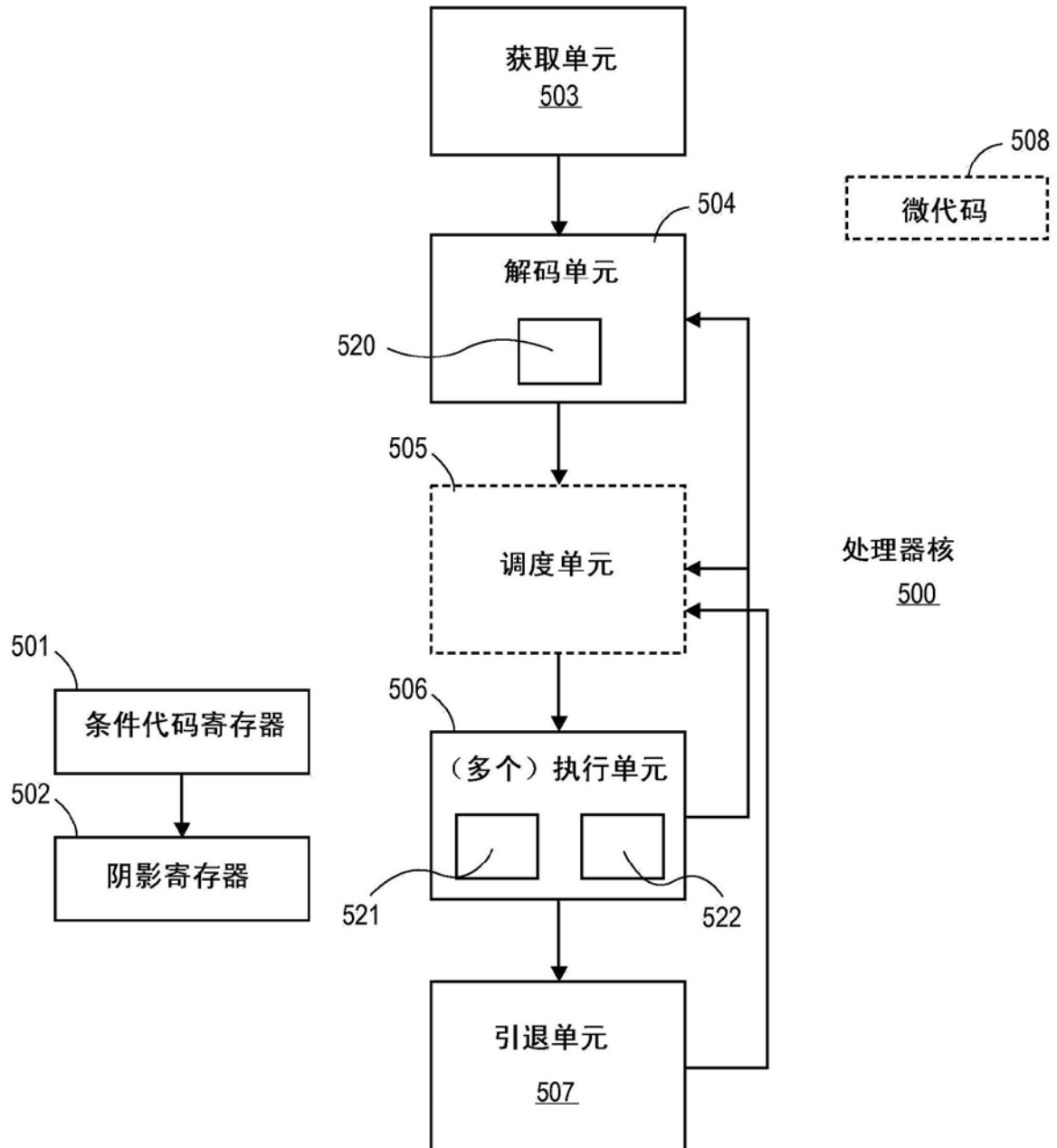


图5