(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 106537329 B
(45)授权公告日 2020.03.06

(21)申请号 201580036289.5

(22)申请日 2015.05.19

(65)同一申请的已公布的文献号
申请公布号 CN 106537329 A

(43)申请公布日 2017.03.22

(30)优先权数据
1412082.8 2014.07.08 GB

(85)PCT国际申请进入国家阶段日
2016.12.30

(86)PCT国际申请的申请数据
PCT/GB2015/051469 2015.05.19

(87)PCT国际申请的公布数据
WO2016/005720 EN 2016.01.14

(73)专利权人 ARM 有限公司
地址 英国剑桥

(72)发明人 大卫·汉纳·曼塞尔
蒂莫西·霍尔罗伊德·哥劳特

(74)专利代理机构 北京东方亿思知识产权代理有限责任公司 11258
代理人 林强

(51)Int.Cl.
G06F 9/30(2006.01)
G06F 9/38(2006.01)
G06F 9/52(2006.01)

(56)对比文件
US 2013290967 A1,2013.10.31,
US 2010083266 A1,2010.04.01,
CN 101854302 A,2010.10.06,

审查员 赖女女

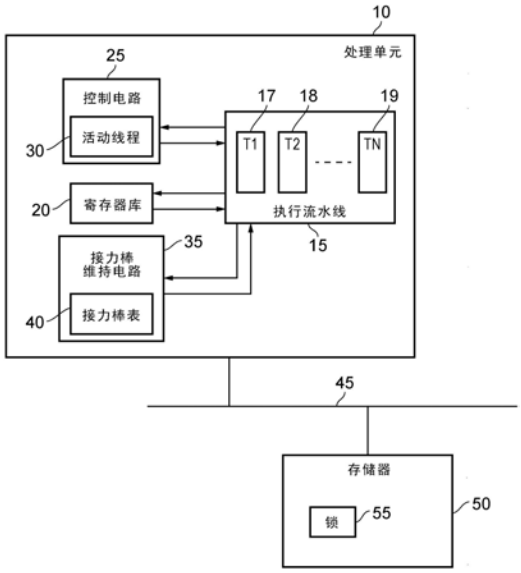
权利要求书4页 说明书13页 附图12页

(54)发明名称

用于为多线程执行锁保护处理操作的数据处理装置及方法

(57)摘要

提供了用于执行多个线程的数据处理装置及方法。处理电路执行多个线程所需的处理操作,处理操作包括与锁相关联的锁保护处理操作,其中在处理电路执行锁保护处理操作之前,需要获取锁。接力棒维持电路用于与多个线程相关联地维持接力棒,接力棒形成锁的代理,并且接力棒维持电路被配置成在多个线程之间分配接力棒。一旦已为线程中的一个获得锁,处理电路在释放锁之前通过处理电路和接力棒维持电路之间的通信针对多个线程执行锁保护处理操作,接力棒维持电路在多线程中标识当前线程,其中,通过将接力棒分配给该当前线程来为其执行锁保护处理操作。因此,可以将接力棒从一个线程传递到下一个线程,而不需要释放和重新获取锁。当跨多线程执行锁保护处理操作时,这提供了显著的性能改进。



CN 106537329 B

1.一种用于执行多个线程的数据处理装置,所述数据处理装置经由互连耦合至存储器,并且包括:

处理电路,被配置为执行所述多个线程所需的处理操作,所述处理操作包括与锁相关联的锁保护处理操作,所述锁需要在所述处理电路执行所述锁保护处理操作之前在存储器中被获取;

接力棒维持电路,被配置为与所述多个线程相关联地维持接力棒,其中所述接力棒形成所述锁的代理,并且所述接力棒维持电路被配置为在所述多个线程之间分配接力棒;以及

存储结构,与存储器分开并且与所述多个线程相关联,并且在所述存储结构中,所述接力棒维持电路被配置为维持所述接力棒;

所述处理电路被配置为与所述接力棒维持电路通信,使得一旦已为所述多个线程中的一个线程在存储器中获取到所述锁,则所述处理电路在在存储器中释放所述锁之前为所述多个线程中的多线程执行所述锁保护处理操作,所述接力棒维持电路被配置为在所述多线程中标识当前线程,其中,通过将所述接力棒分配给所述当前线程来为所述当前线程执行所述锁保护处理操作,

其中,当所述处理电路已为线程执行完所述锁保护处理操作时,所述处理电路被配置为向所述接力棒维持电路发出传递接力棒请求,所述接力棒维持电路响应于所述传递接力棒请求来确定在所述多线程中是否存在仍需要执行所述锁保护处理操作的任何剩余线程,并且如果存在,则将所述接力棒分配给所述剩余线程中的一个线程;并且

其中,每个线程包括指令序列,并且对于需要执行所述锁保护处理操作的线程,所述线程包括传递接力棒指令,所述传递接力棒指令在执行完所述线程的所述锁保护处理操作时被所述处理电路执行,并使得所述处理电路向所述接力棒维持电路发出所述传递接力棒请求。

2.根据权利要求1所述的数据处理装置,其中所述接力棒维持电路被配置为将所述接力棒一次分配给一个线程。

3.根据权利要求1或2所述的数据处理装置,其中,对于需要执行所述锁保护处理操作的需求线程,所述处理电路被配置为请求所述接力棒维持电路将所述接力棒分配给所述需求线程,如果所述接力棒当前未被分配给另一线程,所述接力棒维持电路被配置为将所述接力棒分配给所述需求线程并使所述处理电路获取所述锁。

4.根据权利要求3所述的数据处理装置,其中当所述处理电路请求所述接力棒维持电路将所述接力棒分配给需要执行所述锁保护处理操作的需求线程时,如果所述接力棒当前被分配给另一线程,所述接力棒维持电路被配置为与所述接力棒相关联地维持所述需求线程正在等待所述接力棒的指示,并且使所述处理电路在等待所述接力棒分配期间停止执行所述需求线程。

5.根据权利要求3所述的数据处理装置,其中每个线程包括指令序列,并且对于需要执行所述锁保护处理操作的需求线程,所述需求线程包括当由所述处理电路执行时使所述处理电路请求所述接力棒维持电路将所述接力棒分配给所述需求线程的获得接力棒指令。

6.根据权利要求5所述的数据处理装置,其中所述获得接力棒指令的执行使得控制数据被存储在寄存器中,以指示所述接力棒维持电路是否需要所述处理电路获取所述锁。

7.根据权利要求5所述的数据处理装置，其中，所述指令序列包括锁获取代码，并且如果所述接力棒维持电路不需要所述处理电路获取所述锁，则所述获得接力棒指令的执行使得处理跳转越过所述锁获取代码。

8.根据权利要求1所述的数据处理装置，其中，如果所述接力棒维持电路确定了在所述多线程中不存在仍需要执行所述锁保护处理操作的剩余线程，则所述接力棒维持电路被配置为将所述接力棒解除分配并使得所述处理电路释放所述锁。

9.根据权利要求1所述的数据处理装置，其中，所述传递接力棒指令的执行使得控制数据被存储在寄存器中，以指示所述接力棒维持电路是否需要所述处理电路释放所述锁。

10.根据权利要求1所述的数据处理装置，其中，所述指令序列包括锁释放代码，并且如果所述接力棒维持电路不需要所述处理电路释放所述锁，则所述传递接力棒指令的执行使得处理跳转越过所述锁释放代码。

11.根据权利要求1所述的数据处理装置，其中，所述装置被配置为通过执行原子操作来获取所述锁。

12.根据权利要求1所述的数据处理装置，其中，所述存储结构包括所述接力棒的条目，所述条目包括对所述接力棒正在为其形成代理的所述锁进行标识的锁标识符字段，以及对在所述多线程中仍需要执行所述锁保护处理操作的任何剩余线程进行标识的剩余线程字段。

13.根据权利要求12所述的数据处理装置，其中所述条目还包括对当前被分配以接力棒的线程进行标识的活动线程字段。

14.根据权利要求1、12、和13中任一项所述的数据处理装置，其中，所述处理操作包括多个锁保护处理操作，所述多个锁保护处理操作中的至少一些具有与其相关联的不同锁，并且所述接力棒维持电路被配置为与所述多个线程相关联地维持多个接力棒，其中每个接力棒是所述锁中一个锁的代理。

15.根据从属于权利要求12或权利要求13时的权利要求14所述的数据处理装置，其中所述存储结构包括多个条目，其中每个条目与不同接力棒相关联。

16.根据权利要求1所述的数据处理装置，其中所述数据处理装置包括一个或多个接力棒域，并且对于每个接力棒域，所述接力棒维持电路被配置为维持单独的接力棒，以在所述接力棒域中形成所述锁的代理。

17.根据权利要求1或16所述的数据处理装置，其中所述多个线程形成线程组，并且所述处理电路还被配置为执行至少一个另外的线程组中的线程所需的处理操作，其中，所述至少一个另外的线程组中的所述线程也需要执行锁保护处理操作。

18.根据从属于权利要求16时的权利要求17所述的数据处理装置，其中一个接力棒域包括多个线程组，使得所述接力棒维持电路被配置为针对所述多个线程组维持同一接力棒。

19.根据从属于权利要求16时的权利要求17所述的数据处理装置，其中，每个线程组在不同的接力棒域中，使得所述接力棒维持电路被配置为与每个线程组相关联地维持单独的接力棒。

20.根据权利要求18至19中任一项所述的数据处理装置，其中对于至少一个线程组，与为所述线程组的每个线程执行的公共程序并行地处理所述线程组内的所述多个线程。

21.一种在数据处理装置内执行多个线程的方法,每个线程包括指令序列,包括:

在所述数据处理装置的处理电路内执行所述多个线程所需的处理操作,所述处理操作包括与锁相关联的锁保护处理操作,所述锁需要在所述处理电路执行所述锁保护处理操作之前在存储器中被获取;

使用接力棒维持电路来在与存储器分开且与所述多个线程相关联的存储结构中与所述多个线程相关联地维持接力棒,其中所述接力棒形成所述锁的代理,并且在所述多个线程之间分配所述接力棒;并且

一旦已经为所述多个线程中的一个线程在存储器中获取到了所述锁,则在在存储器中释放所述锁之前执行所述多个线程中的多线程的所述锁保护处理操作,并且在所述多线程中标识当前线程,其中,通过将所述接力棒分配给所述当前线程来为所述当前线程执行所述锁保护处理操作;

当已为线程执行完所述锁保护处理操作时,向所述接力棒维持电路发出传递接力棒请求,所述接力棒维持电路响应于所述传递接力棒请求来确定在所述多线程中是否存在仍需要执行所述锁保护处理操作的任何剩余线程,并且如果存在,则将所述接力棒分配给所述剩余线程中的一个线程;并且

对于需要执行所述锁保护处理操作的线程,所述线程包括传递接力棒指令,所述传递接力棒指令在执行完所述线程的所述锁保护处理操作时被执行,并使得向所述接力棒维持电路发出所述传递接力棒请求。

22.一种用于执行多个线程的数据处理装置,包括:

处理装置,用于执行所述多个线程所需的处理操作,所述处理操作包括与锁相关联的锁保护处理操作,所述锁需要在所述处理装置执行所述锁保护处理操作之前在存储器中被获取;

接力棒维持装置,用于在与存储器分开且与所述多个线程相关联的存储结构中与所述多个线程相关联地维持接力棒,所述接力棒形成所述锁的代理,并且所述接力棒维持装置被配置为在所述多个线程之间分配接力棒;以及

所述处理装置用于与所述接力棒维持装置通信,使得一旦已在存储器中获取到所述多个线程中的一个线程的所述锁,则所述处理装置在所述锁在存储器中被释放之前执行所述多个线程中的多线程的所述锁保护处理操作,所述接力棒维持装置用于在所述多线程中标识当前线程,其中,通过将所述接力棒分配给所述当前线程来为所述当前线程执行所述锁保护处理操作;

其中,当所述处理装置已为线程执行完所述锁保护处理操作时,所述处理装置被配置为向所述接力棒维持装置发出传递接力棒请求,所述接力棒维持装置响应于所述传递接力棒请求来确定在所述多线程中是否存在仍需要执行所述锁保护处理操作的任何剩余线程,并且如果存在,则将所述接力棒分配给所述剩余线程中的一个线程;并且

其中,每个线程包括指令序列,并且对于需要执行所述锁保护处理操作的线程,所述线程包括传递接力棒指令,所述传递接力棒指令在执行完所述线程的所述锁保护处理操作时被所述处理装置执行,并使得所述处理装置向所述接力棒维持装置发出所述传递接力棒请求。

23.一种非暂态计算机可读存储介质,存储有指令,所述指令在被处理器执行时,使得

所述处理器执行如权利要求21所述的方法。

用于为多线程执行锁保护处理操作的数据处理装置及方法

技术领域

[0001] 本发明涉及一种用于对多线程执行锁保护处理操作的数据处理装置和方法。

背景技术

[0002] 提供一种被配置为执行多个线程的数据处理装置是已知的。每个线程通常需要执行一系列处理操作，并且每个线程通常包括对那些处理操作进行定义的指令序列。

[0003] 还已知使用锁作为同步机制，当存在多线程执行时，用于限制对数据处理系统内的相关资源(例如，共享数据结构)的访问。如果处理操作需要访问由锁保护的资源，则该处理操作在本文将被称为锁保护处理操作，并且在针对特定线程执行这种锁保护处理操作之前，将需要确保获取了锁。锁可以采取各种形式(并且可以应用本发明而不考虑锁的基本形式)，但是一种常用形式是存储在存储器中的特定地址处的二进制值。为了获取锁，处理电路在存储器中的锁地址处查找值，并且如果其处于指示锁可用的第一值(例如，零)，则通过将该数据值设置为指示锁已被获取的第二值(例如，一)来获取锁。在为特定线程获取锁的同时，寻求获取锁的任何其它线程将在从锁地址读取时确定该锁当前处于第二值，并且相应地已被获取。然后，不针对其他线程执行锁保护处理操作，直到该线程已被释放，并且随后由其他等待的线程获取。

[0004] 相应地，当多线程包括由同一锁保护的锁保护处理操作时，则将仅能一次针对一个线程执行该锁保护处理操作。锁保护处理操作可以是由每个线程执行的相同操作(例如每个线程正执行相同的程序代码)，或者可以是不同线程中的不同处理操作(例如，其中一个线程希望执行处理操作以访问共享资源以便将值添加到该共享资源中的列表中，而另一个线程希望执行处理操作以从该共享资源中的列表中删除值)。

[0005] 指定锁保护处理操作的一个或多个指令在本文将被称为关键代码段，相应地，关键代码段是多线程程序的一部分，其一次只能由一个线程安全地执行，因为例如它会更新共享数据结构。当在特定线程内遇到关键代码段时，首先获取锁(其防止任何其它线程获取到锁，直到它被释放)，并且在完成关键代码段之后，释放锁。这种方法允许任意复杂度的关键代码段被支持锁机制的任何硬件支持。

[0006] 然而，为了确保正确的操作，锁机制必须包括线程之间的通信和同步，使得所有线程在任何一个时间都知道哪个线程拥有锁，并且此外，每当获取和释放锁时，必须调用该机制(即，每次每个线程想要执行锁保护处理操作时)。通常这通过使用原子操作来实现，其确保每次只有单个线程可以获取锁。因此，锁的获取是相对耗时的过程，涉及多个指令的执行以及对存储器中的锁地址的读和写访问。因此，期望提供在数据处理设备上执行多线程时用于处理锁保护处理操作的更高效的机制。

发明内容

[0007] 从第一方面看，本发明提供了一种用于执行多个线程的数据处理装置，包括：处理电路，被配置为执行所述多个线程所需的处理操作，所述处理操作包括与锁相关联的锁保

护处理操作，锁需要在处理电路执行锁保护处理操作之前被获取；以及接力棒维持电路，被配置为与多个线程相关联地维持接力棒，接力棒形成锁的代理，并且接力棒维持电路被配置为在多个线程之间分配接力棒；处理电路被配置为与接力棒维持电路进行通信，使得一旦已为多个线程中的一个线程获取到锁，则处理电路在锁被释放之前针对多个线程中的多线程执行锁保护处理操作，接力棒维持电路被配置为在多线程中标识当前线程，其中，通过将接力棒分配给该当前线程来为其执行锁保护处理操作。

[0008] 根据本发明，提供接力棒维持电路以与多个线程相关联地维持接力棒。接力棒形成锁的代理，并且接力棒维持电路在线程之间分配接力棒。在一个实施例中，接力棒维持电路以下述方式布置：将接力棒一次仅分配给一个线程。当试图针对多个线程执行锁保护处理操作时，本发明的接力棒机制确保了只需获取和释放锁一次，而不是必须为每个单独线程分别获取和释放锁。具体地，一旦已经为线程中的第一线程获取了锁，则在多个线程之间传递接力棒，以针对那些线程中的每一个执行锁保护处理操作，而同时始终保持着获取锁。一旦已针对多线程中的所有线程执行了锁保护处理操作，则释放锁。在一个实施例中，该机制用于使得在接下来释放锁之前按顺序针对每个线程执行锁保护处理操作。

[0009] 这种方法通过确保一次只有一个线程执行锁保护处理操作来确保锁保护处理操作的正确处理，同时减少与获取和释放锁相关联的开销。因此，这可以在执行多个线程的数据处理系统内(特别是在多个线程包括大量锁保护处理操作的情况下)产生显著的性能改进。

[0010] 接力棒与锁相关联。因此，在使用多个锁的实施例中，单独的接力棒将与每个锁相关联。

[0011] 存在处理电路可以与接力棒维持电路通信以实现上述讨论的功能的多种方式。在一个实施例中，对于需要执行锁保护处理操作的需求线程，处理电路被配置为请求接力棒维持电路将接力棒分配给该需求线程，如果接力棒当前未被分配给另一线程，接力棒维持电路被配置为将接力棒分配给需求线程并且使处理电路获取锁。因此，如果锁尚未被获取，则可以将接力棒分配给寻求执行锁保护处理操作的线程，然后使处理电路在执行锁保护处理操作之前获取锁。

[0012] 在一个实施例中，接力棒可以永久存在，因此在上述情况下，“分配”过程仅包括以任何适当的方式标识下述情形：已将该接力棒分配给需求线程。在替代实施例中(其避免任何时候都需要针对每个单个锁的接力棒的存在)，可以根据需要而创建和终止接力棒，因此在上述情况下，“分配过程”包括创建接力棒然后以任何适当的方式标识下述情形：已将该接力棒分配给需求线程。

[0013] 此外，在一个实施例中，如果在请求接力棒维持电路将接力棒分配给需求线程时，确定了接力棒当前被分配给另一线程，则这指示了已经在针对该另一线程执行锁保护处理操作(或者该另一线程在获取锁的过程中，使得可以执行锁保护处理操作)，并且在该情况下，接力棒维持电路与接力棒相关联地维持需求(后续)线程正在等待接力棒的指示，并且使处理电路在等待接力棒分配期间停止执行需求线程。

[0014] 通过这种方法，在针对当前已被分配了接力棒的线程执行锁保护处理操作的同时，可以使得一个或多个线程停止而等待接力棒的分配。一旦当前被分配了接力棒的线程完成执行锁保护处理操作，则可以将接力棒重新分配给停止的线程中的一个，以使得该线

程能够继续运行并且执行锁保护处理操作，而无需释放和重新获取锁。

[0015] 存在多种方式，其中处理电路可以请求接力棒维持电路将接力棒分配给线程。在一个实施例中，每个线程包括指令序列，并且对于需要执行锁保护处理操作的需求线程，需求线程包括当由处理电路执行时使处理电路请求接力棒维持电路将接力棒分配给需求线程的获得接力棒指令。因此，在此实施例中，将明确的指令添加到指令序列中以触发针对接力棒维持电路的接力棒的请求。

[0016] 获得接力棒指令的执行可以以各种方式实现。例如，在一个实施例中，获得接力棒指令的执行使得控制数据被存储在寄存器中，以指示接力棒维持电路是否需要处理电路来获取锁。在已经执行了获得接力棒指令之后，接着处理电路通常将执行一个或多个进一步指令以分析寄存器中的控制数据，以便确定是否需要获取锁。

[0017] 形成线程的指令序列通常将包括锁获取代码。在获得接力棒指令的替代实施方案中，获得接力棒指令的执行不需要将任何控制数据写入寄存器，并且取而代之的是，如果接力棒维持电路不需要处理电路获取锁，则获得接力棒指令的执行将结合所需的跳转功能以跳转越过锁获取代码。否则，获得接力棒指令的执行将不会引起这种跳转发生，并且相反，执行将继续到锁获取代码以使得锁被获取。这避免了将控制数据写入寄存器的需要，或者执行原本分析控制数据所需的一个或多个进一步指令的需要。在一个实施例中，在这种情况下跳转的地址将作为获得接力棒指令的一部分被提供，并且因此它是实现所需跳转行为的硬件和软件的组合。

[0018] 除了提供用于请求接力棒维持电路将接力棒分配给线程的机制之外，还提供了释放接力棒并允许接力棒被传递到等待线程的机制。尤其地，在一个实施例中，当处理电路已为线程执行完锁保护处理操作时，处理电路被配置为向接力棒维持电路发出传递接力棒请求，接力棒维持电路正响应于该传递接力棒请求来确定在多线程中是否存在仍需要执行锁保护处理操作的任何剩余线程，并且如果存在，则将接力棒分配给剩余线程中的一个线程。因此，在这种情况下，仅将接力棒重新分配给等待执行锁保护处理操作的另一线程，而无需释放和重新获取潜在的锁。

[0019] 在一个实施例中，如果接力棒维持电路确定了在多线程中不存在仍需要执行锁保护处理操作的剩余线程，则接力棒维持电路被配置为解除分配接力棒并使得处理电路释放锁。因此，当不存在等待执行锁保护处理操作的另外的线程时，则随后释放锁。

[0020] 在一个实施例中，接力棒可以永久存在，因此在上述情况下，“解除分配”过程仅包括以任何适当的方式标识下述情形：不存在分配给该接力棒的线程。在替代实施例中，可以根据需要而创建和终止接力棒，因此在上述情况下，“解除分配过程”包括终止(关闭)接力棒。

[0021] 存在多种方式，其中处理电路可以触发以上讨论的传递接力棒功能。在一个实施例中，每个线程包括指令序列，并且对于需要执行锁保护处理操作的线程，线程包括传递接力棒指令，该传递接力棒指令在执行完线程的锁保护处理操作时被处理电路执行，并使得处理电路向接力棒维持电路发出传递接力棒请求。

[0022] 在一个实施例中，传递接力棒指令的执行使得控制数据被存储在寄存器中，以指示接力棒维持电路是否需要处理电路释放锁。接着处理电路可以执行一个或多个进一步指令以便分析该控制数据并确定是否需要释放锁。

[0023] 在替代实施例中，直接将这些一个或多个进一步指令的功能实现到传递接力棒指令中，从而如果接力棒维持电路不需要处理电路释放锁，则传递接力棒指令的执行使处理跳转越过锁释放代码。这避免了将控制数据写入寄存器的需要，或者执行原本分析控制数据所需的一个或多个进一步指令的需要。

[0024] 在一个实施例中，锁被存储在存储器中，并且装置被配置为通过执行原子操作来获取锁。因此，存在与获取锁相关联的显著开销。

[0025] 相比之下，接力棒通常不存储在存储器中。相反，在一个实施例中数据处理装置还包括与多个线程相关联的存储结构，并且在该存储结构中，接力棒维持电路被配置为维持接力棒。因此，锁是存储在存储器中并且可能由数据处理装置内的各种组件引用的全局实体，而接力棒是维持在存储结构中并且特定地与多个线程相关联的本地特征。由于其对多个线程的局部适用性，所以不需要执行原子操作来控制对接力棒的访问。

[0026] 在某些实施例中，接力棒的局部性质是尤其有益的。例如，在一个实施例中，装置可以提供多个接力棒域，并且对于每个接力棒域，都可以本地维持单独的接力棒以在该接力棒域中形成锁的代理。因此，对于同一个锁，可以在多个接力棒域中的每一个接力棒域中本地提供不同的接力棒。

[0027] 存储结构可以采取各种形式。然而，在一个实施例中，存储结构包括接力棒的条目，条目包括对接力棒正在为其形成代理的锁进行标识的锁标识符字段，以及对在多线程中仍需要执行锁保护处理操作的任何剩余线程进行标识的剩余线程字段。锁标识符字段可以采取各种形式，只要它对该接力棒正在为其形成代理的锁进行唯一地标识即可。在一个实施例中，锁标识符字段从标识了锁在存储器中的位置的存储器地址形成。

[0028] 在一个实施例中，条目还包括对当前被分配以接力棒的线程进行标识的活动线程字段。虽然在一些实施例中，条目不需要明确地包括活动线程字段以对当前被分配接力棒的线程进行标识，但是这种活动线程字段的存在使得某些附加检查能够被执行。例如，当线程正寻求传递接力棒时，可以首先评估寻求传递接力棒的线程是否是当前该条目认为被分配以接力棒的活动线程。

[0029] 在一个实施例中，处理操作可以包括多个锁保护处理操作，锁保护处理操作中的至少一些具有与其相关联的不同锁，并且接力棒维持电路被配置为与多个线程相关联地维持多个接力棒，其中每个接力棒是锁中的一个锁的代理。因此，可以为每个不同的锁提供不同的接力棒，每个接力棒被限制用于相关联的多个线程。

[0030] 在这种实施例中，存储结构通常包括多个条目，其中每个条目与不同接力棒相关联。

[0031] 如前所述，在一个实施例中，数据处理装置包括一个或多个接力棒域，并且对于每个接力棒域，接力棒维持电路被配置为维持单独的接力棒，以在该接力棒域中形成锁的代理。因此，对于同一个锁，可以在多个接力棒域中的每一个接力棒域中提供不同的接力棒。因此锁是与所有接力棒域相关的全局实体，而接力棒本身是与具体的接力棒域相关联的局部实体。这具有的益处是，对于每个接力棒域，相关联的接力棒维持电路可以是局部范围的并且因此更简单，同时仍然通过全局锁来确保操作的正确性(因为只有在已获取锁的情况下，才可以执行锁保护处理操作)。

[0032] 在一个实施例中，多个线程形成线程组，并且处理电路还被配置为执行至少一个

另外的线程组中的线程所需的处理操作，所述至少一个另外的线程组中的线程还需要执行锁保护处理操作。

[0033] 有多种方式可以将各线程组分配给接力棒域。例如，在一个实施例中，一个接力棒域包括可以多个线程组，并且在这种情况下，接力棒维持电路将针对该多个线程组维持同一接力棒。在这种布置中，接力棒可以在一个线程组的线程之间连续地被传递，然后在下一个线程组的线程之间被传递，而无需释放和重新获取锁。

[0034] 在替代实施例中，每个线程组可以在不同的接力棒域中，使得接力棒维持电路被配置为与每个线程组相关联地维持单独的接力棒。在这种布置中，一旦接力棒域中的一个已获取到锁，则可以在该接力棒域的线程组内的线程之间传递相关联的接力棒，并且一旦该线程组中所有的需求线程都已执行完锁保护处理操作，则可以释放锁，在此处，锁可以由不同接力棒域的线程组获取。上述两种替代方法中的每一种具有不同的优点。具体地，具有单个较大的域减少了到全局锁的流量，但是具有较小的域提高了公平性(因为锁被更频繁地竞争)。

[0035] 线程组内的线程可以采取各种形式。在一个实施例中，与对线程组的每个线程执行的公共程序并行地处理线程组内的多个线程。这有效地实现了对不同的输入数据执行多次相同的代码片段。在一个具体的实施方案中，每个线程具有相关联的程序计数器，此外，数据处理装置具有通用程序计数器。数据处理装置寻求步调一致地执行线程组的各线程，并且特别地，在其相关联的程序计数器与通用程序计数器相匹配的所有线程上并行地执行指令。当到达锁保护处理操作时，将不可能继续跨多个线程并行地执行每个指令。取而代之的是，线程中的一个将成功获取锁，并且将成为获得接力棒分配的线程。然后依次连续地针对每个线程执行由锁保护处理操作指定的指令，通过使用在线程间被传递的接力棒来允许每个线程的锁保护处理操作被依次执行，而不需要释放和重新获取锁。一旦已执行了所有线程的锁保护处理操作，则可以释放该锁。在此处，然后可以继续先前描述的跨多线程步调一致地执行每个指令。这种步调一致的多线程处理可以被称为“单指令多线程(SIMT)”处理。

[0036] 考虑到先前对如何可以将接力棒域分配给线程组的讨论，在采用这种SIMT处理的一个实施例中，向每个单独的线程组分配接力棒域是有益的(因为它释放线程组去做其他事情)，但是可能不如跨线程组集合的共享接力棒域有益(因为它可能潜在地使系统中的其他线程组饥饿)。

[0037] 存在许多这种SIMT处理可以特别有用的情况。例如，在一个实施例中，这种SIMT处理可以在图形处理单元(GPU)内实现，以便允许跨多组输入数据并行地执行特定的代码片段。上述实施例的接力棒机制在这种布置中是特别有益的，因为它允许在连续执行线程组中的每个线程的锁保护处理操作的同时获取和保持锁，从而改进在下述情况中这种SIMT处理的性能：公共地执行的代码包括需要一次仅由一个线程执行的一个或多个关键代码段。

[0038] 从第二方面看，本发明提供一种在数据处理装置内执行多个线程的方法，包括：在数据处理装置的处理电路内执行多个线程所需的处理操作，所述处理操作包括与锁相关联的锁保护处理操作，锁需要在处理电路执行锁保护处理操作之前被获取；与多个线程相关联地维持接力棒，其中接力棒形成锁的代理，并且在多个线程之间分配接力棒；并且一旦已经为多个线程中的一个线程获取到了锁，则在释放锁之前执行所述多个线程中的多线程的

锁保护处理操作，并在多线程中标识当前线程，其中，通过将接力棒分配给该当前线程来为其执行锁保护处理操作。

[0039] 从第三方面看，本发明提供了一种用于执行多个线程的数据处理装置，包括：处理装置，用于执行多个线程所需的处理操作，所述处理操作包括与锁相关联的锁保护处理操作，锁需要在处理装置执行锁保护处理操作之前被获取；以及接力棒维持装置，用于与多个线程相关联地维持接力棒，接力棒形成锁的代理，并且接力棒维持装置被配置为在多个线程之间分配接力棒；处理装置，用于与接力棒维持装置通信，使得一旦已获取到多个线程中的一个线程的锁，则处理装置在锁被释放之前执行所述多个线程中的多线程的锁保护处理操作，接力棒维持装置用于在多线程中标识当前线程，其中，通过将接力棒分配给该当前线程来为其执行锁保护处理操作。

[0040] 从第四方面看，本发明提供了一种以非暂态形式存储的计算机程序产品，用于控制计算机为与根据本发明第一方面的数据处理装置相对应的程序指令提供虚拟机执行环境。

附图说明

[0041] 将仅通过示例的方式，参考附图中所示的实施例进一步描述本发明，其中：

[0042] 图1是根据一个实施例的包括数据处理装置的系统的框图；

[0043] 图2A示意性地示出了根据一个实施例的图1的接力棒表，而图2B示意性地示出了根据一个实施例的图1的活动线程列表；

[0044] 图3示意性地示出了用于获取和释放锁的已知代码序列；

[0045] 图4是示出了响应于线程执行的获得接力棒指令，由根据一个实施例的接力棒维持电路执行的步骤的流程图；

[0046] 图5是示出了响应于线程执行的传递接力棒指令，由根据一个实施例的接力棒维持电路执行的步骤的流程图；

[0047] 图6是示出了根据一个实施例在执行完获得接力棒指令之后由线程执行的步骤的流程图；

[0048] 图7是示出了根据一个实施例在执行完传递接力棒指令之后由线程执行的步骤的流程图；

[0049] 图8A示出了使用一个实施例的获得接力棒指令和传递接力棒指令的示例代码序列；

[0050] 图8B示出了使用另一实施例的获得接力棒指令和传递接力棒指令的示例代码序列；

[0051] 图9示意性地示出了根据一个实施例如何在线程组的多线程之间传递接力棒；

[0052] 图10示意性地示出了根据一个实施例的多个接力棒域的使用；以及

[0053] 图11示意性地示出了根据一个实施例的数据处理装置的虚拟机实现。

具体实施方式

[0054] 图1是根据一个实施例的包括形成数据处理装置的处理单元10的系统的框图。处理单元10包括被布置为执行多个线程17、18、19的执行流水线15。每个线程被布置为执行指

令序列，并且在一个实施例中，多个线程包括指定锁保护处理操作的至少一个关键代码段。虽然由每个线程指定的指令序列可以不同，但是在一个具体实施例中，每个线程指定相同的指令序列，因此通过示例的方式，多线程用于在多个不同的数据集上执行特定的指令序列。

[0055] 在执行线程17、18、19所需的处理操作期间，执行流水线将从寄存器库20读取数据值并将数据值写入寄存器库20，其中寄存器库20提供用于执行流水线的多个工作寄存器。控制电路25用于控制执行流水线的操作。在一个实施例中，控制电路25包括活动线程列表30，其在对在任何特定时间点多个线程17、18、19中哪些是活动线程(即，执行流水线当前正执行其指令的线程)以及哪些是非活动线程(例如，由于它们被停止以等待特定事件)进行标识。

[0056] 对于每个线程内的关键代码段，需要在这些关键代码段被执行之前获取存储器50中的锁55。锁是用作确保关键代码段一次只能由一个线程执行的机制。存在许多需要它的原因，但一个典型的原因是因为关键代码段的执行会更新共享数据结构，并且在任何特定时间点只有一个线程能够更新任何共享数据结构是十分重要的。

[0057] 锁可以采取各种形式，但是出于描述图1的目的，将假设锁55采取存储在存储器50内的特定地址(在本文称为锁地址)处的数据值的形式。为了寻求获取锁，特定线程将首先寻求从锁地址读取锁的当前值，以便确定锁是可用的还是已被获取的。如果锁具有第一值(例如，零)，这指示锁是可用的，而如果锁具有第二值(例如，一)，则这指示锁已被获取，因此是不可用的。如果锁是可用的，线程则可以发起写操作，以将锁值更新为第二数据值，这将防止接下来任何其他线程获取该锁。

[0058] 将参考图3更详细地进行讨论，为了确保只有一个线程成功地获取到锁，将获取锁所需的步骤执行为原子操作是十分重要的。例如，考虑锁的上述示例，其中零指示锁是可用的，并且一指示锁已被获取，两个线程可以基本上同时读取当前锁值，并且标识锁值为零。如果接着它们都能够将一写入锁，则它们都将认为它们已获取到锁，并且因此开始执行关键代码段。因此，确保只有一个线程可以成功地将一写入锁中是十分重要的，并且这通过执行原子操作来确保。

[0059] 锁通常是对系统内的多个组件可用的全局实体，并且因此通常存储在存储器中，在此示例中是可经由互连45访问的存储器50。因此，上述用于获取锁的过程包括经由互连45对存储器执行读和写操作，这是相对耗时的并且消耗大量能量。此外，为了获取锁而构建原子操作的需要进一步增加了与获取锁相关联的复杂度和时间。如果多线程包括由同一锁保护的关键代码段，则根据已知技术，每个线程必须独立地寻求获取锁，并且一旦完成执行相关联的关键代码段，每个线程必须释放锁。

[0060] 出于提高性能并同时仍确保锁机制完整性的目的，图1所示实施例的处理单元10包括由处理单元10内的接力棒维持电路35实现的接力棒机制。接力棒维持电路35维持与多个线程相关联的接力棒表40，并且对于由多个线程使用的每个锁，该接力棒维持电路35维持形成该锁的代理的本地接力棒。接力棒维持电路35确保在任何时间点线程17、18、19中只有一个线程被分配以特定接力棒。

[0061] 如稍后将更详细地进行描述的，当第一线程遇到由锁55保护的关键代码段时，它向接力棒维持电路35请求与该锁相关联的接力棒。假设相关锁的接力棒当前未在使用，则

通过填入接力棒表的条目来创建接力棒，然后将该接力棒分配给请求线程。然后，该线程的执行将使得锁被获取，之后可以执行关键代码段。当该线程已完成执行关键代码段时，则向接力棒维持电路发出传递接力棒请求，以允许该接力棒被传递到当前等待执行由同一锁保护的关键代码段的任何其他线程。接力棒维持电路35知道这种线程，因为这种线程已向接力棒维持电路发出过对接力棒的请求，但是该请求还未被实现，因为接力棒维持电路已确定接力棒已被分配给线程。然而，在接收到传递接力棒请求时，接力棒维持电路可以将接力棒重新分配给等待线程中的一个线程，以允许关键代码段由该另一线程执行。这可以在无需释放存储在存储器中的锁55的情况下进行，并且相反，将在整个过程中保持获取锁的状态。事实上，接力棒机制的操作与如何实现锁机制无关；通过使用接力棒机制，避免了与锁相关联的释放/重新获取步骤，而不管它们是什么。

[0062] 可以对等待执行由锁保护的关键代码段的所有待处理线程重复此过程。一旦所有线程都执行了关键代码段，则可以在接力棒表内将该接力棒无效，然后可以使得执行流水线15释放存储器50中的锁55。

[0063] 图2A示意性地示出了根据一个实施例的在接力棒表40的每个条目内提供的字段。在所示实施例中，在表内为当前活动的每个接力棒提供单独的条目。因此，当创建接力棒时，将对接力棒表内的可用条目(这将是其有效字段105被清除以标识该条目当前无效的条目)进行标识，然后填入标签字段110以标识锁。在一个实施例中，锁地址被用作标记字段，因为它能够唯一地标识锁。然后将设置有效位105以标识该条目包含有效接力棒。

[0064] 等待线程列表字段120用于保持已请求了接力棒但还未被分配接力棒的所有线程的列表。提供可选的活动线程字段125，其中可以记录当前被分配了接力棒的线程。在创建接力棒时，这将是使得接力棒被创建的请求线程。然而，在适当的时候，将从等待线程列表120中的等待线程中选择活动线程，在此处，从等待线程列表中移除所选择的线程。

[0065] 地址空间ID字段115是当使用不止一个地址空间时可以提供的可选字段，以便帮助对属于不同地址空间的接力棒进行标识。例如，在同一接力棒域中的线程在不同地址空间中运行的系统中，则需要对哪些接力棒属于哪个地址空间进行标识，并且实现这一点的简单机制是提供地址空间ID字段115，使得可以捕获与接力棒相关的特定地址空间。因此，仅在以下情况下可在接力棒表中找到匹配：对存储在接力棒表的有效条目中具有特定标签值的接力棒进行请求的线程也与字段115中的地址空间ID指示的地址空间相关。作为替代方法，在任何给定时间所有的活动线程都与同一地址空间相关联的实施例中，在上下文切换上，可以在交换线程的同时换出接力棒的内容以反映改变。

[0066] 图2B示意性地示出了根据一个实施例的图1的活动线程列表30。在此特定示例中，为每个线程提供单独的位，并将其设置为1以标识相关联的线程是活动的，并且将其设置为零以标识相关联的线程是非活动的。应理解的是，在替代实施例中可以交换一值和零值的含义。稍后将更详细地进行讨论，接力棒维持电路35可以在处理对存储在接力棒表40中的接力棒进行获取和释放的请求期间更新活动线程列表30中的信息。

[0067] 图3示意性地示出了在没有所述实施例的接力棒机制的情况下可以用于获取和释放锁的示例代码序列。具体地，图3示出了与特定线程相关联的代码序列。根据所示的代码，寄存器由字母“X”或字母“W”标识。在特定示例中，由字母X表示的寄存器是64位寄存器，并且由字母W标识的寄存器对相应64位寄存器内的32位字段进行标识。

[0068] 移动指令用于将一存储在寄存器W2中。此后，执行指令序列以便寻求原子地获取锁。首先，执行加载指令以将存储在锁地址处的锁值加载到寄存器W1中，其中锁地址存储在寄存器X0中。所示特定形式的加载指令在已读取锁值之后继续监控锁地址，并且特别地监控锁地址以检测在执行随后的存储指令(stxr指令)之前是否向该锁地址执行任何写操作。cbnz指令是比较并且若非零则跳转的指令。具体地，如果存储在寄存器W1中的值是非零的，则处理跳转返回到trylock位置。因此，如果已经将锁设置为一以指示已获取了锁，则过程将返回到trylock位置，以重复作为获取锁的第一步骤的加载指令。只有从锁地址加载的值为零时，过程才会进入存储指令。然后，存储指令寻求将存储在寄存器W2中的值(即，值一)写入锁地址，并且更新寄存器W1以标识存储是否成功。具体地，只有在由加载指令发起的对锁地址的监控指示了未对锁地址执行过中间的写入访问时，才存储成功。如果存储成功，则将零写入寄存器W1，否则将一写入寄存器W1。

[0069] 因此，如果寄存器W1中的值是非零的，则随后的cbnz指令使过程返回到trylock位置，从而指示存储未成功。这将使得原子操作被重试。只有当寄存器W1中的内容为零时，该过程才越过获取锁阶段，此后执行关键代码段。

[0070] 然后通过执行存储指令stlr以将特定寄存器WZR的内容写入锁地址，从而释放锁。在此实施例中，WZR是包括零的专用寄存器。因此，该存储过程释放锁。

[0071] 应理解的是，如果每个线程必须独立地执行该过程，则每个线程可能消耗大量时间和能量来寻求获得锁。这在线程包括大量关键代码段或者线程全都基本上同时执行同一程序并且因此基本上同时寻求获取锁的情况下尤其成问题。这通常需要在所有线程最终成功地获取锁并执行其对应的关键代码段之前，对形成用于获取锁的原子操作的一部分的加载和存储操作进行多次重试。

[0072] 图4是示出根据一个实施例的由接力棒维持电路35执行的步骤的流程图，以将接力棒实现为锁的代理，并且避免每个线程独立地寻求获取和释放锁。在一个实施例中，与具有关键代码段的每个线程相关联的代码包括指定了标签字段和目的寄存器字段Ws的获得接力棒指令。在一个实施例中，标签字段对要提供给接力棒的锁的地址进行标识，并且一旦接力棒被分配给请求线程，则接力棒维持电路填充由字段Ws标识的目的寄存器以标识是否需要获取锁。

[0073] 当特定线程(本文称为请求线程)执行获得接力棒指令时，执行流水线15联系接力棒维持电路35以在接力棒表40中执行查找(参见图4的步骤200)。具体地，对接力棒表中的每个有效条目进行评估，以确定该条目的标签字段数据110是否与由获得接力棒指令指定的标签相匹配。若存在，还考虑地址空间ID信息，以确定是否存在如前所述的匹配。

[0074] 在步骤205处，确定是否已经找到匹配。如果未找到，则在步骤210处确定在接力棒表中是否存在空闲条目。接力棒表的大小以及接力棒表内条目的数目可以根据实施例而变化，并且接力棒表中不必具有足够的空间来具有可由线程使用的针对所有锁的有效接力棒。如果没有空闲条目，则过程直接进行到步骤220，但是假设存在空闲条目，则在接力棒表中为所请求的接力棒创建条目，并且将活动线程设置为等同于请求线程。

[0075] 在步骤215之后，或者直接在步骤210之后，如果没有空闲条目，则将目的寄存器Ws设置为等于1。存在可以实现这一点的多种方式，但在一个实施例中，这包括接力棒维持电路指示执行流水线将一写入目的寄存器Ws。此后，在步骤225处，请求线程继续其执行。随后

将参考图6讨论请求线程接下来采取的步骤。

[0076] 假设在步骤205处发现匹配，这指示存在被分配了接力棒的另一个线程。因此，在步骤230处，将请求线程的详细信息添加到相关的接力棒表条目内的等待线程列表字段120。在步骤235处，例如通过指示执行流水线将零写入目的寄存器将零写入目的寄存器Ws。此外，在步骤240处，接力棒维持电路35使得由控制电路25维持的活动线程列表30中的相关位被清除，以标识请求线程现在是非活动的。因此，在步骤245处，请求线程停止，等待接力棒被传递给它。

[0077] 图5是示出当由执行流水线线针对特定线程(本文称为传递线程)发出传递请求时由接力棒维持电路执行的步骤的流程图。在一个特定实施例中，通过在由线程执行的指令序列内包括传递指令来实现此功能，使得一旦已执行完关键代码段，则执行传递指令。与获得接力棒指令一样，传递接力棒指令指定了两个字段，即标识相关锁的锁地址的标签字段和标识目的寄存器的目的寄存器字段Ws，其中根据响应于传递接力棒指令由接力棒维持电路执行的分析向目的寄存器写入控制数据值。

[0078] 在步骤250处，在执行流水线执行传递接力棒指令时，则在接力棒表40中执行查找，以确定是否已存在由传递接力棒指令标识的锁的条目。然后在步骤255处确定是否已找到匹配，如果是，则在步骤260处确定存储在相关条目的活动线程字段125中的活动线程ID是否与该传递线程的ID相匹配。假设软件已被正确地写入，则应该是这种情况，但是如果不是这种情况，则在步骤265处提出异常。在一个实施例中，未使用活动线程字段125，并省略步骤260。在替代实施例中，使用活动线程字段的活动线程检查可以合并到匹配检测步骤255内，使得如果存储在相关条目的活动线程字段125中的活动线程ID与传递线程的ID不匹配，则过程进行到步骤280。

[0079] 在步骤260之后，或者直接在步骤255之后，如果没有实现步骤260，则在步骤270处确定是否存在接力棒表条目中标识的任何等待线程。如前参考图2A所讨论的，任何这种等待线程将在相关条目的等待线程列表字段120中被标识。如果不存在等待线程，则在步骤275处通过清除相关联的有效位字段105来使接力棒表条目无效。此后，过程进行到步骤280，其中将一写入目的寄存器Ws。该步骤也在步骤255处未找到匹配之后直接被执行。此后，在步骤285处传递线程继续执行。稍后将参考图7讨论传递线程在此处执行的步骤。

[0080] 假设在步骤270处确定在接力棒表条目中存在至少一个等待线程，则在步骤290处选择等待线程中的一个线程并将其设置为活动线程。存在多种可以选择线程的方式。例如，这可以随机地进行，或者可以基于线程请求接力棒的顺序来进行，使得等待接力棒时间最长的线程先被分配到接力棒。

[0081] 在步骤290之后，在步骤295处，例如通过指示执行流水线将零写入目的寄存器将零写入目的寄存器Ws。此后，在步骤300处，将活动线程列表30中在步骤290处设置的活动线程的活动线程位置位，以标识该线程现在可以继续执行。因此，然后在步骤305处，活动线程继续处理。在此处，该活动线程现被分配以其先前通过执行获得接力棒指令请求的接力棒。仍如图5所示，在步骤310处，自传递线程成功释放接力棒之后，其还可以继续执行。

[0082] 图6是示出了由正获取接力棒的线程执行的步骤的流程图。在步骤350处，执行获得接力棒指令。参考图4，由于为其创建接力棒，这可以使得请求线程在步骤225处继续执行。可选地，线程可以最初在图4中的步骤245处停止，并且一旦其已被传递了接力棒，则最

终将在图5中的步骤305处继续进行处理。一旦线程已获取了接力棒，则在步骤355处，将确定目的寄存器Ws的内容是否等于一。如果是，则该过程将进行到步骤360，其中将使用原子操作来获取全局锁。一旦已获取到锁，则该过程将进行到步骤365，其中关键代码段被执行。然而，如果在步骤355处确定了目的寄存器Ws的内容不等于一，则将不需要获取锁(由于其先前已被获取，并且尚未被释放)并且因此该过程可以直接进行到步骤365，其中关键代码段被执行。

[0083] 图7是示出了由释放接力棒的线程执行的步骤的流程图。在步骤370处，执行传递接力棒指令。这将使得传递线程在图5的步骤285或步骤310处继续执行。在此处，将接下来在步骤375处确定目的寄存器Ws是否被设置为等于1。如果是，则将在步骤380处执行一个过程以释放全局锁，然后在步骤385处继续正常执行。然而，如果目的寄存器Ws不等于1，则不需要释放锁，则继续保持获取锁以供已被传递接力棒的另一线程使用，并且因此传递线程可以在步骤385处仅直接继续正常执行。

[0084] 图8A示意性地示出了如何修改图3的代码序列以根据先前描述的实施例将获得接力棒指令和传递接力棒指令合并。具体地，获得接力棒指令将指定X0的内容作为标签值，如前所述X0包括锁地址。在此示例中，指定目的寄存器Ws为W1。然后执行比较并且若为零则跳转(cbz)指令，其使得如果W1的内容为零，则指令流跳转到got lock位置。如前所述，如果W1的内容为零，则这指示不需要获取锁，因此可以绕过原子地获取锁所需的代码序列。相反，如果W1的内容非零，则执行直接继续到移动指令，然后继续到用于原子地获取锁的指令集。

[0085] 一旦执行完关键代码段，则执行传递接力棒指令。这再次使用寄存器X0的内容作为输入操作数，其标识了锁地址。再次指定目的寄存器为寄存器W1。此后，执行比较并且若为零则跳转指令，如果W1的内容为零，其将跳转到代码中的done release处。在图5和7的先前讨论中显而易见的是，这是锁不需要被释放的情形，并且因此可以绕过用于释放锁的存储指令。

[0086] 图8B示出了获得接力棒指令和传递接力棒指令的替代实施方式，其中将后续的cbz指令的功能有效地并入获得接力棒指令和传递接力棒指令。因此，不仅不需要执行后续的cbz指令，而且不需要在寄存器W1中捕获中间结果，因此避免了对寄存器库的任何访问。取而代之的是，获得接力棒指令仅取X0的值作为输入操作数，并且执行所有所需的处理以确定是否需要获取锁，或者该过程是否可以直接跳转到gotlock位置。因此，不需要执行图4的步骤220和235，并且将图6中用于确定是否要获取锁的功能并入到获得接力棒指令的执行内。

[0087] 类似地，传递接力棒指令可以包括跳转功能，再次使用X0的内容作为输入操作数，并且根据接力棒维持电路执行的分析选择性地跳转到代码中的donerelease处。参考图5，因此不再需要步骤280和295，并且将图7中用于确定是否要释放锁的功能并入到传递接力棒指令的执行内。

[0088] 图9示意性地示出了如何在多线程之间传递接力棒，以避免锁的重复获取和释放。在此示例中，线程组由四个线程405、410、415、420组成，所有这些线程被配置为基本上并行地执行同一代码序列。在一个特定实施方式中，这通过以下方式来实现：每个线程具有相关联的程序计数器，并且数据处理装置也具有通用程序计数器。在任何时间点，在其相关联的程序计数器与通用程序计数器相匹配的所有线程上执行指令。通过这种方法，各种线程的

执行可以保持步调一致，并可以跨所有线程执行同一指令。这种处理在本文中被称为SIMT处理。

[0089] 然而，当遇到关键代码段时，这种步调一致的处理是不可能的，因为必须获取锁以使得能够执行关键代码段，并且锁一次只能由一个线程获得。因此，当遇到获得接力棒指令时，只有一个线程将成功地被分配以接力棒。在图9所示的示例中，假设它是第一线程405，如箭头1和2所指示。具体地，第一线程对接力棒的请求是成功请求，使得在接力棒表435中分配了条目，并且将接力棒分配给第一线程405。在此处，如箭头3所指示，执行流水线将使得从存储器440获取锁445，其中，通过使用先前描述的技术实现锁的获取。

[0090] 一旦获取到锁，可以执行关于第一线程405的关键代码段，并且在执行完之后执行传递接力棒指令。这将包括在接力棒维持电路内执行查找，并且将接力棒分配给其他待决线程中的一个。在此示例中，如箭头4A所指示，假设在执行完那些线程内相关联的获得接力棒指令之后，在等待线程列表120中对每个其他线程进行标识。在一个特定实施例中，虽然时序可以变化，假设最初四个线程都将同时执行获得接力棒指令，但只有第一线程被成功地分配了接力棒，并因此将在等待线程列表120中对所有其他线程的相关条目进行标识。

[0091] 因此，在此阶段可以将接力棒分配给那些待决线程中的任何线程，但是为了简单起见，假设以连续顺序为线程分配接力棒，并且因此在此处向下一个线程410分配接力棒，如箭头5所指示。在此处，可以执行关于第二线程410的关键代码段而无需重新获取锁，因为锁仍然被获取并且尚未被释放。当接下来与线程2相关地执行传递接力棒指令时，这使得接力棒被传递到第三线程415，如箭头6所指示。同样，可以与第三线程415相关地执行关键代码段而无需重新获取锁。当与第三线程415相关地执行传递接力棒指令时，这使得接力棒被分配给第四线程420，如箭头7所指示，并且同样执行关键代码段而无需重新获取锁。

[0092] 然而，当第四线程420执行传递接力棒指令时，如箭头8所指示，在等待线程列表中不存在剩余的条目了，因此在此处接力棒表条目被无效，并且如箭头9所指示，消息返回到第四线程420(例如，如先前参考图5所讨论，通过向目的寄存器Ws写入一)，此后第四线程继续执行使得锁445被释放，如箭头10所指示。

[0093] 图10示出如何在数据处理装置内建立多个接力棒域。具体地，为每个接力棒域500、530提供单独的接力棒表520、550，在该示例中每个接力棒域包括多个线程510、540。因此，在此示例中，可以将单独的线程组分配给不同的接力棒域，每个接力棒域具有其自身的相关联接力棒表，并且因此每个接力棒域具有自身的用于特定锁的相关联接力棒。在此示例中，假设共享存储器560中的锁570由线程组510、540两者使用。然而，在此示例中，如箭头1所指示，假设首先由第一接力棒域500的线程组510内的一个线程获取锁，并且相应地在接力棒表520内建立接力棒表条目。根据先前参考图9描述的过程，可以针对接力棒域1的线程保持获取锁，允许线程组510内的每个线程连续执行与锁相关联的关键代码段，如箭头2所指示。只有当所有这些线程都执行了关键代码段时，才将锁释放，如箭头3所指示。

[0094] 同时，线程组540内的线程也可以寻求请求接力棒，并实际上可以在接力棒表550内创建接力棒表条目，并且将接力棒分配给那些线程中的一个。然而，线程组540内已被分配了接力棒的第一线程将需要获取锁，并且在其可以获取到锁之前将必须等待线程组510释放该锁，如箭头4所指示。一旦获取到锁，则线程组540内的所有线程可以通过在接力棒表550内这些线程之间传递本地接力棒来连续执行其关键代码段，而同时锁570保持被分配，

如箭头5所指示。在线程组540内的所有线程执行完关键代码段之后，可以释放锁，如箭头6所指示。

[0095] 图11示出了可以使用的虚拟机实现。尽管先前描述的实施例在用于操作支持相关技术的特定处理硬件的装置和方法方面实现了本发明，但是也可以提供硬件设备的所谓的虚拟机实现。这些虚拟机实现在主机处理器630上运行，主机处理器630通常运行支持虚拟机程序610的主机操作系统620。通常要求大功率处理器来提供以合理速度执行的虚拟机实现，但可以在某些情况下调整这种方法，例如当出于兼容性或重用原因而期望在另一处理器的本地运行代码时。虚拟机程序610能够执行应用程序(或操作系统)600，以给出与由这样的真实硬件设备执行程序给出的结果相同的结果。因此，可以使用虚拟机程序610从应用程序600内执行程序指令(包括以上描述的获得接力棒指令和传递接力棒指令)。

[0096] 从上述实施例，应理解的是，所描述实施例的接力棒机制(其中接力棒形成锁的本地代理)允许多个线程连续地执行锁保护处理操作，而不需要各个线程获取和释放锁。相反，一旦获取锁，多线程中的每个线程可以连续地执行锁保护处理操作，并同时保持获取锁。只有当所有的这些多线程都执行了锁保护处理操作时，才释放锁。

[0097] 这种方法可以显著提高多线程系统的性能，其中线程包括锁保护处理操作。与图9所示类型的线程组相关联尤其受益，其中每个线程基本上步调一致地执行同一代码，因此会在基本相同的时间遇到需同一锁的关键代码段。然后，所述接力棒机制使得锁能够被获取一次，此后在锁被释放之前，线程组中的每个线程可以连续地执行关键代码段。这提高了线程组的执行性能，并且还减少了否则与对存储器的多次访问相关联以寻求多次获取和释放锁的能量消耗。

[0098] 尽管本文已经描述了特定的实施例，但应理解的是，本发明不限于此并且可以在本发明的范围内对其进行许多修改和添加。例如，可以将以下独立权利要求的特征与从属权利要求的特征进行各种组合，而不背离本发明的范围。

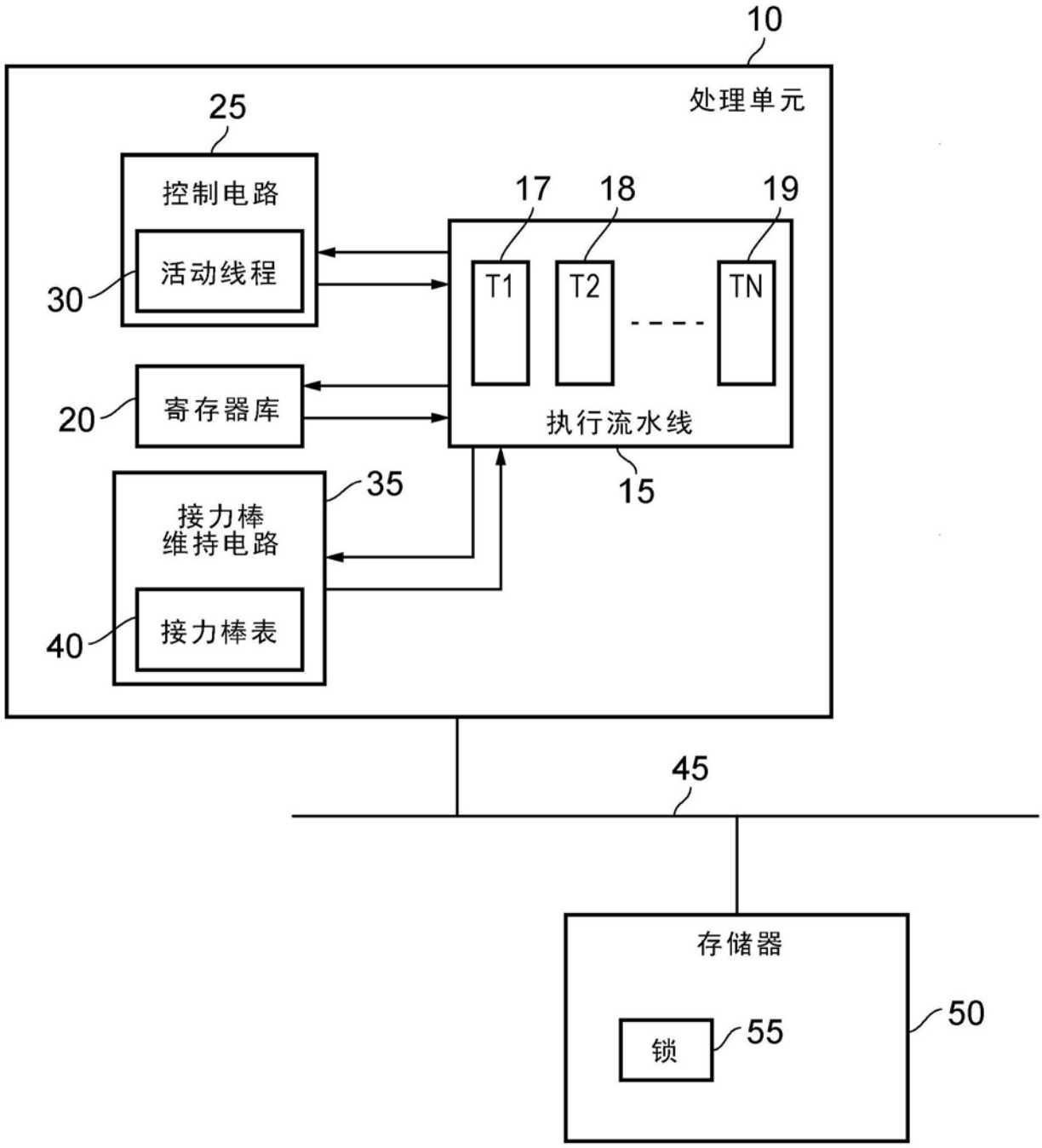

10
处理单元
25
控制电路
活动线程
30
寄存器库
20
17
18
19
T1
T2
TN
执行流水线
15
35
接力棒
维持电路
接力棒表
40
45
存储器
锁
55
50

图1

接力棒表

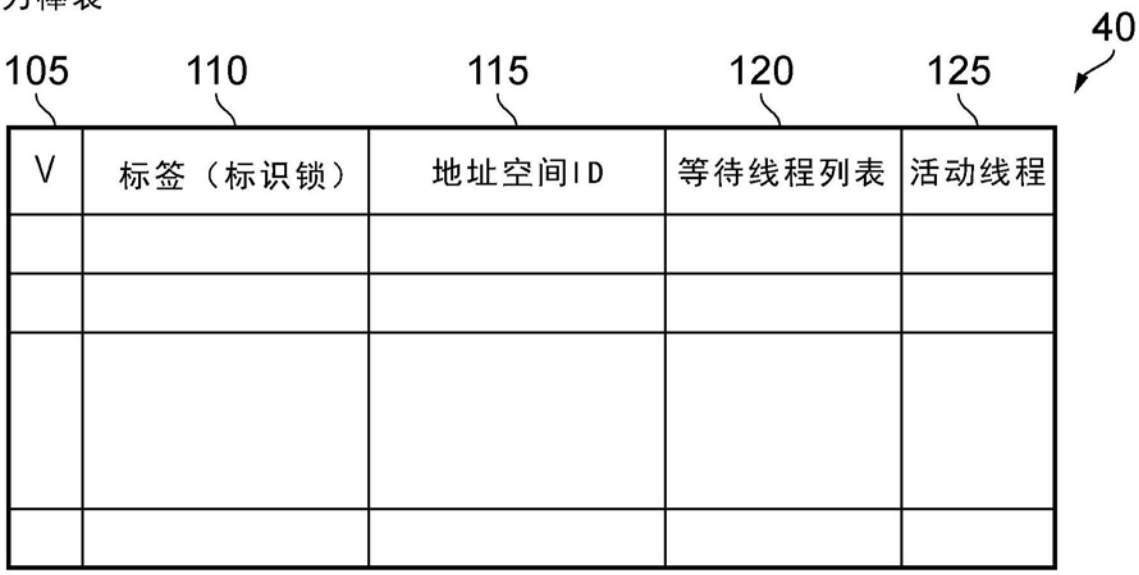

40
105
110
115
120
125
V
标签（标识锁）
地址空间ID
等待线程列表
活动线程

图2A

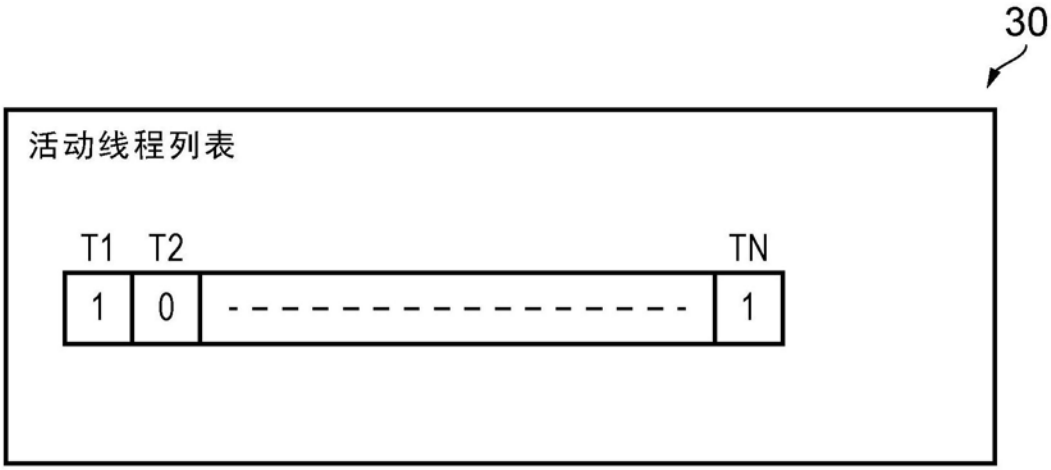
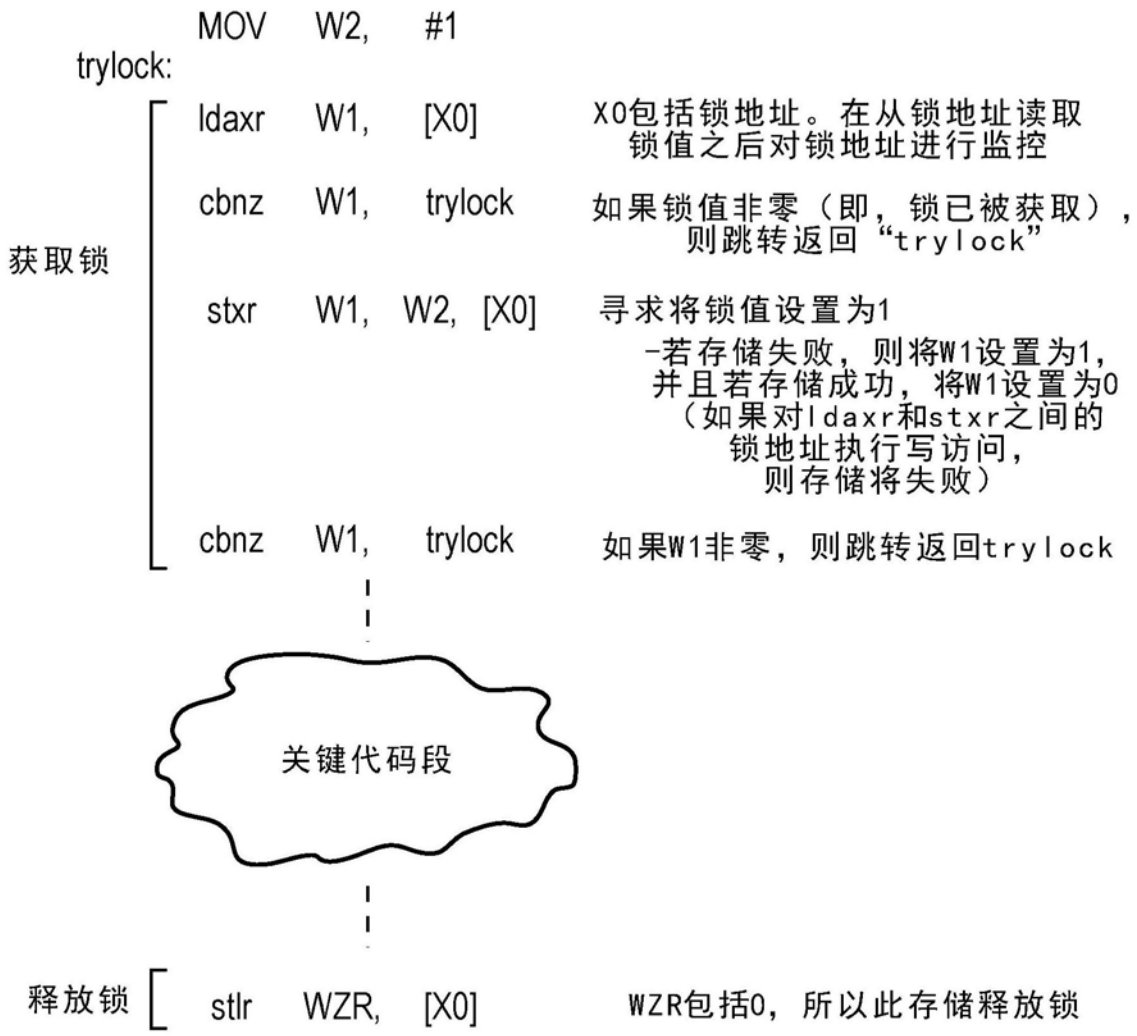

30
活动线程列表
T1
T2
TN
1
0
1

图2B


MOV W2, #1
trylock:
ldaxr W1, [X0]
X0包括锁地址。在从锁地址读取锁值之后对锁地址进行监控
cbnz W1, trylock
如果锁值非零（即，锁已被获取），则跳转返回“trylock”
获取锁
stxr W1, W2, [X0]
寻求将锁值设置为1
-若存储失败，则将W1设置为1，并且若存储成功，将W1设置为0
（如果对ldaxr和stxr之间的锁地址执行写访问，则存储将失败）
cbnz W1, trylock
如果W1非零，则跳转返回trylock
关键代码段
释放锁
stlr WZR, [X0]
WZR包括0，所以此存储释放锁

（现有技术）

图3

请求线程执行获得接力棒<WS>、<标签>时的接力棒维持处理

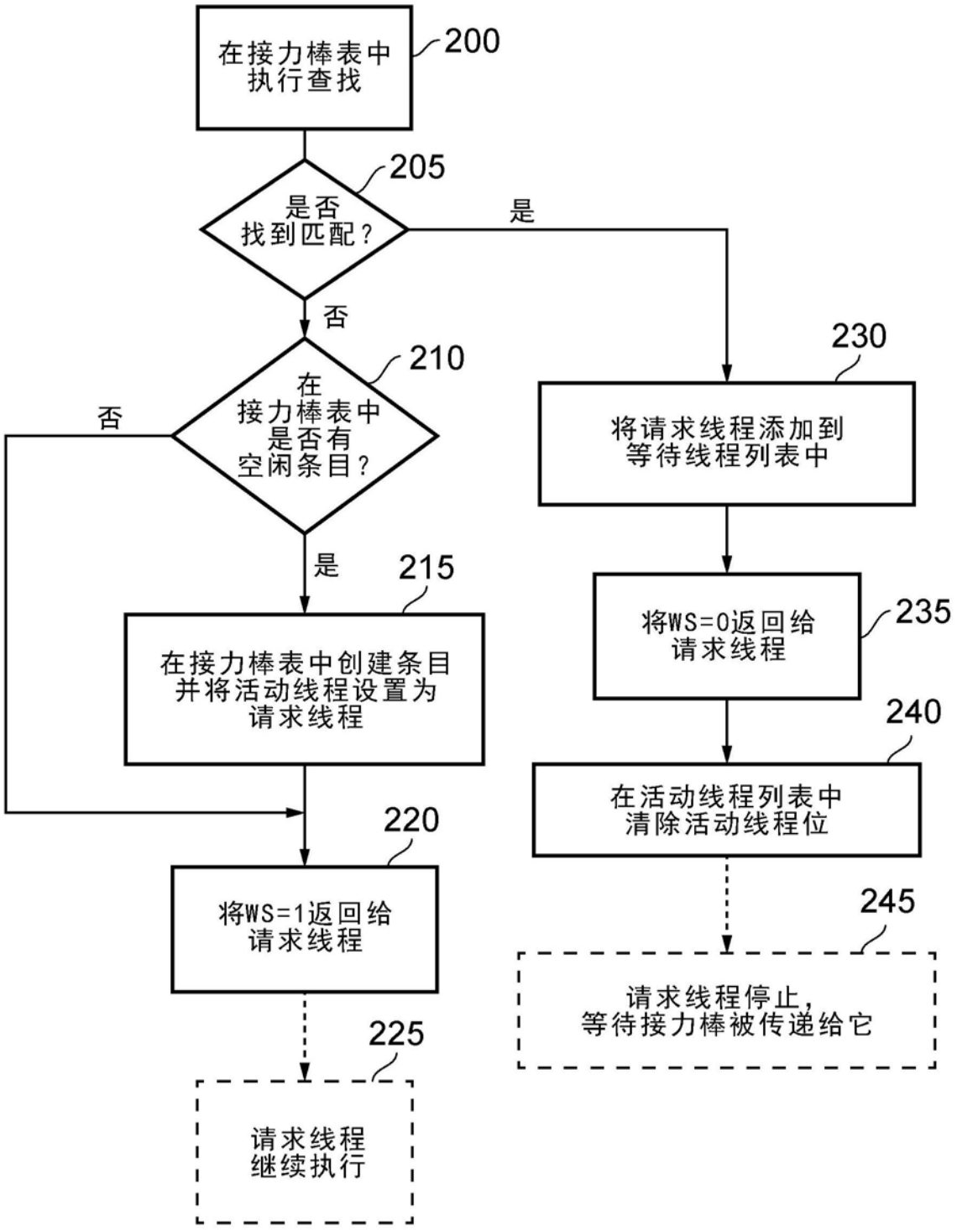


图4

传递线程执行传递接力棒<WS>、<标签>时的接力棒维持电路处理

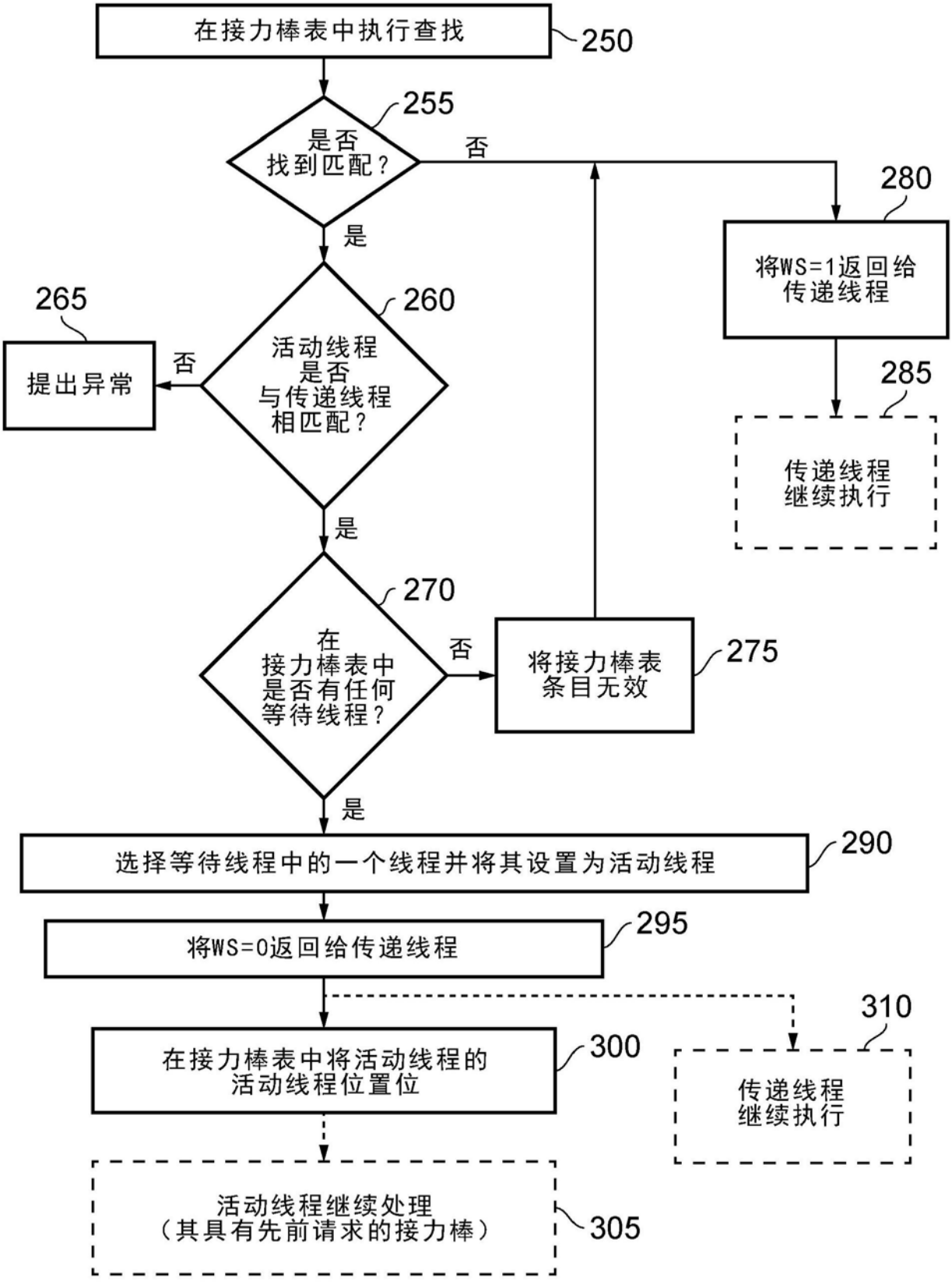


图5

线程获取接力棒

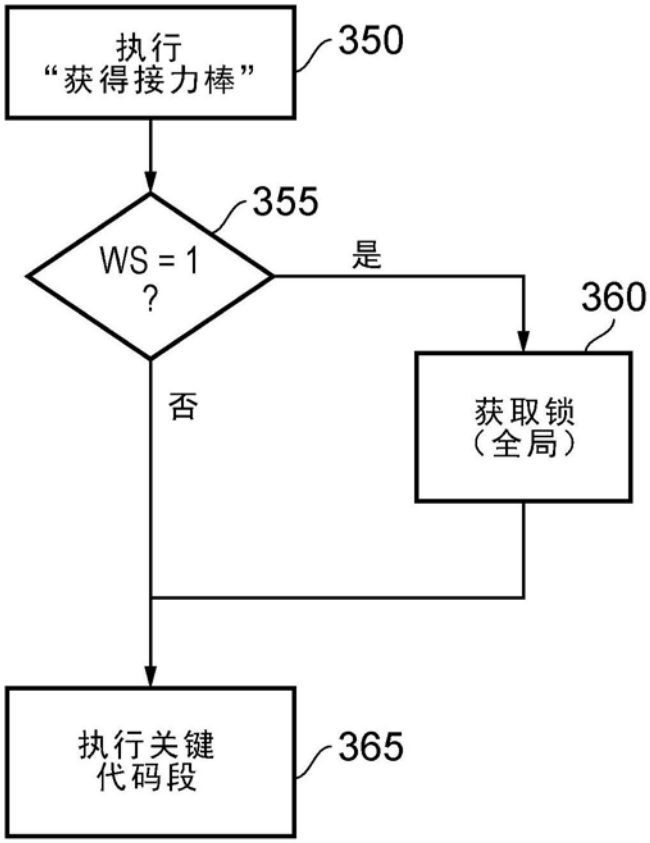


图6

线程释放接力棒

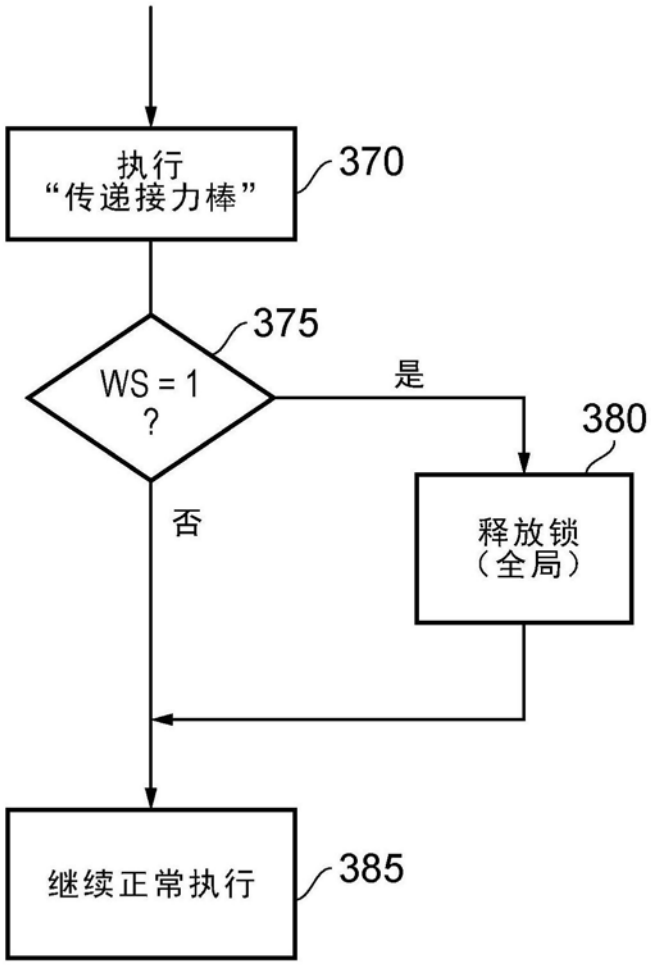


图7

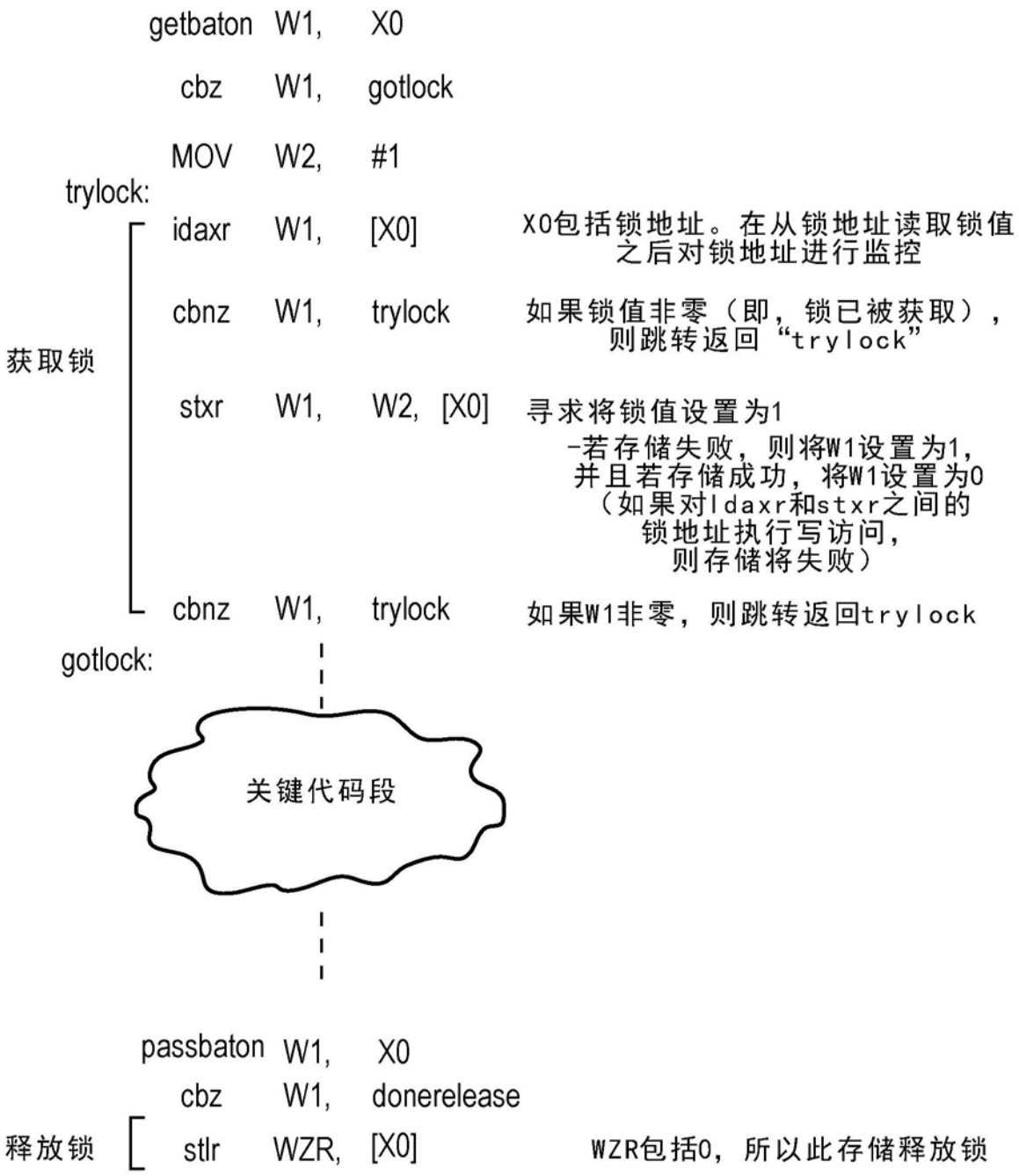

getbaton W1, X0
cbz W1, gotlock
MOV W2, #1
trylock:
获取锁
idaxr W1, [X0]
X0包括锁地址。在从锁地址读取锁值之后对锁地址进行监控
cbnz W1, trylock
如果锁值非零（即，锁已被获取），则跳转返回“trylock”
stxr W1, W2, [X0]
寻求将锁值设置为1
-若存储失败，则将W1设置为1，并且若存储成功，将W1设置为0
（如果对ldaxr和stxr之间的锁地址执行写访问，则存储将失败）
cbnz W1, trylock
如果W1非零，则跳转返回trylock
gotlock:
关键代码段
passbaton W1, X0
cbz W1, donerelease
释放锁
stlr WZR, [X0]
WZR包括0，所以此存储释放锁
donerelease:

图8A

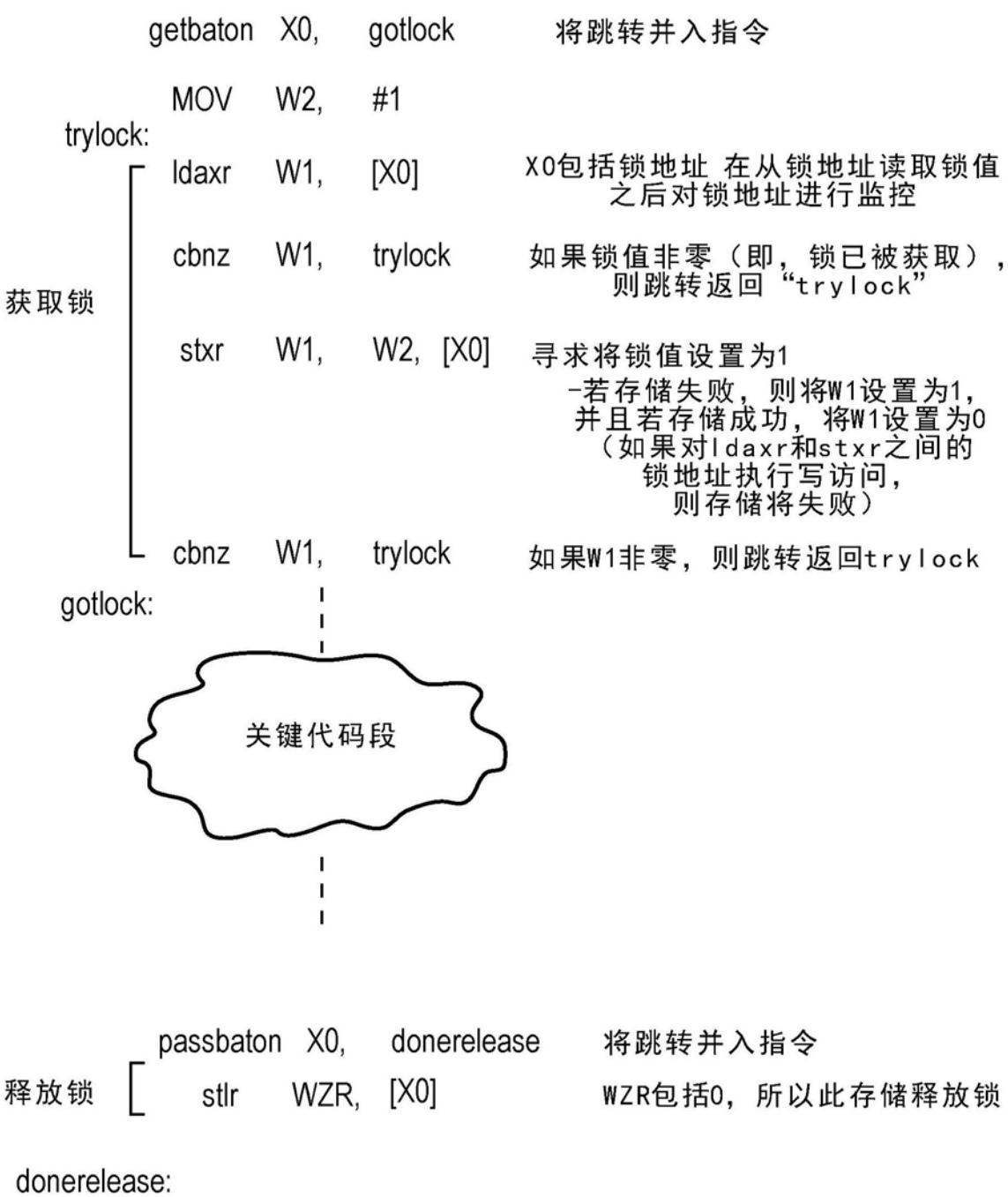

getbaton X0, gotlock
将跳转并入指令
MOV W2, #1
trylock:
ldaxr W1, [X0]
X0包括锁地址 在从锁地址读取锁值之后对锁地址进行监控
cbnz W1, trylock
如果锁值非零（即，锁已被获取），则跳转返回“trylock”
获取锁
stxr W1, W2, [X0]
寻求将锁值设置为1
-若存储失败，则将W1设置为1，并且若存储成功，将W1设置为0
（如果对ldaxr和stxr之间的锁地址执行写访问，则存储将失败）
cbnz W1, trylock
如果W1非零，则跳转返回trylock
gotlock:
关键代码段
passbaton X0, donerelease
将跳转并入指令
释放锁
stlr WZR, [X0]
WZR包括0，所以此存储释放锁
donerelease:

图8B

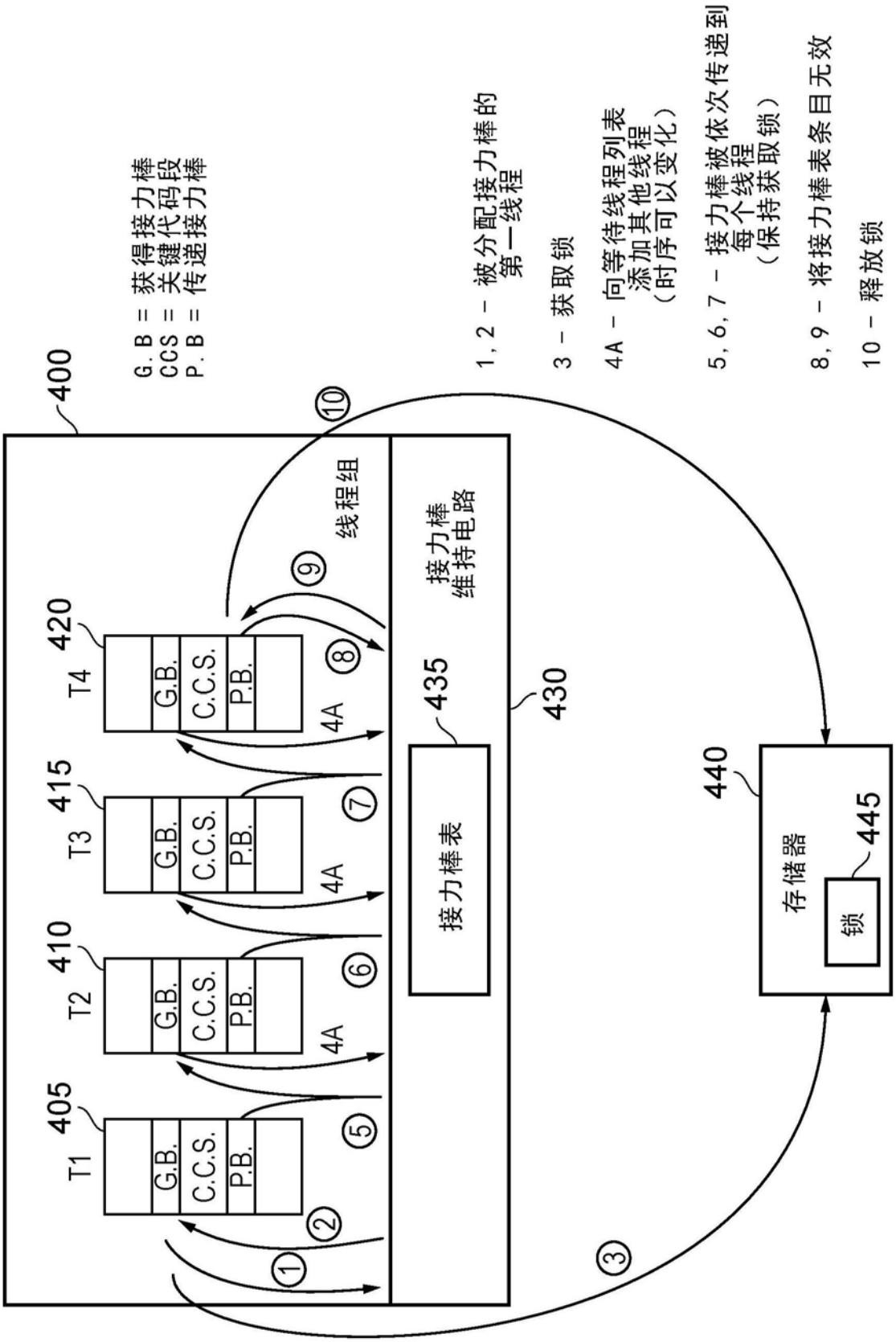

G.B = 获得接力棒
CCS = 关键代码段
P.B = 传递接力棒
1, 2 – 被分配接力棒的第一线程
3 – 获取锁
4A – 向等待线程列表添加其他线程（时序可以变化）
5, 6, 7 – 接力棒被依次传递到每个线程（保持获取锁）
8, 9 – 将接力棒表条目无效
10 – 释放锁
400
405
410
415
420
T1
T2
T3
T4
G.B.
C.C.S.
P.B.
线程组
接力棒维持电路
430
435
接力棒表
440
存储器
445
锁
4A

图9

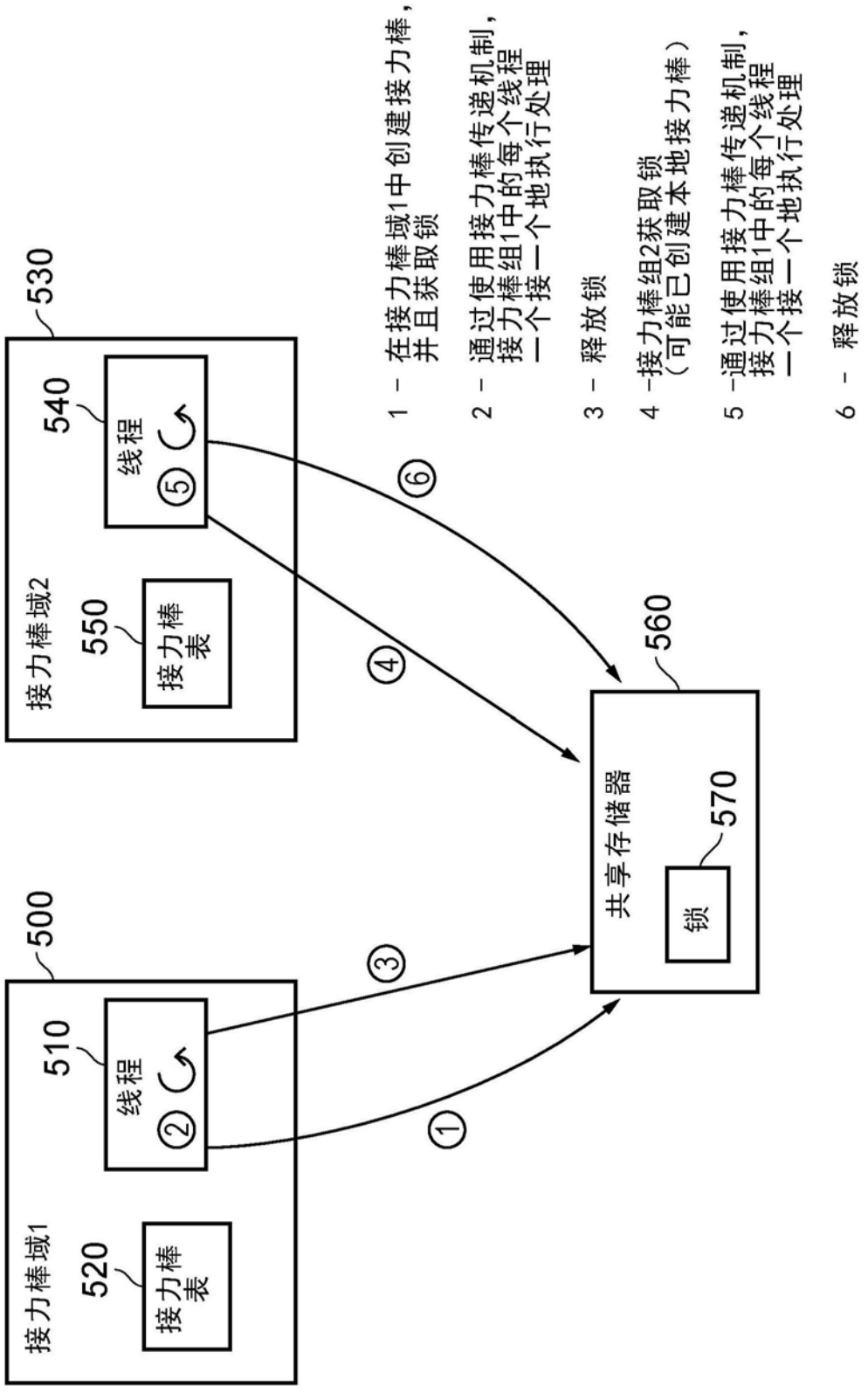

1 - 在接力棒域1中创建接力棒，并且获取锁
2 - 通过使用接力棒传递机制，接力棒组1中的每个线程一个接一个地执行处理
3 - 释放锁
4 -接力棒组2获取锁（可能已创建本地接力棒）
5 -通过使用接力棒传递机制，接力棒组1中的每个线程一个接一个地执行处理
6 - 释放锁
500
接力棒域1
520
接力棒表
510
线程
530
接力棒域2
550
接力棒表
540
线程
560
共享存储器
570
锁

图10

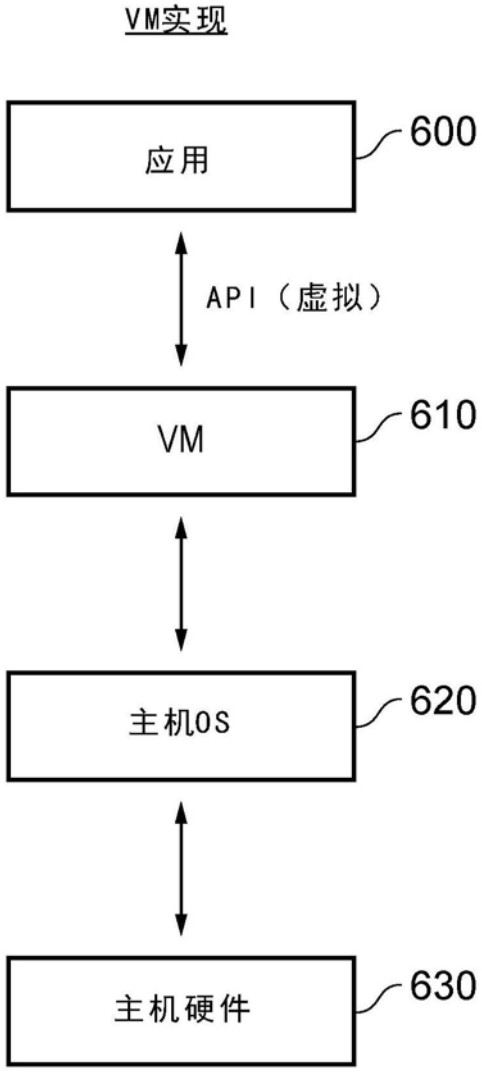


图11