(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 103562866 B
(45)授权公告日 2018.03.30

(21)申请号 201280024054.0

(22)申请日 2012.03.23

(65)同一申请的已公布的文献号
申请公布号 CN 103562866 A

(43)申请公布日 2014.02.05

(30)优先权数据
61/467,939 2011.03.25 US

(85)PCT国际申请进入国家阶段日
2013.11.18

(86)PCT国际申请的申请数据
PCT/US2012/030383 2012.03.23

(87)PCT国际申请的公布数据
WO2012/135041 EN 2012.10.04

(73)专利权人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 M·阿布达拉

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

(51)Int.Cl.
G06F 9/46(2006.01)
G06F 9/30(2006.01)
G06F 12/0811(2016.01)

(56)对比文件
US 2004/0216120 A1,2001.10.28,
CN 101449256 A,2009.06.03,
CN 101449256 A,2009.06.03,
US 2009/0150647 A1,2009.06.11,
US 7711929 B2,2010.05.04,
CN 101627365 A,2010.01.13,

审查员 阳升

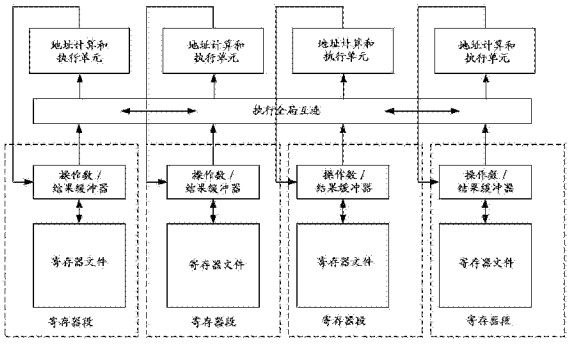
权利要求书2页 说明书14页 附图23页

(54)发明名称

用于通过使用由可分割引擎实例化的虚拟核来支持代码块执行的寄存器文件段

(57)摘要

一种用于使用用于处理器的多个寄存器文件片段来执行指令的系统。该系统包括:全局前端调度器,用于接收传入指令序列,其中全局前端调度器将传入指令序列分割成指令的多个代码块并且生成描述在代码块的指令之间的相互依赖性的多个继承性矢量。该系统还包括:处理器的多个虚拟核,被耦合用于接收全局前端调度器分配的代码块,其中每个虚拟核包括多个可分割引擎的资源的相应子集,其中根据虚拟核模式并且根据相应的继承性矢量通过使用可分割引擎来执行代码块。多个寄存器文件段耦合到可分割引擎用于提供数据存储。



CN 103562866 B

1.一种用于使用用于处理器的多个寄存器文件段来执行指令的系统,所述系统包括:

全局前端调度器,用于接收传入指令序列,其中所述全局前端调度器将所述传入指令序列分割成指令的多个代码块并且生成描述在所述代码块的指令之间的相互依赖性的多个继承性矢量;

所述处理器的多个虚拟核,被耦合用于接收由所述全局前端调度器分配的代码块,其中每个虚拟核包括多个可分割引擎的资源的相应子集,其中每个可分割引擎的资源可操作为被分隔以使用其他可分割引擎的被分割的资源来实例化虚拟核,其中所述多个可分割引擎中的每个引擎的资源之间的通信由全局互连结构支持,并且其中根据虚拟核模式并且根据相应的所述继承性矢量通过使用所述可分割引擎来执行所述代码块;以及

多个寄存器文件段,耦合到所述可分割引擎用于提供数据存储,其中所述全局互连结构将所述多个寄存器文件段中的每个寄存器文件段链接至所述多个可分割引擎中的每个可分割引擎。

2.根据权利要求1所述的系统,其中所述多个寄存器文件段实施执行模式,其中分配每个寄存器文件段的物理资源的子集以支持逻辑核的单个逻辑线程的执行。

3.根据权利要求2所述的系统,其中每个寄存器文件段实施多个逻辑核的部分。

4.根据权利要求1所述的系统,其中所述多个寄存器文件段实施执行模式,其中根据可调节阈值来动态分配每个寄存器文件段的物理资源以支持单个逻辑核的单个逻辑线程的执行。

5.根据权利要求4所述的系统,其中所述多个寄存器文件段实施多个逻辑核的部分。

6.根据权利要求1所述的系统,其中所述多个寄存器文件段实施执行模式,其中分配每个寄存器文件段的物理资源的集合以支持单个逻辑线程的执行。

7.根据权利要求1所述的系统,其中每个寄存器文件段还包括公共分割调度器、操作数和结果缓冲器以及线程式寄存器文件。

8.根据权利要求1所述的系统,其中所述全局互连结构包括路由矩阵,所述路由矩阵可操作为允许所述多个可分割引擎访问在所述多个寄存器文件段中的任何点处存储的数据。

9.一种用于使用多个寄存器文件段来执行指令的处理器,所述处理器包括:

全局前端调度器,用于接收传入指令序列,其中所述全局前端调度器将所述传入指令序列分配到指令的多个代码段中并且生成描述在所述代码段的指令之间的相互依赖性的多个继承性矢量;

所述处理器的多个虚拟核,被耦合用于接收由所述全局前端调度器分配的代码块,其中每个虚拟核包括多个可分割引擎的资源的相应子集,其中每个可分割引擎的资源可操作为被分隔以使用其它可分隔引擎的被分割的资源来实例化虚拟核,其中所述多个可分割引擎中的每个引擎的资源之间的通信由全局互连结构支持,并且其中根据虚拟核模式并且根据相应的所述继承性矢量通过使用所述可分割引擎来执行所述代码块;

多个执行单元,耦合到所述全局前端调度器用于根据相应的继承性矢量执行所述代码段;以及

多个寄存器文件段,耦合到所述执行单元用于提供数据存储,其中所述全局互连结构将所述多个寄存器文件段中的每个寄存器文件段链接至所述多个可分割引擎中的每个可分割引擎。

10.根据权利要求9所述的处理器，其中所述多个寄存器文件段实施执行模式，其中分配每个寄存器文件段的物理资源的子集以支持逻辑核的单个逻辑线程的执行。

11.根据权利要求10所述的处理器，其中每个寄存器文件段实施多个逻辑核的部分。

12.根据权利要求9所述的处理器，其中所述多个寄存器文件段实施执行模式，其中根据可调节阈值来动态分配每个寄存器文件段的物理资源以支持单个逻辑核的单个逻辑线程的执行。

13.根据权利要求12所述的处理器，其中所述多个寄存器文件段实施多个逻辑核的部分。

14.根据权利要求9所述的处理器，其中所述多个寄存器文件段实施执行模式，其中分配每个寄存器文件段的物理资源的所述集合以支持单个逻辑线程的执行。

15.根据权利要求9所述的处理器，其中每个寄存器文件段还包括公共分割调度器、操作数和结果缓冲器以及线程式寄存器文件。

16.根据权利要求9所述的处理器，其中所述全局互连结构包括路由矩阵，所述路由矩阵可操作为允许所述多个可分割引擎访问在所述多个寄存器文件段中的任何点处存储的数据。

用于通过使用由可分割引擎实例化的虚拟核来支持代码块执行的寄存器文件段

[0001] 本申请要求对Mohammad A.Abdallah于2011年3月25日提交的、名称为"REGISTER FILE SEGMENTS FOR SUPPORTING CODE BLOCK EXECUTION BY USING VIRTUAL CORES INSTANTIATED BY PARTITIONABLE ENGINES"的、共同未决、共同转让的美国临时专利申请第61/467,939号的权益，并且其全部内容并入于此。

[0002] 有关申请的交叉引用

[0003] 本申请与Mohammad A.Abdallah于2007年4月12日提交的、名称为"APPARATUS AND METHOD FOR PROCESSING AN INSTRUCTION MATRIX SPECIFYING PARALLEL IN DEPENDENT OPERATIONS"的、共同未决、共同转让的美国专利申请公开第2009/0113170号有关，并且其全部内容并入于此。

[0004] 本申请与Mohammad A.Abdallah于2007年11月14日提交的、名称为"APPARATUS AND METHOD FOR PROCESSING COMPLEX INSTRUCTION FORMATS IN A MULTITHREADED ARCHITECTURE SUPPORTING VARIOUS CONTEXT SWITCH MODES AND VIRTUALIZATION SCHEMES"的、共同未决、共同转让的美国专利申请公开第2010/0161948号有关，并且其全部内容并入于此。

技术领域

[0005] 本发明总体上涉及数字计算机系统、更具体地涉及用于选择包括指令序列的指令的系统和方法。

背景技术

[0006] 处理器被要求处置依赖或者全独立的多个任务。这样的处理器的内部状态通常由可以在每个特定程序执行的实例保持不同值的寄存器构成。在每个程序执行的实例，内部状态映像被称为处理器的架构状态。

[0007] 如果将代码执行进行切换以运行另一功能(例如另一线程、进程或者程序)，则必须保存机器/处理器的状态，从而新功能可以利用内部寄存器以构建它的新状态。一旦终止新功能，则可以丢弃它的状态，并且将还原先前上下文的状态而且执行恢复。这样的切换过程被称为上下文切换并且尤其对于运用大量寄存器(例如64、128、256个)和/或无序执行的现代架构通常包括数十或者数百个周期。

[0008] 在线程感知硬件架构中，硬件通常针对有限数目的由硬件支持的线程而支持多个上下文状态。在这一情况下，硬件针对每个支持的线程重复所有架构状态单元。这消除了在执行新线程时对于上下文切换的需要。然而这仍然具有多个缺点、即针对硬件中支持的每个附加线程重复所有架构状态单元(即寄存器)的面积、功率和复杂性。此外，如果软件线程的数目超过显式支持的硬件线程的数目，则仍然必须执行上下文切换。

[0009] 这随着在要求大量线程的细粒度基础上需要并行性而变得常见。具有重复上下文状态硬件存储的硬件线程感知架构无助于非线程式软件代码而仅针对线程化的软件减少

上下文切换数目。然而那些线程通常是针对粗粒度并行性而构造的并且导致用于发起和同步的繁重软件开销，从而让细粒度并行性、比如功能调用和循环并行执行而没有高效线程化发起/自动生成。如此描述的开销伴随有难以使用用于非显式/容易并行化/线程化的软件代码的现有技术编译器或者用户并行化技术来自动地并行化这样的代码。

发明内容

[0010] 在一个实施例中，将本发明实施为一种用于使用用于处理器的多个虚拟核来执行指令的系统。该系统包括：全局前端调度器，用于接收传入指令序列，其中所述全局前端调度器将所述传入指令序列分割成指令的多个代码块并且生成描述在所述代码块的指令之间的相互依赖性的多个继承性矢量。该系统还包括：所述处理器的多个虚拟核，被耦合用于接收由所述全局前端调度器分配的代码块，其中每个虚拟核包括多个可分割引擎的资源的相应子集，其中根据虚拟核模式并且根据相应的所述继承性矢量通过使用所述可分割引擎来执行所述代码块。多个寄存器文件段，耦合到所述可分割引擎用于提供数据存储。

[0011] 本发明的其它实施例利用公共调度器、公共寄存器文件和公共存储器子系统以实施用于处理器的多个可分割引擎的片段式地址空间。可分割引擎可以用来实施多个虚拟核。片段通过允许附加虚拟核以协同地执行指令序列来实现微处理器性能升级。片段分级可以跨越每个高速缓存分级(例如L1高速缓存、L2高速缓存和公共寄存器文件)相同。片段分级可以使用地址位将地址空间划分成片段，其中使用地址位使得片段在高速缓存线边界以上并且在页面边界以下。每个片段可以被配置为利用多端口存储体结构用于存储。

[0012] 前文是发明内容、因此必然包含细节的简化、概括和省略；因而本领域技术人员将理解，发明内容仅为示例而并非旨在于以任何方式进行限制。如仅由权利要求限定的本发明的其它方面、发明特征和优点将在下文阐述的非限制具体描述中变得清楚。

附图说明

[0013] 在附图的各图中通过示例而非通过限制举例说明本发明，并且在附图中，相似标号指代相似单元。

[0014] 图1A示出全局前端生成代码块和继承性矢量以支持在代码序列的相应可分割引擎上执行代码序列的方式的概况。

[0015] 图1B示出根据本发明的一个实施例的可分割引擎及其部件的概况图，这些部件包括用于多核处理器的分段式调度器和寄存器文件、全局互连以及分段式存储器子系统。

[0016] 图2示出根据本发明的一个实施例的调度器流程图。

[0017] 图3示出根据本发明的一个实施例的示例硬件电路的图，该图示出存储具有互连的操作数和结果的分段式寄存器文件。

[0018] 图4示出根据本发明的一个实施例的描绘全局前端提取和调度器的图。

[0019] 图5示出根据本发明的一个实施例的跨越许多虚拟核的指令分布的备选实现方式。

[0020] 图6示出根据本发明的一个实施例的具有对应多个寄存器文件以及操作数和结果缓冲器的多个寄存器段。

[0021] 图7示出根据本发明的一个实施例的用于多核处理器的片段式存储器子系统的更

具体图。

[0022] 图8示出描绘根据本发明的一个实施例的通过地址生成如何使用地址的位以枚举片段的图。

[0023] 图9示出本发明的实施例的如何处置加载和存储的图。

[0024] 图10示出根据本发明的一个实施例的可以将片段拆分成两个或者更多域的方式。

[0025] 图11示出根据本发明的一个实施例的处理器的操作模式，在该操作模式中虚拟核被配置作为在执行应用中与逻辑核对应的物理核。

[0026] 图12示出根据本发明的一个实施例的处理器的操作模式，在该操作模式中虚拟核被配置作为在执行应用中与逻辑核对应的软核。

[0027] 图13示出根据本发明的一个实施例的处理器的操作模式，在该操作模式中虚拟核被配置作为在执行应用中与单个逻辑核对应的软核。

[0028] 图14示出根据本发明的一个实施例的用来支持逻辑核和虚拟核功能的片段分割的示例性实现方式。

[0029] 图15示出根据本发明的一个实施例的实施多物理到多逻辑模式的示例性四片段处理器的片段存储器。

[0030] 图16示出根据本发明的一个备选实施例的实施多物理到多逻辑模式的示例性四片段处理器的片段存储器。

[0031] 图17示出根据本发明的一个实施例的实施多软核到多逻辑核模式的示例性四片段处理器的片段存储器。

[0032] 图18示出根据本发明的一个实施例的实施多软核到一个逻辑核模式的示例性四片段处理器的片段存储器。

[0033] 图19示出根据本发明的一个实施例的实施物理到逻辑模式的示例性四片段处理器的地址计算和执行单元、操作数/结果缓冲器、线程式寄存器文件以及公共分割调度器。

[0034] 图20示出根据本发明的一个实施例的用来实施多物理到多逻辑模式的示例性四片段处理器的地址计算和执行单元、操作数/结果缓冲器、线程式寄存器文件以及公共分割调度器的备选实现方式。

[0035] 图21示出根据本发明的一个实施例的实施多软核到多逻辑模式的示例四片段处理器的地址计算和执行单元、寄存器文件以及公共分割调度器。

[0036] 图22示出根据本发明的一个实施例的实施多软核到一个逻辑核模式的示例性四片段处理器的地址计算和执行单元、寄存器文件以及公共分割调度器。

[0037] 图23示出根据本发明的一个实施例的示例性微处理器流水线的图。

具体实施方式

[0038] 虽然已经结合一个实施例来描述本发明，但是本发明未旨在局限于这里阐述的具体形式。与此相反，它旨在覆盖如能够在如所附权利要求限定的本发明的范围内合理包括的这样的备选、修改和等同。

[0039] 在以下具体描述中，已经阐述许多具体细节、比如具体方法顺序、结构、单元和连接。然而将理解，无需利用这些和其它具体细节以实现本发明的实施例。在其它境况中，已经省略或者尚未特别具体描述公知结构、单元或者连接以免不必要地使本描述混淆。

[0040] 在说明书内对“一个实施例”或者“实施例”的引用旨在于指示结合该实施例描述的特定特征、结构或者特性包含于本发明的至少一个实施例中。在说明书内的各处出现短语“在一个实施例中”未必都指代相同实施例，也并非是与其它实施例互斥的单独或者备选实施例。另外，描述可以被一些实施例而未被其它实施例表现的各种特征。类似地，描述各种要求，这些要求可以是针对一些实施例、但是并非其它实施例的要求。

[0041] 以流程、步骤、逻辑块、处理和对计算机存储器内的数据位的操作的其它符号表示形式呈现以下具体描述的一些部分。这些描述和表示形式是数据处理领域中的技术人员用来向本领域其他技术人员最有效传达他们的工作实质的手段。过程、计算机执行的步骤、逻辑块、处理等在这里以及总体上被理解为是导致所期望结果的步骤或者指令的自一致序列。步骤是需要对物理量进行物理操纵的步骤。这些量尽管未必、但是通常采用计算机可读存储介质的电或者磁信号的形式并且能够在计算机系统中被存储、传送、组合、比较和以其他方式进行操纵。主要出于普遍使用的原因而将这些信号称为位、数值、单元、符号、字符、项、数等已经证实有时是方便的。

[0042] 然而应当谨记，所有这些和相似术语将与适当物理量关联并且仅为应用于这些量的方便标注。除非如从以下讨论中清楚的那样另有特别指出，否则要理解，贯穿本发明利用诸如“处理”或者“访问”或者“写入”或者“存储”或者“重复”等术语的讨论指代计算机系统或者相似电子计算设备的动作和过程，该计算机系统或者电子计算设备将计算机系统的寄存器和存储器以及其它计算机可读介质内表示为物理(电子)量的数据操纵和变换成计算机系统存储器或者寄存器或者其它这样的信息存储、传输或者显示设备内相似地表示为物理量的其它数据。

[0043] 本发明的实施例利用公共全局前端调度器、多个片段式寄存器文件和存储器子系统以实施用于多核处理器的多个核的片段式地址空间。在一个实施例中，片段通过允许附加虚拟核(例如软核)以协同地执行包括一个或者多个线程的指令序列来实现微处理器性能升级。片段分级跨越每个高速缓存分级(例如L1高速缓存、L2高速缓存和公共寄存器文件)相同。片段分级使用地址位将地址空间划分成片段，其中使用地址位使得由在高速缓存线边界以上并且在页面边界以下的位来标识片段。每个片段被配置为利用多端口存储体结构用于存储。在以下图1A和1B中进一步描述本发明的实施例。

[0044] 图1A示出根据本发明的一个实施例的处理器的概况图。如图1A中描绘的，处理器包括全局前端提取和调度器150以及多个可分割引擎11-14。

[0045] 图1A示出全局前端生成代码块和继承性矢量以支持在代码序列的相应可分割引擎上执行代码序列的方式的概况。代码序列20-23中的每个代码序列可以根据特定虚拟核执行模式属于相同逻辑核/线程或者属于不同逻辑核/线程。全局前端提取和调度器将处理代码序列20-23以生成代码块和继承性矢量。如图所示这些代码块和继承性矢量被分配至特定可分割引擎11-14。

[0046] 可分割引擎根据选择的模式实施虚拟核。可分割引擎包括段、片段和多个执行单元。在可分割引擎内的资源可以用来实施具有多个模式的虚拟核。如虚拟核模式调配的那样，可以实施一个软核或者许多软核以支持一个逻辑核/线程。在图1A的实施例中，根据选择的模式，虚拟核可以支持一个逻辑核/线程或者四个逻辑核/线程。在虚拟核支持四个逻辑核/线程的实施例中，跨越可分割引擎中的每个可分割引擎展开每个虚拟核的资源。在虚

拟核支持一个逻辑核/线程的实施例中，所有引擎的资源专用于该核/线程。引擎被分割使得每个引擎提供包括每个虚拟核的资源的子集。换而言之，虚拟核将包括引擎11-14中的每个引擎的资源的子集。在引擎11-14中的每个引擎的资源之间的通信由全局互连结构30提供以便有助于这一过程。备选地，引擎11-14可以用来实施物理模式，在该物理模式中，引擎11-14的资源专用于支持专用核/线程的执行。以这一方式，由引擎实施的软核包括让资源跨越引擎中的每个引擎展开的虚拟核。在以下图中进一步描述虚拟核执行模式。

[0047] 应当注意在常规核实现方式中，仅仅一个核/引擎内的资源被唯一分配给一个逻辑线程/核。对照而言，在本发明的实施例中，任何引擎/核的资源可以被分割以与其它引擎/核分割共同实例化向一个逻辑线程/核分配的虚拟核。此外，本发明的实施例可以实施多个虚拟执行模式，在执行虚拟执行模式中，那些相同引擎可以被分割以支持许多专用核/线程、许多动态分配的核/线程或者所有引擎的所有资源都支持单个核/线程的执行的实施例。在以下描述中进一步描述这些实施例。

[0048] 图1B示出根据本发明的一个实施例的可分割引擎及其部件的概况图，这些部件包括用于多核处理器的分段式调度器和寄存器文件、全局互连以及片段式存储器子系统。如图1B中描绘的，示出四个片段101-104。片段分级跨越每个高速缓存分级(例如L1高速缓存、L2高速缓存和加载存储缓冲器)相同。可以经由存储器全局互连110a在L1高速缓存中的每个高速缓存、L2高速缓存中的每个高速缓存和加载存储缓冲器中的每个加载存储缓冲器之间交换数据。

[0049] 存储器全局互连包括路由矩阵，该路由矩阵允许多个核(例如地址计算和执行单元121-124)访问可以在片段式高速缓存分级(例如L1高速缓存、加载存储缓冲器和L2高速缓存)中的任何点存储的数据。图1B也描绘地址计算和执行单元121-124可以经由存储器全局互连110a访问片段101-104中的每个片段的方式。

[0050] 执行全局互连110b类似地包括路由矩阵，该路由矩阵允许多个核(例如地址计算和执行单元121-124)访问可以在片段式寄存器文件中的任何片段式寄存器文件存储的数据。因此，核具有经由存储器全局互连110a或者执行全局互连110b对在片段中的任何片段中存储的数据和在段中的任何段中存储的数据的访问。此外，应当注意在一个实施例中，在公共分割提取和调度器中的每个公共分割提取和调度器之间存在另一全局互连。这由在每个公共分割提取和调度器之间并且连接每个公共分割提取和调度器的水平箭头所示出。

[0051] 图1B还示出全局前端提取和调度器150，该全局前端提取和调度器具有整个机器的了解并且管理寄存器文件段和片段式存储器子系统的利用。地址生成包括用于片段定义的基础。全局前端提取和调度器通过向每个段的分割调度器分配指令序列来工作。公共分割调度器然后分派那些指令序列用于在地址计算和执行单元121-124上执行。

[0052] 应当注意在一个实施例中可以向全局前端调度器150中并入公共分割提取和调度器的功能。在这样的实施例中，段将未包括相应公共分割提取和调度器，并且将无需在它们之间的互连。

[0053] 此外，应当注意可以用分级方式嵌套图1A中所示可分割引擎。在这样的实施例中，第一级可分割引擎将包括本地前端提取和调度器以及连接到它的多个次级可分割引擎。

[0054] 图2示出根据本发明的一个实施例的调度器流程图。如图2中描绘的，示出包括推测性线程存储桶指针、存储桶源列表和目的列表的存储桶缓冲器。调度器和执行存储桶包

括存储桶分派选择器以及虚拟寄存器匹配和读取、包括寄存器分级和寄存器高速缓存的可能性。后端是记录执行的存储桶之处，并且在引退之前实行异常排序。寄存器分级/高速缓存也用为用于执行的存储桶结果的中间存储装置直至它们为非推测性并且可以更新架构状态。以下公开记录执行的存储桶的前端、分派级和后端的一个可能实现方式。

[0055] 图2示出该概念从管理少量紧密耦合的线程的存储桶缓冲器升级成管理多个存储桶缓冲器和线程的硬件电路的方式。将描述那些可以被扩展以处理可能具有更少紧密交互的更大量线程的电路为全局前端(例如图1B中所示全局前端调度器150)。

[0056] 该过程通过提取新线程矩阵/存储桶/块来开始，然后向存储桶缓冲器中的空余存储桶槽中指派新线程存储桶。线程分配指针阵列852中的线程分配指针中的每个线程分配指针组成存储桶间隔，该存储桶间隔物理上允许线程在该存储桶间隔中放置它的指令块/存储桶。这些线程中的每个线程保持以轮循方式向在它的对应连续空间间隔内部的存储桶缓冲器阵列中分配存储桶。每当新存储桶/块被指派时在每个线程空间内部的存储桶/块获得指派的递增的新编号854。针对每个存储桶中的有效源，针对每个存储桶，有效源具有有效读取位“Rv”，该有效读取位“Rv”指示在这一个存储桶内部的指令需要这一个源。按照相同约定，在这一个存储桶中的将回写指令的每个指令寄存器具有在存储桶中设置的有效位“Wv”，并且它在目的继承性矢量853中具有字段。在将向存储桶缓冲器中提取新存储桶时，它从由线程存储桶分配指针852指向的先前分配的存储桶继承目的继承性矢量。该继承性矢量从先前分配的存储桶复制并且然后它改写与由那些存储桶指令将更新的寄存器对应的那些有效目的字段。将用当前存储桶编号来标注有效目的，而从在存储桶内部的对应继承性矢量复制无效目的。然后对于新提取的存储桶通过递增线程存储桶指针来更新它的指针(指针在它的间隔内卷绕)。

[0057] 在存储桶分派和执行阶段中，无论何时执行存储桶都没有任何异常处置，然后设置并且贯穿存储桶缓冲器而且在每个如下存储桶内锁存/监视广播存储桶异常标志(包含存储桶编号)854，该存储桶以具有该存储桶编号的源为源。也有可能随着存储桶编号传递其它有关信息、比如关于虚拟寄存器位置的信息。如果存储桶内设置源存储桶的所有执行标志，则设置该存储桶就绪位855并且该存储桶准备好被分派和执行。如果存储桶执行而无任何异常并且它准备好按照程序的依次顺序更新架构状态，则它引退存储桶并且将引退线程指针857递增到阵列中的下一个存储桶。可以向新存储桶指派引退的存储桶位置。

[0058] 那些紧密相关线程可以都共存于矩阵/存储桶/块缓冲器内部；每个线程将占用属于该线程的连续存储桶间隔。该线程的分配指针以轮循方式在这一存储桶间隔内部移动从而提取新指令存储桶并且以描述的轮循方式在线程间隔内部分配它们。利用这样的间隔分节，用不同或者相等存储桶间隔长度动态划分整个存储桶缓冲器。

[0059] 这里针对指令存储桶以及针对线程引入继承性矢量概念。每个指令矩阵/块/存储桶向在架构寄存器之中的特定寄存器中写入。每个新存储桶在分配阶段时向这一矢量中写入它自己的线程和存储桶编号而更新这一继承性矢量、留下它没有写入的寄存器的字段而不更新。按照程序顺序从每个存储桶向下一个存储桶转发这一存储桶继承性矢量B_iv 856。在图2中，如果每个矩阵中的指令向架构目的寄存器中写入，则该矩阵向那些寄存器中写入它自己的编号，否则它从该线程中的先前存储桶的B_iv继承该值。

[0060] 图3示出根据本发明的一个实施例的示例硬件电路的图，该图示出存储具有互连

的操作数和结果的分段式寄存器文件。图3示出经由执行全局互连而耦合到多个执行单元的操作数结果缓冲器。

[0061] 图4示出根据本发明的一个实施例的描绘全局前端调度器的图。全局前端调度器被配置用于处理可以具有更少紧密交互的更大量线程(例如图1B中所示全局前端调度器150)。该图示出如何跨越许多虚拟核来分布来自一个逻辑核的指令序列。这一过程将针对机器中存在的每个逻辑核进行重复。应当注意图4的“引擎”包括虚拟核的部件,其中显式地描绘寄存器文件以示出在寄存器文件级的虚拟核间通信的各方面。

[0062] 例如如图4中所描绘的,全局前端调度器可以处理线程头部902、但是无需处理线程内的实际指令以跨越那些远线程实行依赖性检查。线程的头部和它的存储桶的子头部仅包含关于那些线程和存储桶写入到的架构寄存器(那些指令的目的寄存器)的信息、在那些头部中无需包括那些指令的实际指令或者源。实际上,列举那些目的寄存器或者位矢量是足够的,在该位矢量中针对用于指令的目的的每个寄存器设置每一个位。该头部无需物理地放置为指令的头部;它可以是可以与其余指令信息一起存储或者不与其余指令信息一起存储的、在线程内的指令的目的寄存器的任何格式化分组或者紧凑表示。

[0063] 这一全局前端按照程序顺序仅提取线程/块的头部并且生成动态线程和/或存储桶继承性矢量901(Tiv和/或Biv)。每当分配新线程时,如903所示通过保持当前线程存储桶不会写入到或者更新的旧字段来转发那些继承性矢量。那些继承性矢量被分布到大量引擎/核/或者处理器904,这些引擎/核/或者处理器中的每个引擎/核/或者处理器可能包括本地前端和提取单元(该本地前端和提取单元将提取和存储针对每个存储桶产生依赖性矢量的实际指令)和具有本地寄存器文件905的本地矩阵/块/存储桶缓冲器。本地前端然后提取实际指令并且使用从全局前端获得的继承性矢量的信息以针对那些被带入引擎用于执行的指令的指令源来填充依赖性信息。图3图示了全局前端实现方式和它仅使用关于指令的简洁信息(该信息仅为那些指令写入到的寄存器)来向不同引擎904散播继承性矢量的方式。放置于头部中有帮助的其它信息是关于在线程内或者跨越线程的控制路径的变化的信息。全局分支预测器可以用来预测跨越那些线程的控制流。因此,这样的头部可以包括分支目的和偏移。除了分支预测器确定控制流之外,硬件/编译器还可以判决跨越分支的2个控制路径来分派独立线程。在这样的情况下,它将随后使用继承性矢量来合并那些2个路径的执行。图3也示出在前端提取新线程的头部时的转发过程,例如线程2(906)将更新向它转发的对应继承性矢量901从而产生矢量910,在该矢量910中用T2标签更新寄存器1、2、3、4、6、0和7。注意在910中T2存储桶未写入寄存器5,因此T2存储桶的标签从先前继承性矢量继承。

[0064] 一个感兴趣的观察是寄存器文件允许在核/引擎之中的交叉通信。一旦在填充源依赖性信息时提取并且在本地存储桶缓冲器中分配线程的指令存储桶就可以提出来自交叉引擎的需要的寄存器的(用于减少访问延时的)早期请求,从而可以在分派实际指令用于执行之前可能很久的时间发布交叉引擎线程引用。在任何情况下,指令直至转发和到达交叉引用的源才将被分派。这一交叉引用的源可以存储于本地多线程式寄存器文件或者寄存器高速缓存中。虽然这一交叉引用的源(可以重用加载存储缓冲器物理存储装置和依赖性检查机制而)可以存储于与加载存储缓冲器相似的缓冲器中,但是它是作为寄存器加载而不是存储器加载。许多技术可以用来跨越引擎/核连接寄存器文件,这可以是环形拓扑或者纵横拓扑或者网状路由式互连。

[0065] 以下讨论可以举例说明如何可以在引擎内部并且也跨越引擎使用寄存器文件分割。在分派存储桶时，向寄存器文件和寄存器高速缓存二者(同时或者依次)发送它的源。如果寄存器文件被物理地统一并且具有针对线程化的直接支持，则从对应线程寄存器分节直接读取操作数。如果寄存器文件是包括使用标记的物理上片段的寄存器文件的虚拟寄存器，则必须完成将标记匹配为虚拟寄存器读取的部分。如果该标记匹配，则从片段的寄存器文件发生读取。

[0066] 公开了一种支持软件线程、硬件生成的线程、VLIW执行、SIMD和MIMD执行以及无序超标量执行的仿真的寄存器架构。虽然它被物理地片段，但是它看来是统一架构资源。这一片段的寄存器是可以包括寄存器分级和寄存器高速缓存以及用于存储和检查寄存器标记的机制的虚拟寄存器文件的部分。如果我们使用利用依赖性继承性矢量的基于位置的方案则可以消除标记访问。该方案工作使得在分派阶段期间对执行的存储桶编号进行广播时后续指令的所有源执行CAM(内容可寻址匹配)，该CAM将它们的源存储桶与刚才分派/执行的存储桶进行比较以设置针对该源的就绪标志。这里，该存储桶执行之处的物理位置也可以与寄存器编号一起传播，从而解决语意模糊。

[0067] 例如考虑有4个寄存器文件段的实现方式，每个寄存器文件段包含16个寄存器。例如在向分节2分派存储桶#x时，将该存储桶编号x向存储桶缓冲器广播并且还随着它广播段#2，从而对存储桶x具有依赖性的所有源将记录有它在段2中写入所有它的寄存器。在分派那些指令的时间到来时，它们知道它们需要从段2而不是任何其它段读取它们的寄存器，即使在其它段中存在相同寄存器编号。这也适用于寄存器高速缓存以避免使用标记。我们可以将这一概念延伸到全局前端，在该全局前端中，除了线程信息之外，继承性矢量还可以指定在哪个引擎中分配向这一寄存器写入的指令存储桶。

[0068] 图5示出根据本发明的一个实施例的跨越许多虚拟核的指令分布的备选实现方式。图5示出通过向虚拟核分布继承性矢量编码段来工作的运行时间优化器调度器550。在一个实施例中，优化器关注多个指令代码块并且跨越所有代码块重新调度指令以创建代码段和继承性矢量。优化器的目标将是使代码段在它们的相应虚拟核上重叠执行的执行效率最大化。

[0069] 图6示出根据本发明的一个实施例的具有对应多个寄存器文件以及操作数结果缓冲器的多个寄存器段。如图6中所描绘的，执行全局互连将每个寄存器段连接到多个地址计算和执行单元。

[0070] 图6中的寄存器段可以用来实施3个执行模式之一：通过由编译器/编程器分组在一起以形成MIMD超指令矩阵，或者可以在单个线程在4个硬件分节中的每个硬件分节上同时执行的线程式模式中独立执行每个矩阵。最后一个可能执行模式是有能力使用硬件依赖性检查来动态执行来自单个线程的4个不同指令矩阵以确保在4个不同硬件分节上同时执行的那些不同矩阵之间没有依赖性存在。

[0071] 可以根据执行模式交替地配置图6中的寄存器文件。在一个模式中，寄存器文件视为服务于MIMD宽度的4个分节的MIMD分节式寄存器文件或者寄存器文件用为4个单独的寄存器文件，每个寄存器文件服务于单个线程。寄存器文件也可以支持动态执行模式，在该动态执行模式中4个分节是一个统一寄存器文件，在该统一寄存器文件中向特定分节中的任何寄存器写入的数据可由其它分节中的所有单元访问。在那些模式之间切换可以无缝，因

为不同执行模式可以在每一个线程基线指令矩阵与MIMD超指令矩阵线程之间交替。

[0072] 在多线程执行模式中，每个寄存器文件和它的执行线程的执行单元完全独立于其它寄存器文件及其线程。这与每个线程具有它自己的寄存器状态相似。然而，可以指定在那些线程之间的依赖性。属于线程的每个矩阵将在该线程的寄存器文件的执行单元中执行。如果在硬件上仅执行一个线程或者单个非线程式程序，则以下方法用来允许属于该单个线程/程序的并行矩阵能够访问向其它分节中的寄存器中写入的结果。做到这一点的方式是通过允许向4个寄存器文件分节中的任何寄存器文件分节中写入结果的任何矩阵生成那些寄存器在其它寄存器文件分节中的副本。在物理上，这通过将每个分节的写入端口延伸到剩余分节中来做到。然而这不可升级，因为我们不能构建高效寄存器文件而每个存储器单元具有如一个分节独自所需写入端口的多达4倍。我们呈现构建寄存器文件使得这样的单线程寄存器广播延伸将不影响它的机制。

[0073] 应当注意可以在Mohammad A.Abdallah于2007年11月14日提交的、名称为"APPARATUS AND METHOD FOR PROCESSING COMPLEX INSTRUCTION FORMATS IN A MULTITHREADED ARCHITECTURE SUPPORTING VARIOUS CONTEXT SWITCH MODES AND VTRTUALTZATTON SCHEMES"的、美国专利申请公开第2010/0161948号中发现关于如在本发明的实施例中使用的寄存器段的各附加方面。

[0074] 图7示出根据本发明的一个实施例的用于多核处理器的片段式存储器子系统的更具体图。图7示出在线程之中和/或一般在加载和存储之中的同步方案的全面方案和实现方式。该方案描述一种用于跨越加载/存储架构和/或跨越存储器引用和/或线程的存储器访问的存储器引用的同步和消歧的优选方法。在图7中，我们示出寄存器文件(地址和/或数据寄存器)的多个段、执行单元、地址计算单元和1级高速缓存的片段和/或加载存储缓冲器和2级高速缓存以及地址寄存器互连1200和地址计算单元互连1201。那些片段式单元可以在一个核/处理器内通过将它的集中式资源片段和分布到若干引擎中来构造，或者它们可以由在多核/多处理器配置中的不同核/处理器的单元构造。在该图中示出那些片段1211之一为片段编号1；片段可以升级成大量(一般如该图中所示升级成N个片段)。

[0075] 这一机制也用为针对在那些引擎/核/处理器之中的存储器架构的相干方案。这一方案通过来自一个片段/核/处理器中的一个地址计算单元的地址请求来开始。例如，假设片段1请求地址(1211)，则它可以使用属于它自己的片段的地址寄存器和/或使用地址互连总线1200从跨越其它片段的寄存器来获得和计算它的地址。在计算地址之后，它创建用来访问高速缓存和存储器的32位地址或者64位地址的参考地址。这一地址通常被片段成标记字段以及设置和线字段。这一特定片段/引擎/核将该地址存储到它的加载存储缓冲器和/或L1地址阵列和/或L2地址阵列1202中，同时它将通过使用压缩技术来创建标记的压缩版本(具有比地址的原有标记字段更少的位数)。

[0076] 另外，不同片段/引擎/核/处理器将使用设置字段或者设置字段的子集作为用于标识在哪个片段/核/处理器中维护地址的索引。地址设置字段对片段的这一索引化确保了地址在特定片段/核/引擎中的所有权独占，即使与该地址对应的存储器数据可以在另一个或者多个其它片段/引擎/核/处理器中存活。即使在每个片段中示出将地址CAM/标记阵列1202/1206与数据阵列1207耦合，但是它们可以仅以物理邻近的放置和布局的方式耦合或者甚至事实是二者属于特定引擎/核/处理器这样，但是在地址阵列中保持的地址与在一个

片段内部的数据阵列中的数据之间没有关系。

[0077] 图8示出描绘根据本发明的一个实施例的通过地址生成如何使用地址的位以枚举片段的图。在本实施例中，如图8中所描绘，片段由在页面边界以上的地址位和在高速缓存线边界以下的地址位定义。本发明有利地保持于页面边界以上以避免在从虚拟地址转译成物理地址期间引起TLB错失。该过程保持于高速缓存线边界以下以便具有完整高速缓存线以便在硬件高速缓存分级内正确相配。例如，在运用64字节高速缓存线的系统中，片段边界将避免后六个地址位。为比较，运用32字节高速缓存线的系统，片段边界将避免后五位。一旦定义，片段分级跨越处理器的所有高速缓存分级相同。

[0078] 图9示出本发明的实施例的如何处置加载和存储的图。如图9中所描绘，每个片段与它的加载存储缓冲器和存储引退缓冲器关联。对于任何给定的片段，向该片段的加载存储缓冲器发送指明与该片段或者另一片段关联的地址范围的加载和存储用于处理。应当注意，它们可以无序地到达，因为核无序地执行指令。在每个核内，核具有不仅对它自己的寄存器文件而且对其它核的寄存器文件中的每个寄存器文件的访问。

[0079] 本发明的实施例实施一种分布式加载存储排序系统。该系统跨越多个片段分布。在片段内，该片段执行本地数据依赖性检查。这是因为片段仅在该特定片段的存储引退缓冲器内加载和存储。这限制了对必须关注其它片段以维护数据相干性的需要。以这一方式，将片段内的数据依赖性进行本地实行。

[0080] 关于数据一致性，存储分派门根据严格的按照程序顺序的存储器一致性规则实行存储引退。存储无序地到达加载存储缓冲器。加载也无序地到达加载存储缓冲器。同时，无序加载和存储被转发到存储引退缓冲器用于处理。应当注意虽然在给定的片段内依序引退存储，但是在它们去往存储分派门时它们可以无序地来自多个片段。存储分派门实行如下策略，该策略确保即使存储可以无序地跨越存储引退缓冲器而驻留，并且即使缓冲器可以相对于其它缓冲器的存储而向存储分派门无序地转发存储，分派门仍然确保向片段存储器严格依序转发它们。这是因为存储分派门具有存储引退的全局了解并且仅允许存储跨越所有片段——即全局——依序离开去往存储器的全局可视侧。以这一方式，存储分派门作为全局观察器工作以确保存储跨越所有片段最终依序返回到存储器。

[0081] 图10示出根据本发明的一个实施例的可以将片段拆分成两个或者更多域的方式。图10示出可以将单个片段拆分成多个域的方式。域拆分可以经由地址生成过程来实施。域拆分改变必须在片段内完成加载存储检查的方式，因为在这一情况下它们必须仅按域来完成，这有别于跨越整个片段。域拆分还有利在于它可以使单端口式存储器表现如同多端口存储器，其中将单个端口访问每个不同的域。

[0082] 图11示出根据本发明的一个实施例的处理器的操作模式，在该操作模式中，可分割引擎的硬件资源在执行应用中如同逻辑核工作。在这一实施例中，将虚拟核的引擎的硬件资源配置为物理核。在图11的模式中，每个物理核被配置为作为逻辑核工作。多线程式应用和多线程式功能取决于应用的软件的线程式可编程性。

[0083] 图12示出根据本发明的一个实施例的处理器的操作模式，在该操作模式中，软件核用来在执行应用中如同逻辑核工作。在这一实施例中，虚拟核的可分割引擎将支持多个软核。在图12的模式中，每个软核被配置为作为逻辑核工作。多线程式应用和多线程式功能取决于应用的软件的线程式可编程性。

[0084] 图13示出根据本发明的一个实施例的处理器的操作模式，在该操作模式中，软核用来在执行应用中如同单个逻辑核工作。在图13的模式中，每个软核被配置为作为单个逻辑核工作。在这样的实现方式中，单个线程式应用让它的指令序列被划分并且在虚拟核之中分配，其中它们被协同地执行以实现高的单线程式性能。以这一方式，单线程式性能可以随着添加附加软核而升级。

[0085] 可以在选择处理器的操作模式时使用多个策略。对于具有大量引擎(例如8个引擎、12个引擎等)的处理器，多个软核可以被配置为作为单个逻辑核工作，而剩余核可以在其它模式中操作。这一属性允许智能资源分割以确保硬件的利用最大化和/或浪费的功耗最小化。例如，在一个实施例中，可以根据执行的应用类型来在每线程基础上分配核(例如软核或者逻辑核)。

[0086] 图14示出根据本发明的一个实施例的用来支持逻辑核和虚拟核功能的片段分割的示例实现方式。如以上讨论的那样，片段分割允许处理器被配置为支持如以上描述的不同虚拟核执行模式。

[0087] 全局互连允许核的线程访问端口1401中的任何端口。应当注意，这里所用术语“线程”是指来自不同逻辑核的指令序列、来自相同逻辑核的指令序列或者二者的某种混合的表示。

[0088] 如图所示根据仲裁器的策略可调节线程利用端口1401之一以访问加载存储缓冲器的方式。因此，使用端口1401中的任何端口的线程可以具有经由端口1402对加载存储缓冲器的更大或者更少量访问。分配的大小和管理分配的方式由仲裁器控制。该仲裁器可以根据特定线程的需求而动态分配对端口的访问。

[0089] 加载存储缓冲器被配置为让多个条目跨越端口展开。对加载存储缓冲器的访问由仲裁器控制。以这一方式，仲裁器可以向不同线程动态地分配加载存储缓冲器中的条目。

[0090] 图14还示出在加载存储缓冲器与L1高速缓存之间的端口上的仲裁器。因此，与以上描述的加载存储缓冲器一样，使用端口1403中的任何端口的线程可以具有经由端口1404对L1高速缓存的更大或者更少量访问。分配的大小和管理分配的方式由仲裁器控制。该仲裁器可以根据特定线程的需求而动态地分配对端口的访问。

[0091] L1高速缓存被配置为让多个方式跨越端口展开。对L1高速缓存的访问由仲裁器控制。以这一方式，仲裁器可以向不同线程动态地分配L1高速缓存中的条目。

[0092] 在一个实施例中，仲裁器被配置以与用于跟踪功能的多个跟踪计数器1460和提供限制功能的多个阈值寄存器1450工作。限制功能针对每个给定的线程指定最大资源分配百分比。跟踪功能跟踪在任何给定的时间向给定的线程分配的实际资源。这些跟踪和限制功能影响用于加载存储缓冲器、L1高速缓存、L2高速缓存或者全局互连的每线程条目、路线或者端口的数目。例如可以对比可变阈值而动态地检查针对每个线程分配的在加载存储缓冲器中的条目总数。这一可变阈值可以根据给定的线程的转发进度来更新。例如在一个实施例中，将减缓的线程(例如大量L2错失等)量化为产生缓慢向前进度，因此降低它们的包括条目阈值、路线阈值和端口阈值的相应资源分配阈值。

[0093] 图14也示出共享L2高速缓存。在本实施例中，共享的L2高速缓存具有固定的端口布置而没有在来自L1高速缓存的访问之间的任何仲裁。在处理器上执行的线程将都共享对L2高速缓存和L2高速缓存的资源的访问。

[0094] 图15示出根据本发明的一个实施例的实施多物理到多逻辑模式的示例性四片段处理器的片段存储器。

[0095] 图15上的阴影示出一个示例性逻辑核及其与处理器的资源的关系。在图11的操作模式(多物理核到多逻辑核的模式,其中物理核用来在执行应用中如同逻辑核而工作)中,每个逻辑核将被配置为具有固定的加载存储缓冲器和L1高速缓存的资源比值。端口可以向每个线程或者核具体指派。可以每线程或者核具体保留加载存储缓冲器中的条目。可以每线程或者核具体保留L1高速缓存内的路线。多线程式应用和多线程式功能取决于应用的软件的线程式可编程性。这由一个逻辑核具有分配的端口以及片段中的每个片段的存储缓冲器和L1高速缓存的分配部分而示出。以这一方式,逻辑核包括固定的每个片段的资源的分配片段。

[0096] 在一个实施例中,在多物理核到多逻辑核模式中,可以根据访问每个片段的端口(端口1401)的数目对四个片段进行分割。例如在每片段有六个端口的实施例中,可以用跨越4个片段和4个分割双引擎来支持6个物理核这样的方式对每个片段的资源进行划分并且因此划分引擎的每个分割的资源。每个分割可以被分配它自己的端口。类似地,将以支持6个物理核这样的方式而分配加载存储缓冲器和L1高速缓存的资源。例如,在其中加载存储缓冲器具有48个条目的实施例中,可以分配48个条目使得每物理核有12个条目以支持实施4个物理核的模式,或者可以分配它们使得每物理核有8个条目以支持实施6个物理核的模式。

[0097] 图16示出根据本发明的一个备选实施例的实施多物理到多逻辑模式的示例性四片段处理器的片段存储器。

[0098] 同图15,图16上的阴影示出一个示例逻辑核及其与处理器的资源的关系。在图11的操作模式(多物理核到多逻辑核模式)中,整个分割表引擎专用于支持单个逻辑核的执行。这由图16中的阴影示出。物理资源引擎用来在执行应用中如同逻辑核工作。

[0099] 图17示出根据本发明的一个实施例的实施多软核到多逻辑核模式的示例性四片段处理器的片段存储器。

[0100] 图17上的阴影示出一个示例性逻辑核及其与处理器的资源的关系。在图12的操作模式(多软核到多逻辑模式,其中虚拟核用来在执行应用中如同逻辑核工作)中,加载存储缓冲器的资源的分配的大小和管理分配的方式由仲裁器控制。该仲裁器可以根据特定线程或者核的需求来动态地分配对端口的访问。类似地,L1高速缓存的资源的分配的大小和管理分配的方式由仲裁器控制。该仲裁器可以根据特定线程或者核的需求来动态地分配对端口的访问。因此,在任何给定的实例,(例如有阴影的)逻辑线程/核可以使用不同仲裁器和不同端口。

[0101] 以这一方式,对加载存储缓冲器的资源的访问和对L1高速缓存的资源的访问可以更由策略驱动并且可以更基于产生向前进度的个别线程或者核的需要。这由一个逻辑核具有动态分配的端口以及片段中的每个片段的存储缓冲器和L1高速缓存的动态分配部分示出。以这一方式,逻辑核包括每个片段的资源的非固定、动态分配片段。

[0102] 图18示出根据本发明的一个实施例的实施多软核到一个逻辑核模式的示例性四片段处理器的片段存储器。

[0103] 在图13的操作模式(多软核到一个逻辑核模式,其中软核用来在执行应用中如同

单个逻辑核工作)中,软核中的每个软核被配置以与其它软核一起作为单个逻辑核协同地工作。单个线程或者核具有加载存储缓冲器的所有资源和L1高速缓存的所有资源。在这样的实现方式中,单个线程式应用让它的指令序列被划分并且在软核之中分配,其中它们被协同地执行以实现高的单线程式性能。以这一方式,单线程式性能可以随着添加附加软核而升级。在图17中示出这一点,其中处理器的所有资源的阴影示出一个示例逻辑核及其与处理器的资源的关系。

[0104] 图19示出根据本发明的一个实施例的实施多物理到多逻辑模式的示例性四片段处理器的地址计算和执行单元、操作数/结果缓冲器、线程式寄存器文件以及公共分割调度器。

[0105] 图19上的阴影示出一个示例性逻辑核及其与处理器的资源的关系。在图11操作模式(多物理核到多逻辑核模式,其中物理核用来在执行应用中如同逻辑核工作)中,每个逻辑核将被配置为具有固定的地址计算单元、操作数/结果缓冲器、线程式寄存器文件和公共分割调度器的资源比值。多线程式应用和多线程式功能取决于应用的软件的线程式可编程性。这由一个逻辑核具有分配的地址计算和执行单元、分配的线程式寄存器文件以及分配的公共分割调度器而示出。以这一方式,逻辑核包括固定的分配段。然而在一个实施例中,在这一操作模式中,仍然可以共享地址计算和执行单元(例如意味着地址计算和执行单元中的每个地址计算和执行单元将无阴影)

[0106] 图20示出根据本发明的一个实施例的用来实施多物理到多逻辑模式的示例性四片段处理器的地址计算和执行单元、操作数/结果缓冲器、线程式寄存器文件以及公共分割调度器的备选实现方式。

[0107] 图20上的阴影示出一个示例性逻辑核及其与处理器的资源的关系。然而在图20的实施例中,跨越片段中的每个片段和可分割引擎中的每个可分割引擎展开物理核的资源。这由一个逻辑核跨越段中的每个段具有地址计算和执行单元的分配部分、线程式寄存器文件的分配部分以及公共分割调度器的分配部分而示出。此外,图20示出将如何已经向一个逻辑核分配地址计算执行单元中的每个地址计算执行单元的资源的部分。以这一方式,逻辑核包括固定的段中的每个段的分配部分。

[0108] 图21示出根据本发明的一个实施例的实施多软核到多逻辑核模式的示例性四片段处理器的地址计算和执行单元、寄存器文件以及公共分割调度器。

[0109] 图21上的阴影示出一个示例性逻辑核及其与处理器的资源的关系。在图12的操作模式(多软核到多逻辑核模式,其中软核用来在执行应用中如同逻辑核工作)中,每个逻辑核将被配置为具有对地址计算单元以及操作数/结果缓冲器、线程式寄存器文件和公共分割调度器的动态分配部分中的任一项的共享访问。多线程式应用和多线程式功能取决于应用的软件的线程式可编程性。

[0110] 图22示出根据本发明的一个实施例的实施多软核到一个逻辑核模式的示例性四片段处理器的地址计算和执行单元、寄存器文件以及公共分割调度器。

[0111] 图22上的阴影示出一个示例性逻辑核及其与处理器的资源的关系。在图13的操作模式(多软核到一个逻辑核模式,其中软核用来在执行应用中如同单个逻辑核操作)中,每个软核将被配置为具有对所有地址计算单元以及所有操作数/结果缓冲器、线程式寄存器文件和公共分割调度器的共享访问。在这样的实现方式中,单个线程式应用让它的指令序

列被划分并且在虚拟核之中分配，其中它们被协同地执行以实现高的单线程式性能。以这一方式，单线程式性能可以随着添加附加软核而升级。

[0112] 图23示出根据本发明的一个实施例的示例性微处理器流水线2300的图。微处理器流水线2300包括实施如以上描述的过程的功能的提取模块2301，该过程用于标识和抽取包括执行的指令。在图23的实施例中，提取模块跟随有解码模块2302、分配模块2303、分派模块2304、执行模块2305和引退模块2306。应当注意，微处理器流水线2300仅为实施以上描述的本发明的实施例的功能的流水线的一个示例。本领域技术人员将认识可以实施包括以上描述的解码模块的功能的其它微处理器流水线。

[0113] 已经出于说明的目的参照具体实施例对前文描述进行了描述。然而以上所示讨论并非旨在详尽的或者使本发明限于公开的精确形式。鉴于以上教导的许多修改和变化是可能的。选择和描述实施例以便最好地说明本发明的原理及其实际应用，以由此使本领域其他技术人员能够借助如可以与设想的特定使用相适应的各种修改最好地利用本发明和各种实施例。

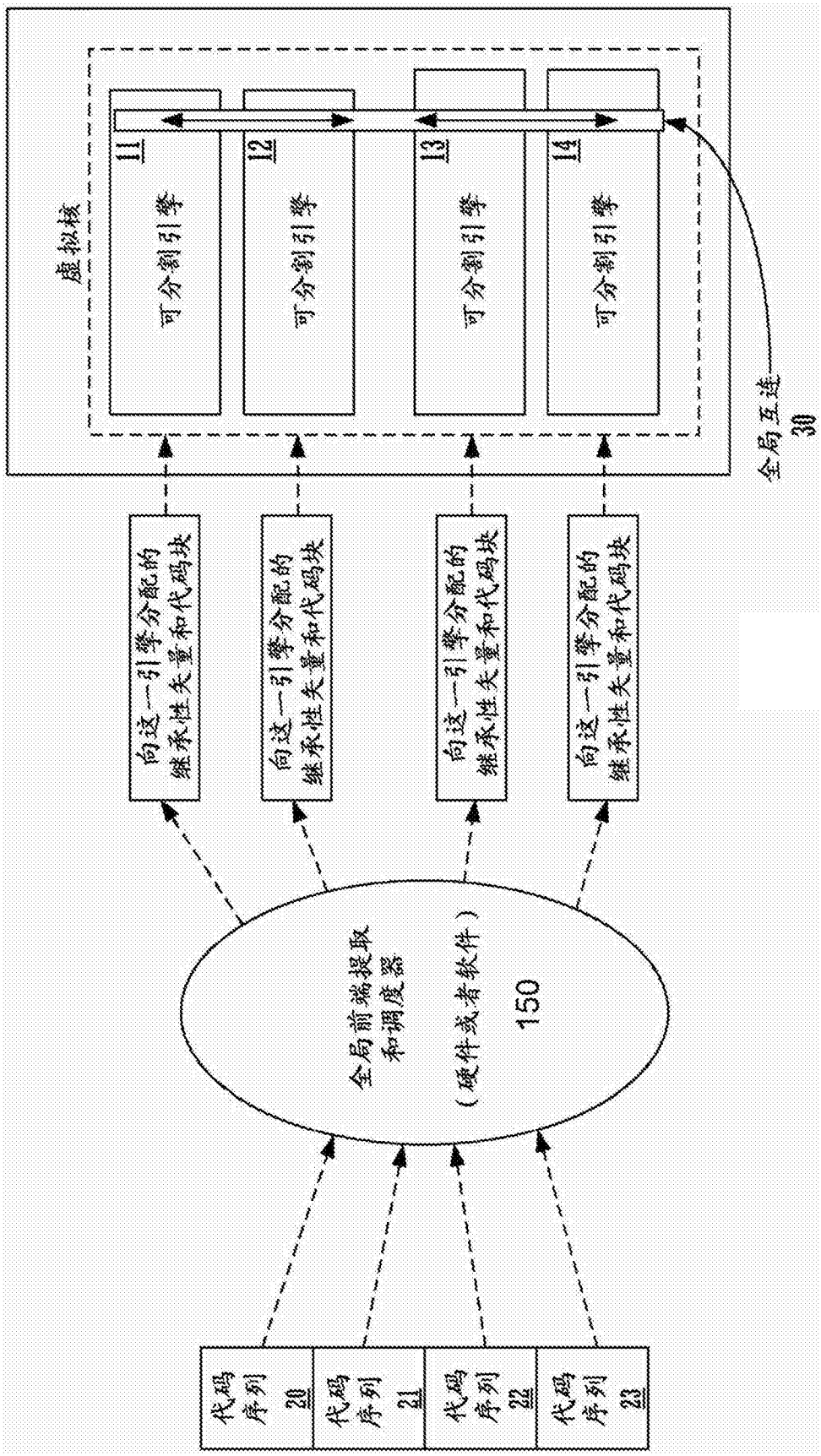

虚拟核
11
可分割引擎
12
可分割引擎
13
可分割引擎
14
可分割引擎
全局互连
30
向这一引擎分配的
继承性矢量和代码块
向这一引擎分配的
继承性矢量和代码块
向这一引擎分配的
继承性矢量和代码块
向这一引擎分配的
继承性矢量和代码块
全局前端提取
和调度器
(硬件或者软件)
150
代码
序列
20
代码
序列
21
代码
序列
22
代码
序列
23

图1A

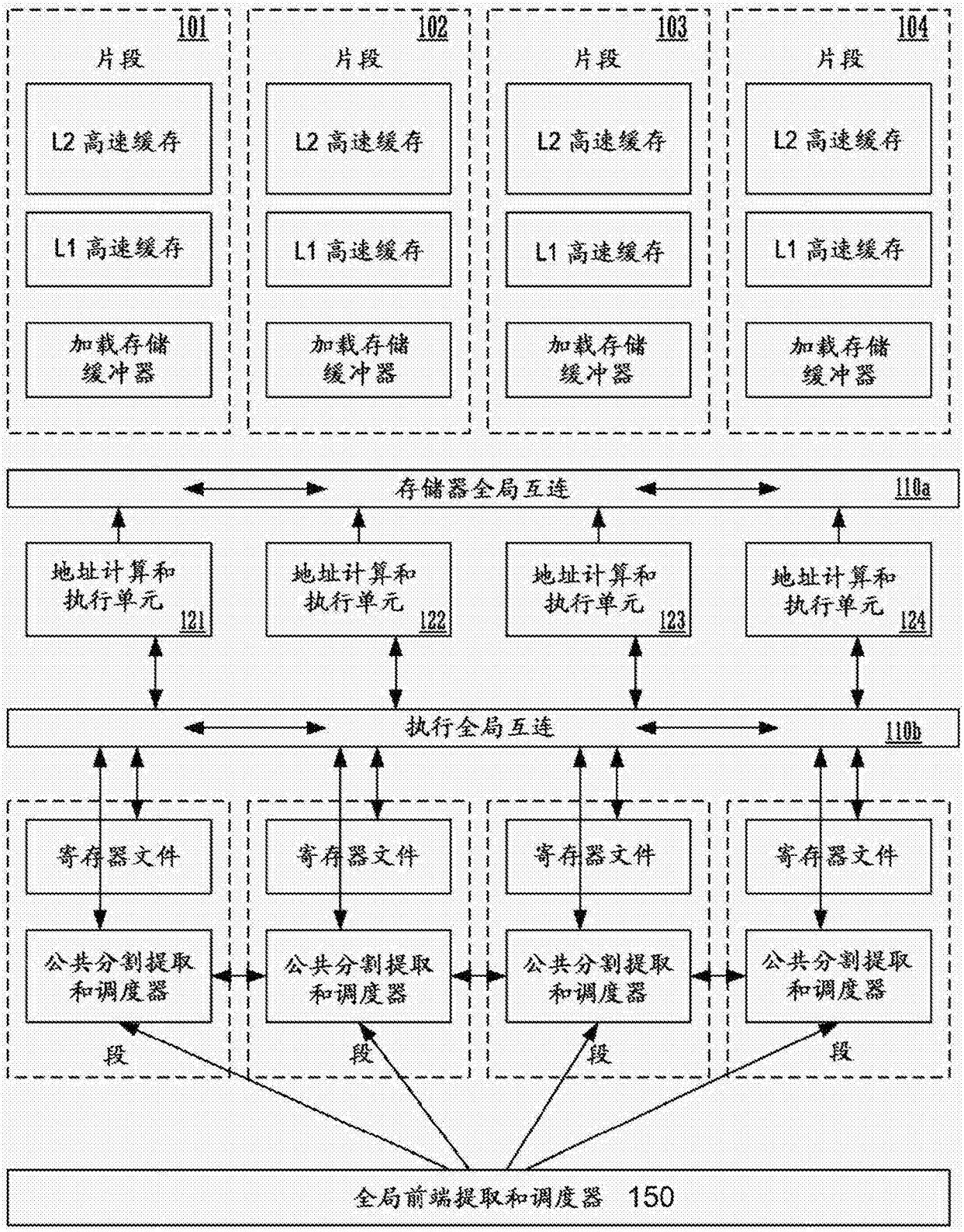

101
片段
L2 高速缓存
L1 高速缓存
加载存储
缓冲器
102
片段
L2 高速缓存
L1 高速缓存
加载存储
缓冲器
103
片段
L2 高速缓存
L1 高速缓存
加载存储
缓冲器
104
片段
L2 高速缓存
L1 高速缓存
加载存储
缓冲器
存储器全局互连
110a
地址计算和
执行单元
121
地址计算和
执行单元
122
地址计算和
执行单元
123
地址计算和
执行单元
124
执行全局互连
110b
寄存器文件
公共分割提取
和调度器
段
全局前端提取和调度器 150

图1B

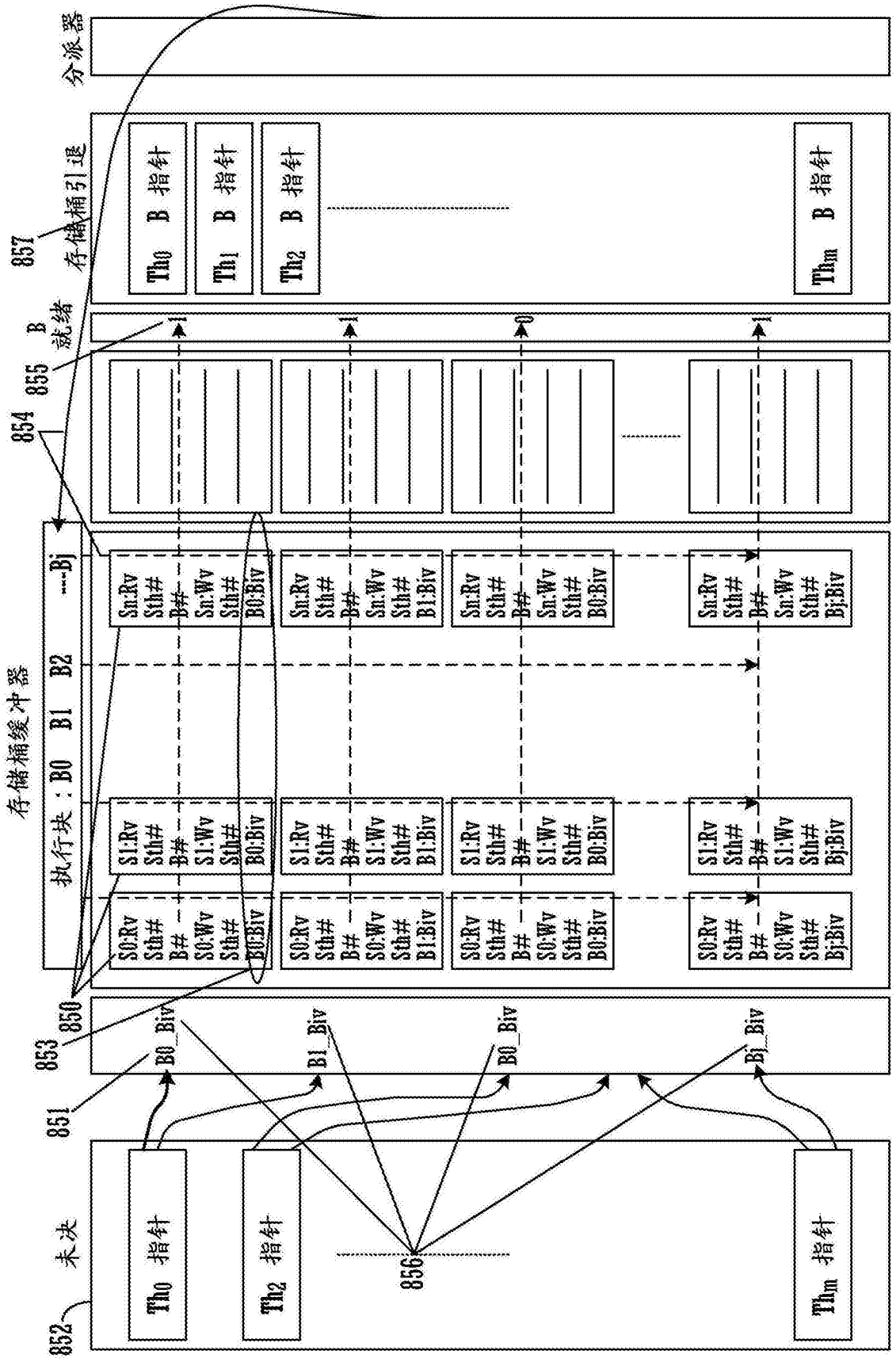

分派器
857
存储桶引退
Th0 B 指针
Th1 B 指针
Th2 B 指针
Thm B 指针
855
B 就绪
854
存储桶缓冲器
执行块：B0 B1 B2 ---Bj
Sn:Rv Sth# B#
Sn:Wv Sth# B0:Biv
S1:Rv Sth# B#
S1:Wv Sth# B1:Biv
S0:Rv Sth# B#
S0:Wv Sth# Bj:Biv
850
853
B0_Biv
B1_Biv
Bj_Biv
851
未决
Th0 指针
Th2 指针
Thm 指针
856
852

图2

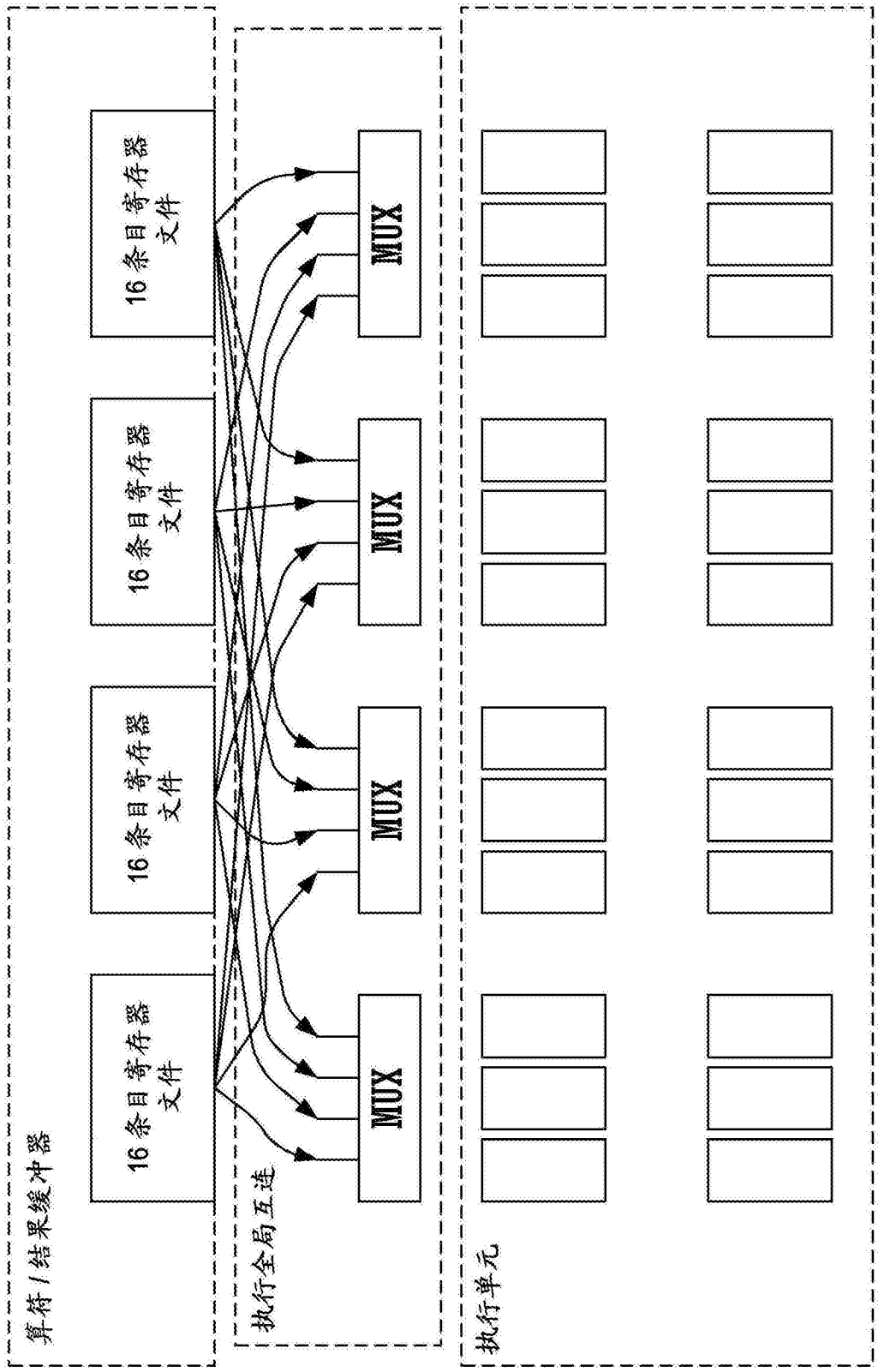

算符 / 结果缓冲器
16 条目寄存器文件
16 条目寄存器文件
16 条目寄存器文件
16 条目寄存器文件
执行全局互连
MUX
MUX
MUX
MUX
执行单元

图3

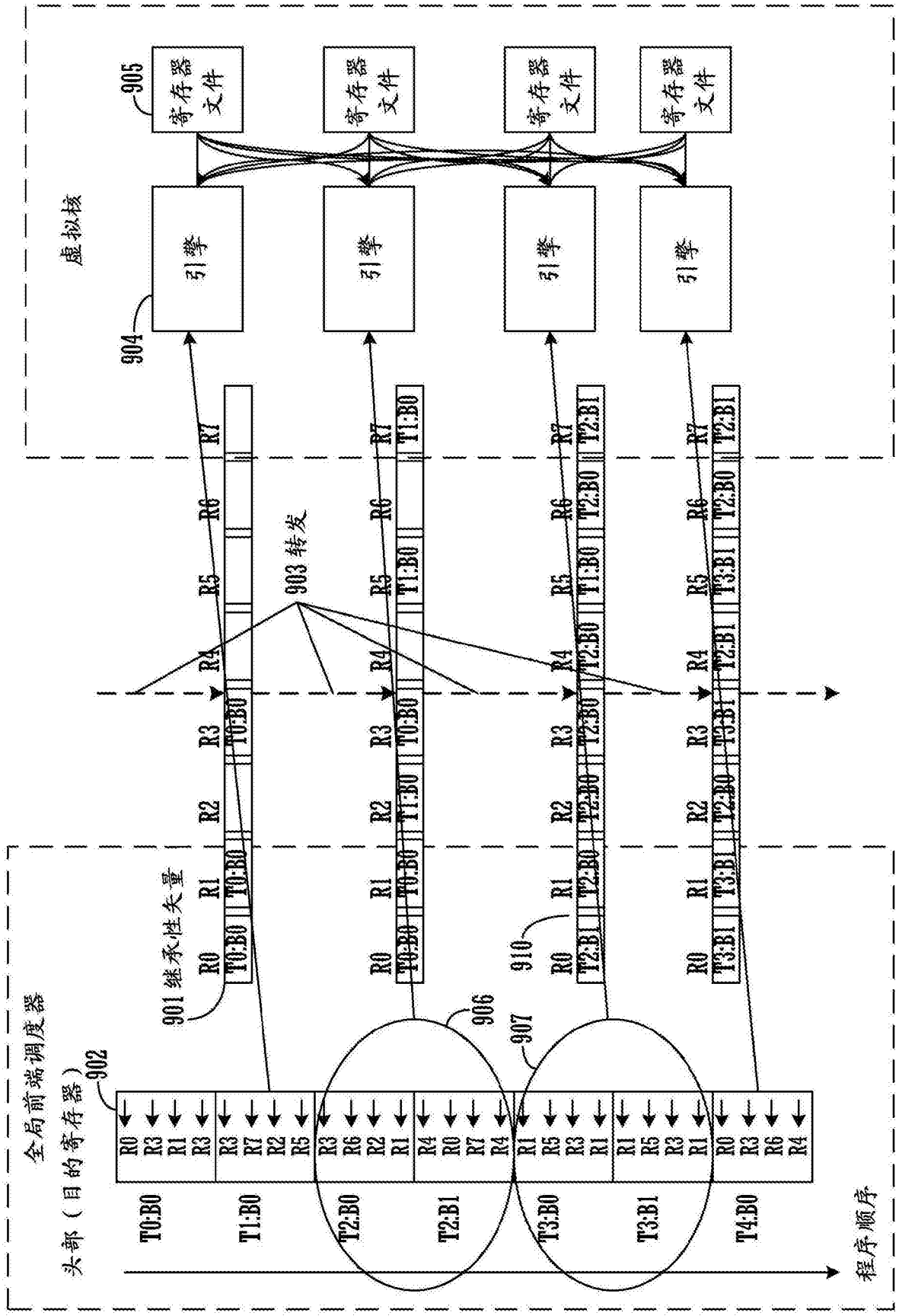

虚拟核
寄存器文件
905
引擎
904
903 转发
901 继承性矢量
910
906
907
902
全局前端调度器
头部（目的寄存器）
程序顺序
T0:B0
T1:B0
T2:B0
T2:B1
T3:B0
T3:B1
T4:B0

图4

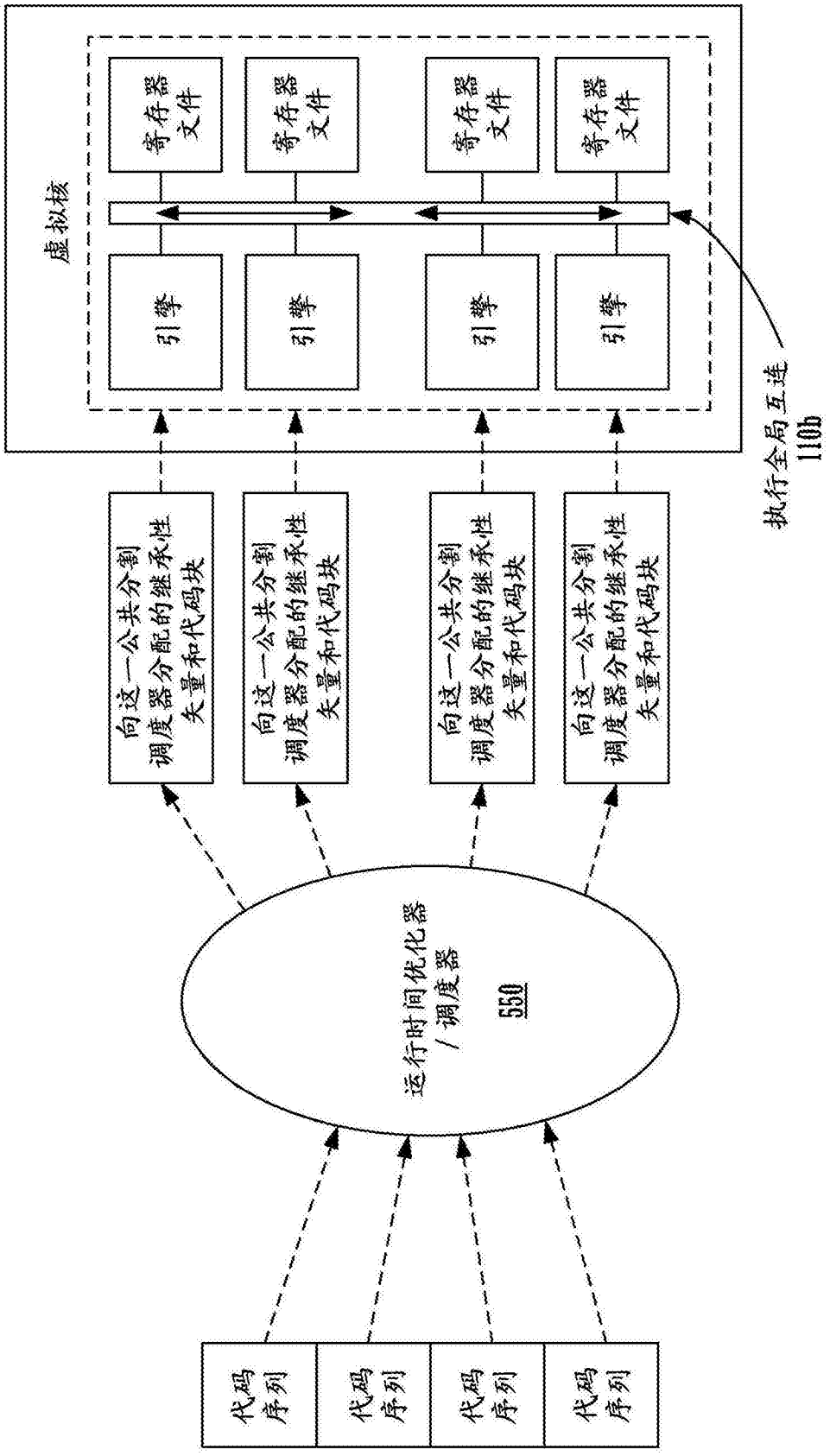

虚拟核
寄存器文件
寄存器文件
寄存器文件
寄存器文件
引擎
引擎
引擎
引擎
执行全局互连
1100
向这一公共分割调度器分配的继承性失量和代码块
向这一公共分割调度器分配的继承性失量和代码块
向这一公共分割调度器分配的继承性失量和代码块
向这一公共分割调度器分配的继承性失量和代码块
运行时间优化器/调度器
550
代码序列
代码序列
代码序列
代码序列

图5

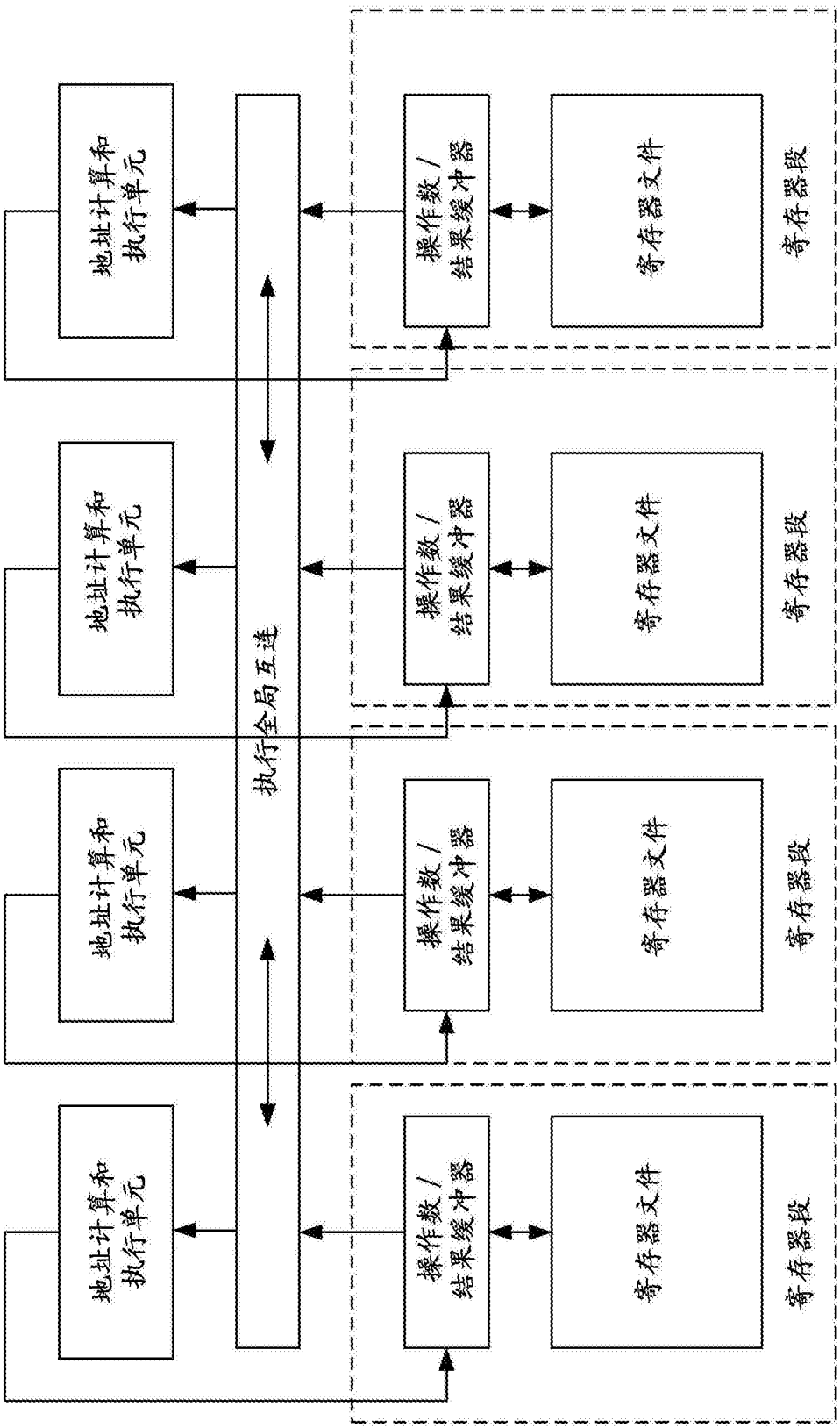


图6

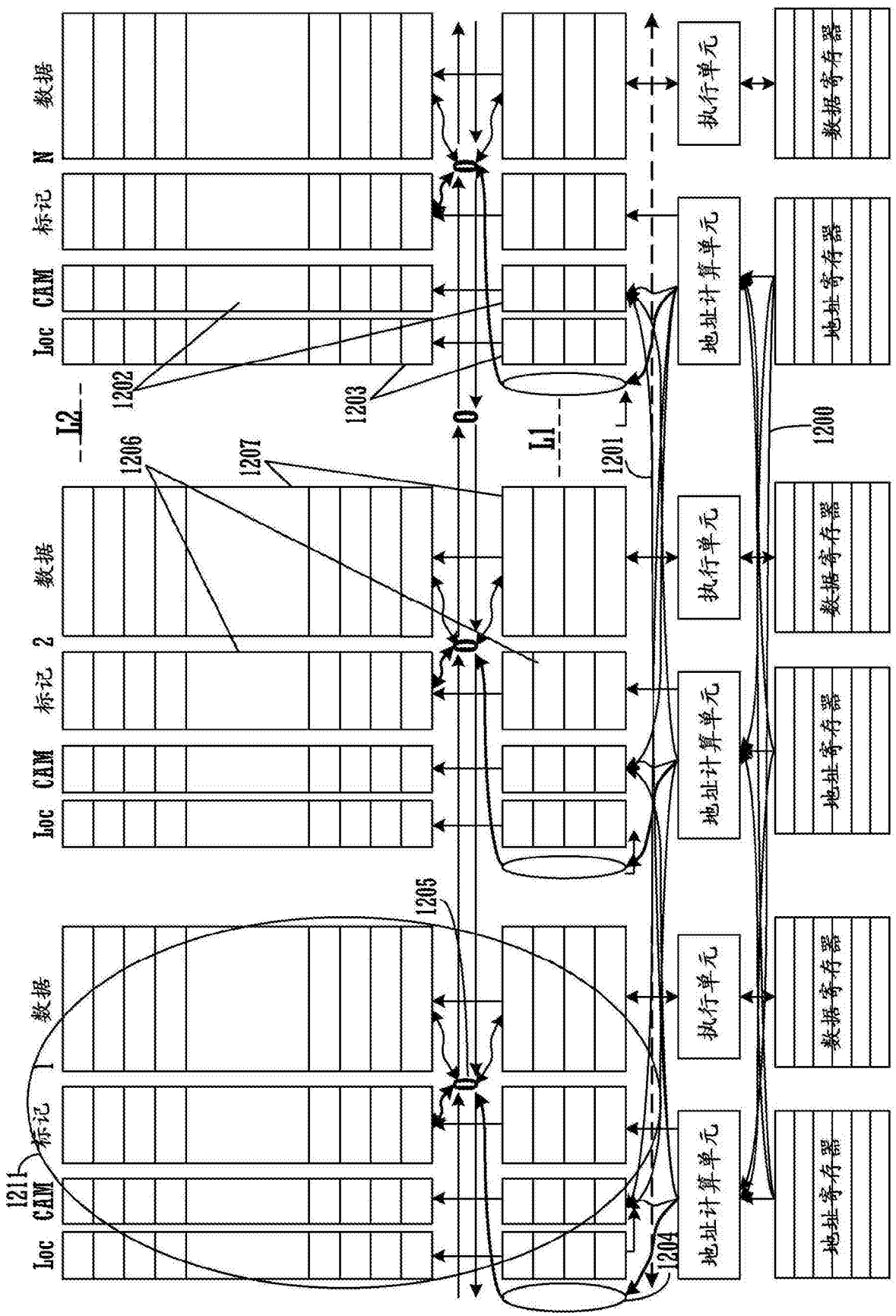

Loc
CAM
标记
数据
N
2
1
0
L2
L1
1202
1203
1206
1207
1201
1200
1205
1211
1204
执行单元
地址计算单元
数据寄存器
地址寄存器

图7

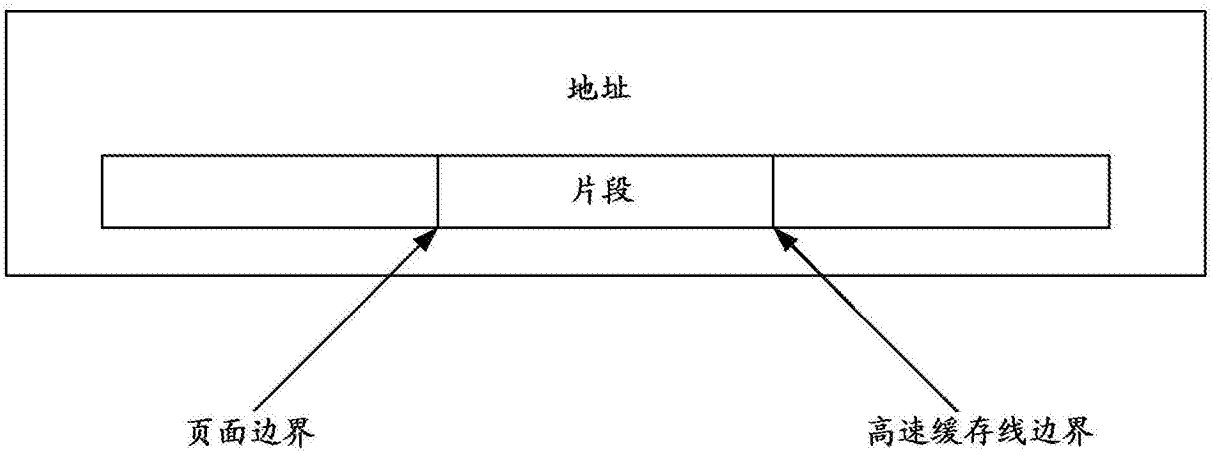


图8

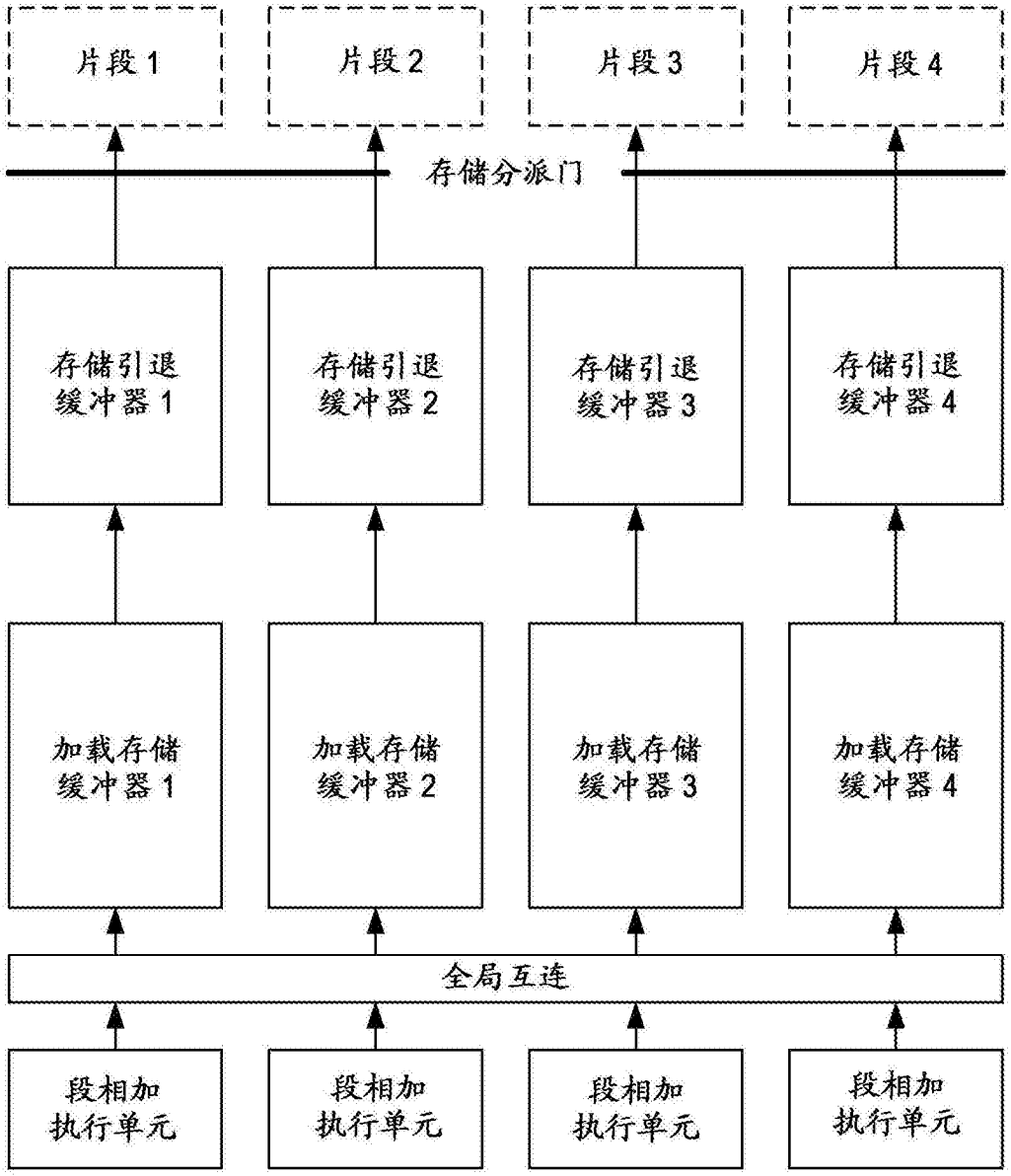

片段 1
片段 2
片段 3
片段 4
存储分派门
存储引退缓冲器 1
存储引退缓冲器 2
存储引退缓冲器 3
存储引退缓冲器 4
加载存储缓冲器 1
加载存储缓冲器 2
加载存储缓冲器 3
加载存储缓冲器 4
全局互连
段相加执行单元
段相加执行单元
段相加执行单元
段相加执行单元

图9

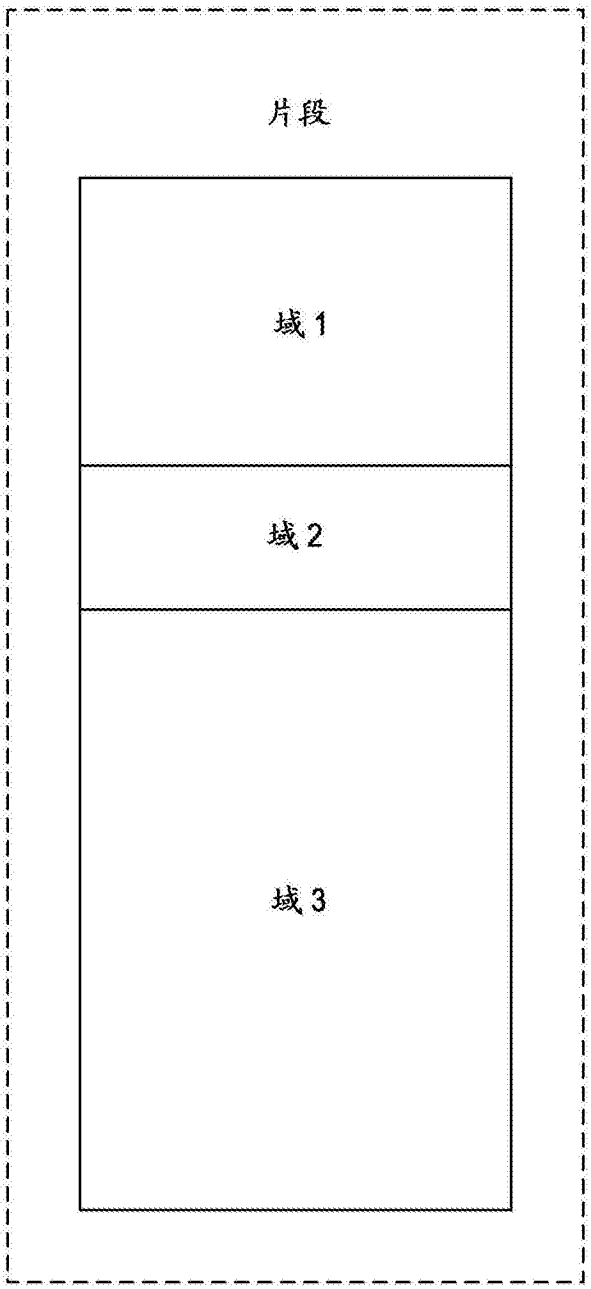


图10

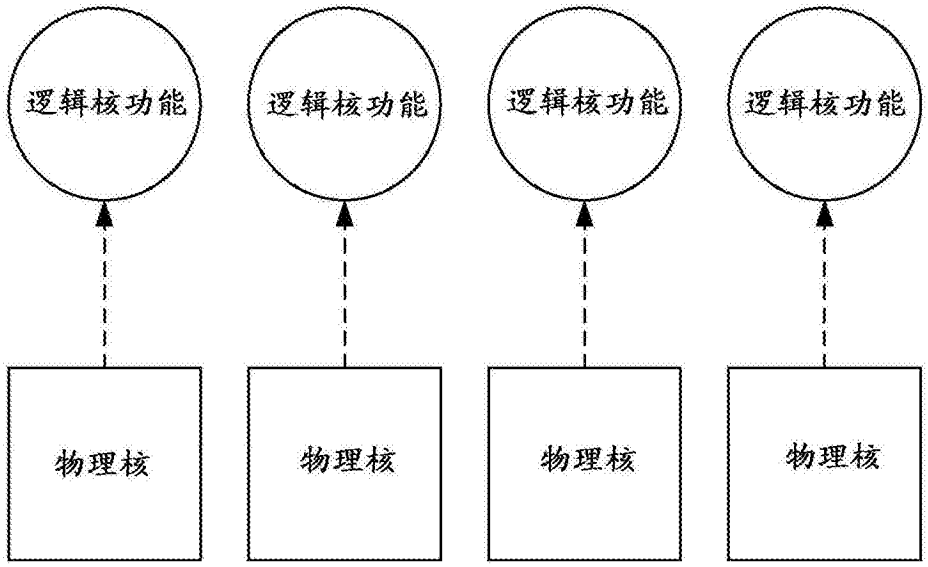

逻辑核功能
逻辑核功能
逻辑核功能
逻辑核功能
物理核
物理核
物理核
物理核

图11

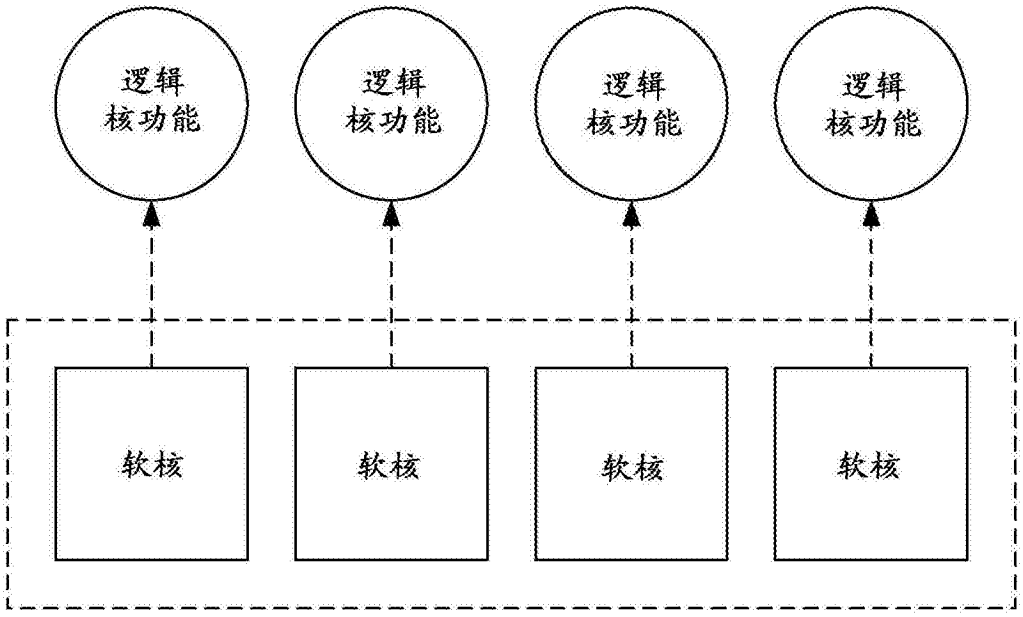

逻辑
核功能
逻辑
核功能
逻辑
核功能
逻辑
核功能
软核
软核
软核
软核

图12

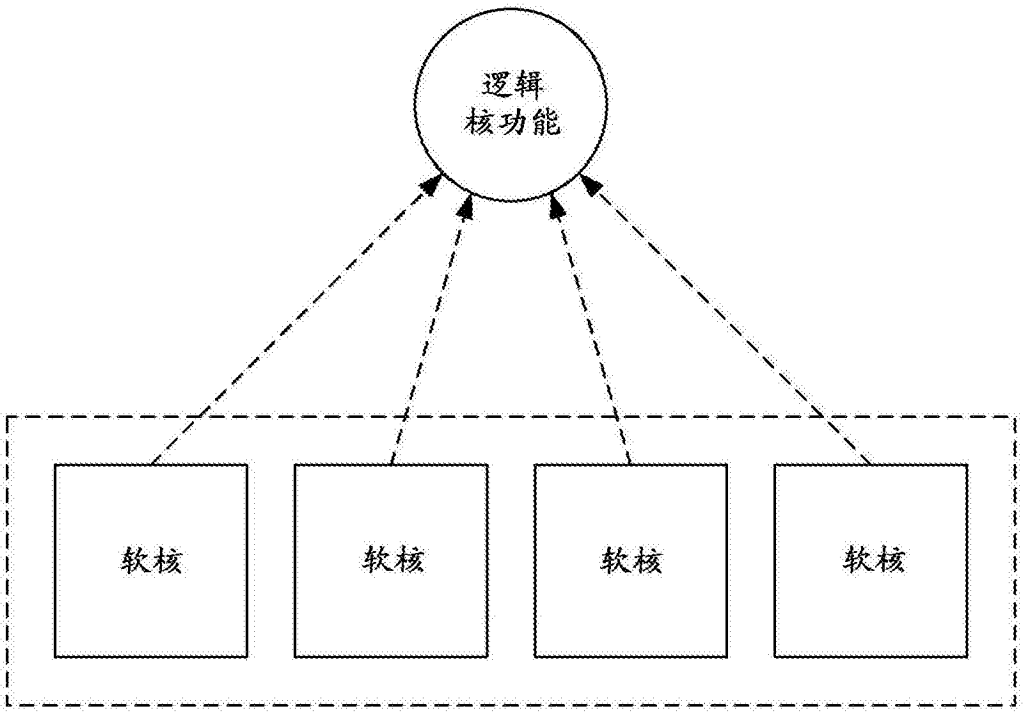


图13

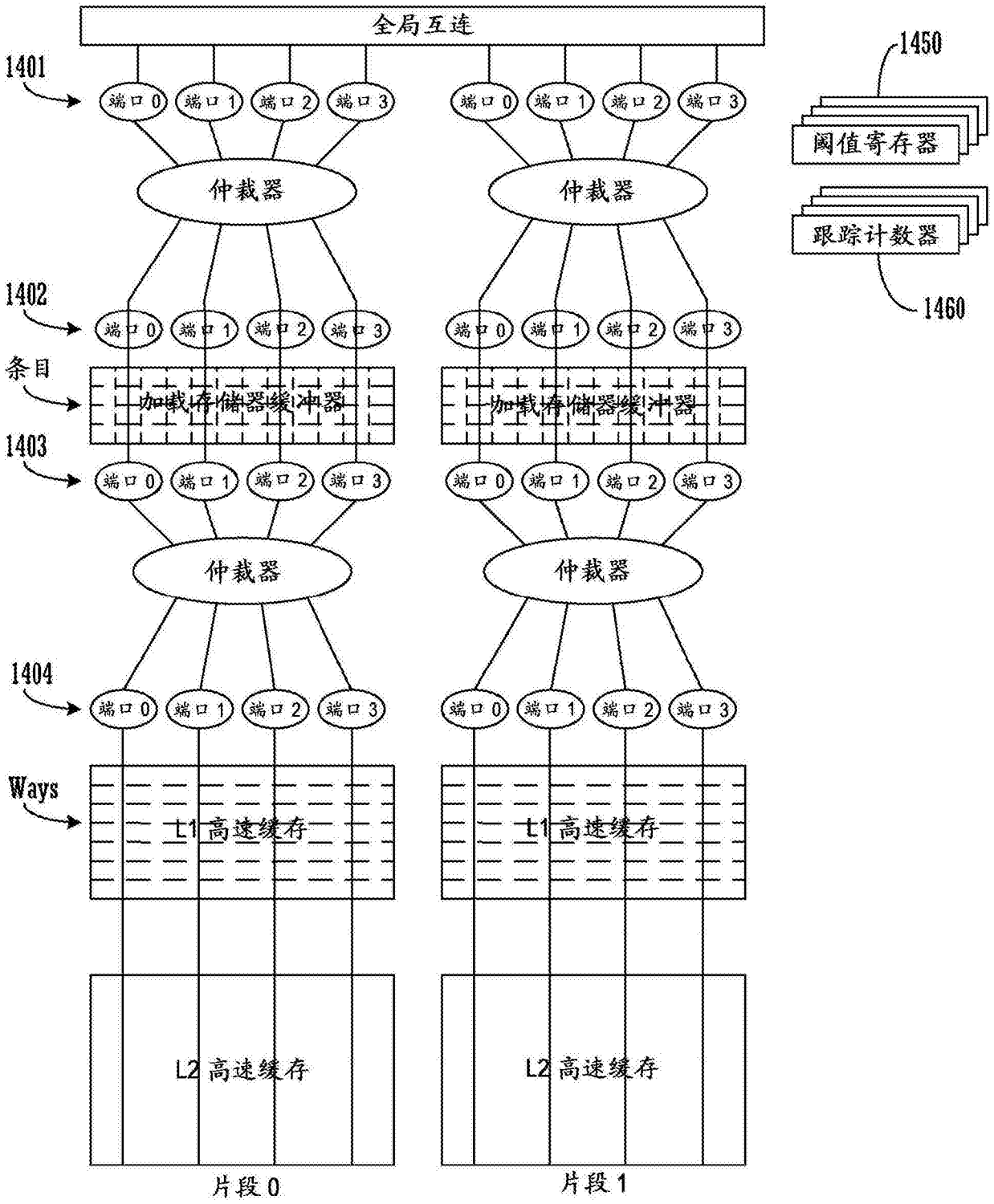

全局互连
1401
端口 0
端口 1
端口 2
端口 3
仲裁器
1402
条目
加载存储器缓冲器
1403
1404
Ways
L1 高速缓存
L2 高速缓存
片段 0
片段 1
1450
阈值寄存器
跟踪计数器
1460

图14

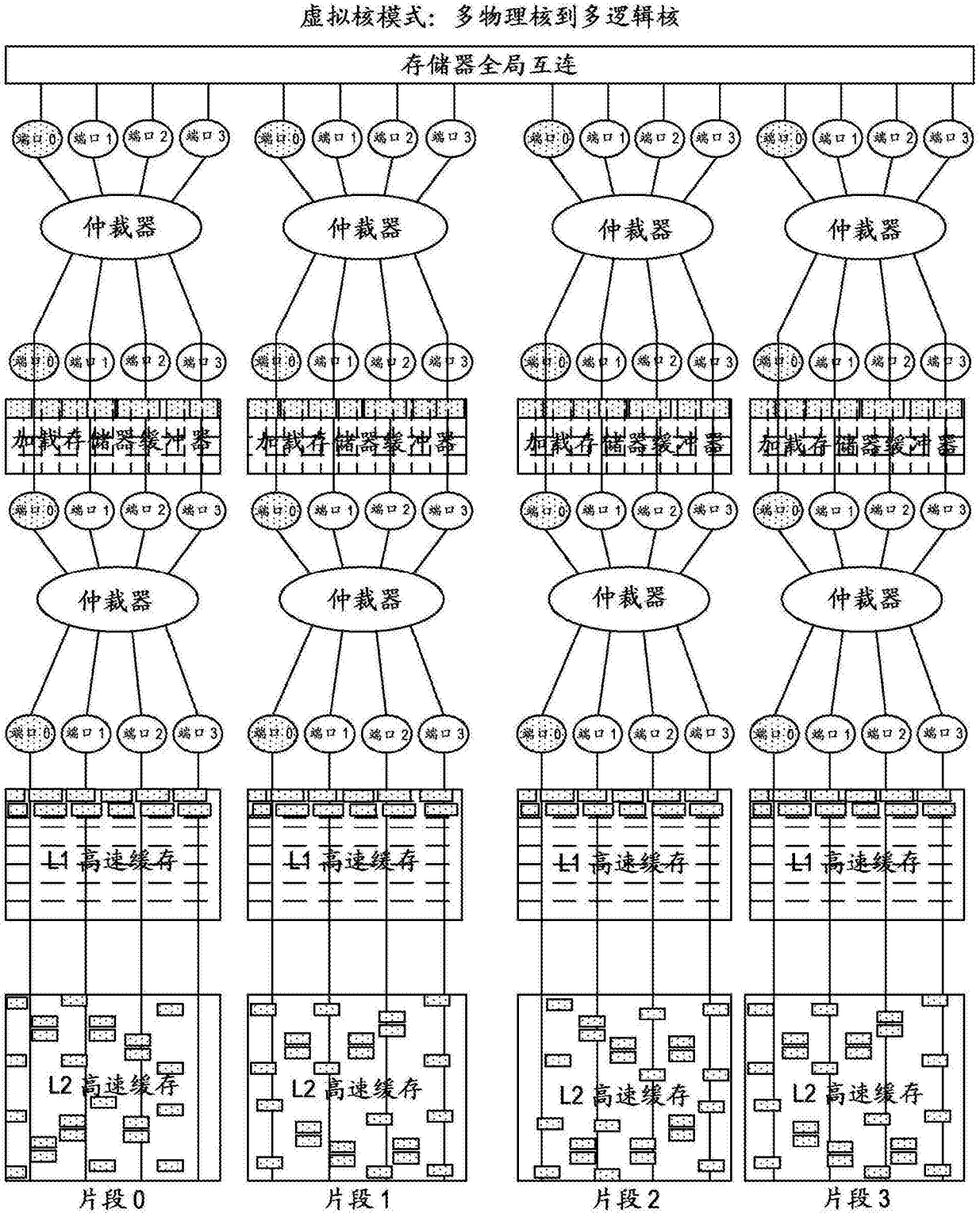

虚拟核模式：多物理核到多逻辑核
存储器全局互连
端口 0
端口 1
端口 2
端口 3
仲裁器
加载存储器缓冲器
L1 高速缓存
L2 高速缓存
片段 0
片段 1
片段 2
片段 3

图15

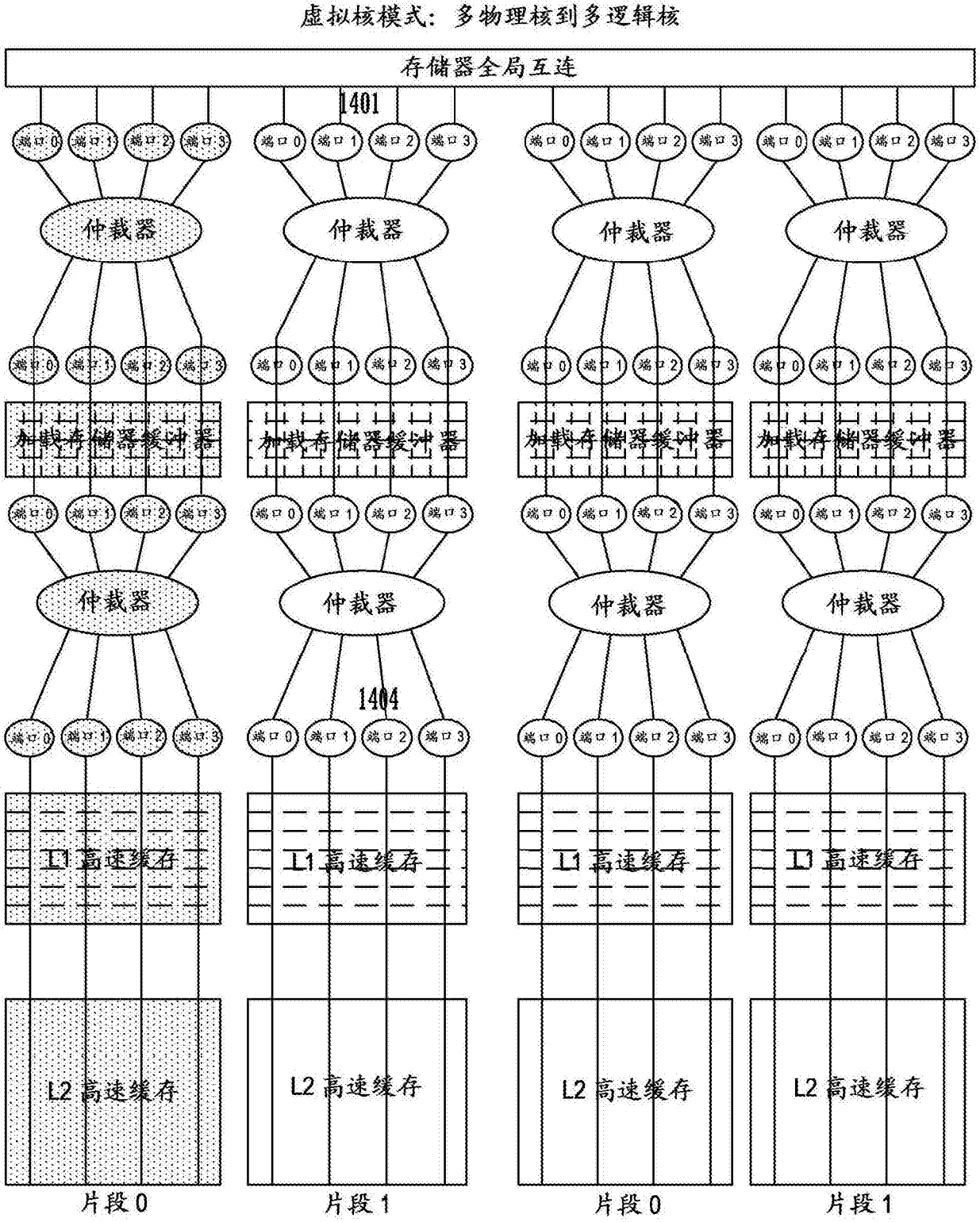

虚拟核模式：多物理核到多逻辑核
存储器全局互连
1401
端口 0
端口 1
端口 2
端口 3
仲裁器
加载存储器缓冲器
1404
L1 高速缓存
L2 高速缓存
片段 0
片段 1
片段 0
片段 1

图16

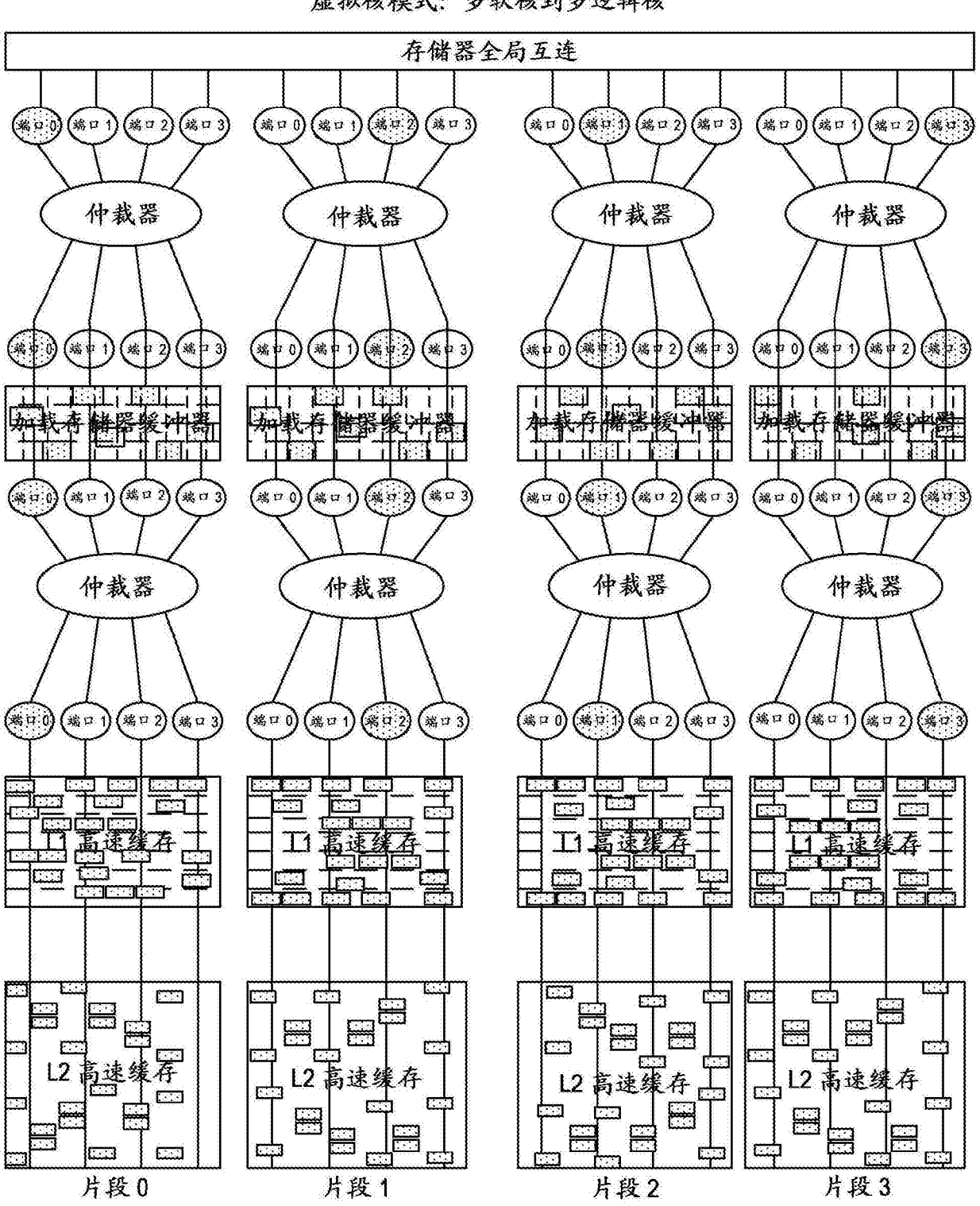

虚拟核模式：多软核到多逻辑核
存储器全局互连
端口0
端口1
端口2
端口3
仲裁器
加载存储器缓冲器
L1 高速缓存
L2 高速缓存
片段0
片段1
片段2
片段3

图17

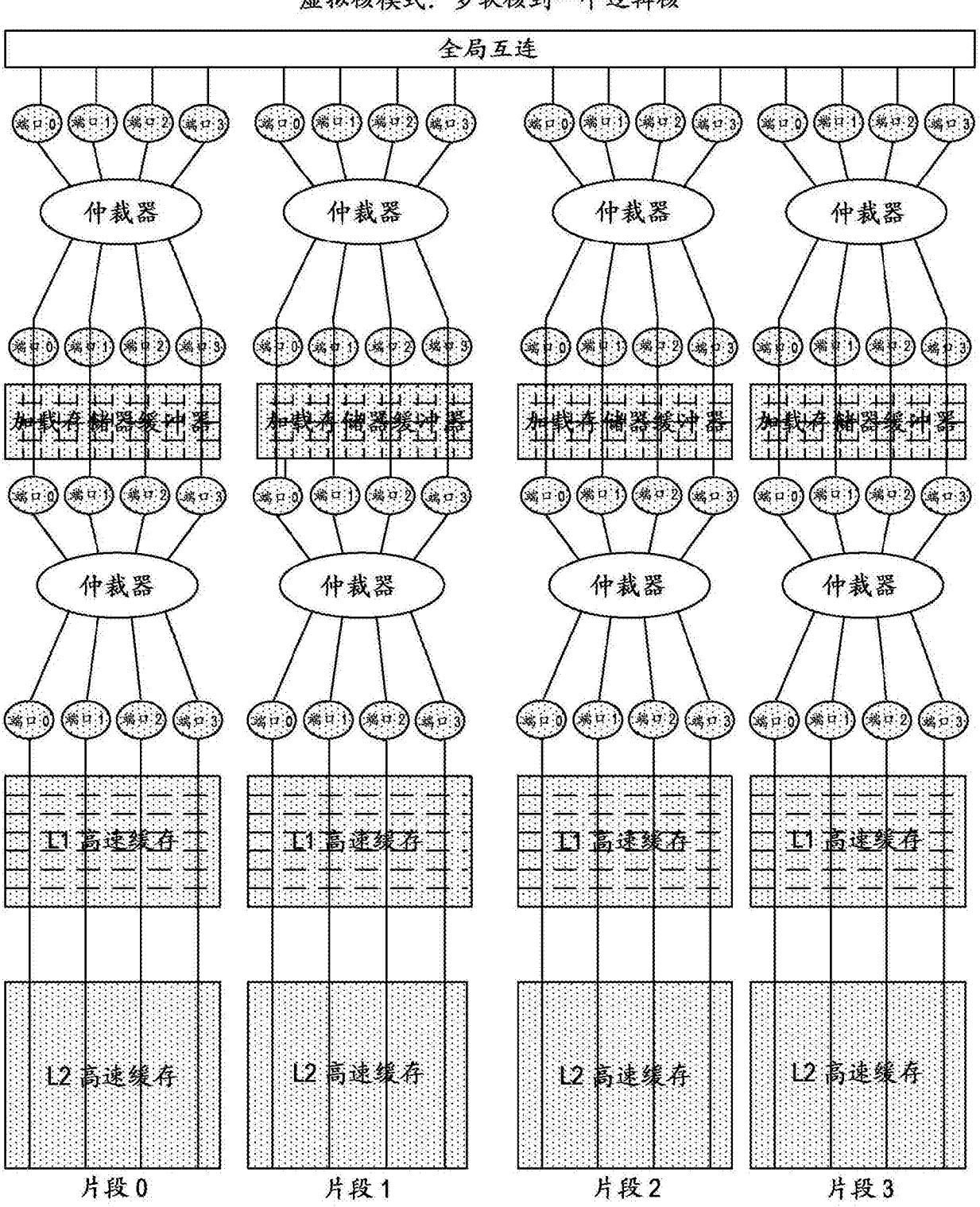

虚拟核模式：多软核到一个逻辑核
全局互连
端口0
端口1
端口2
端口3
仲裁器
加载存储器缓冲器
L1 高速缓存
L2 高速缓存
片段 0
片段 1
片段 2
片段 3

图18

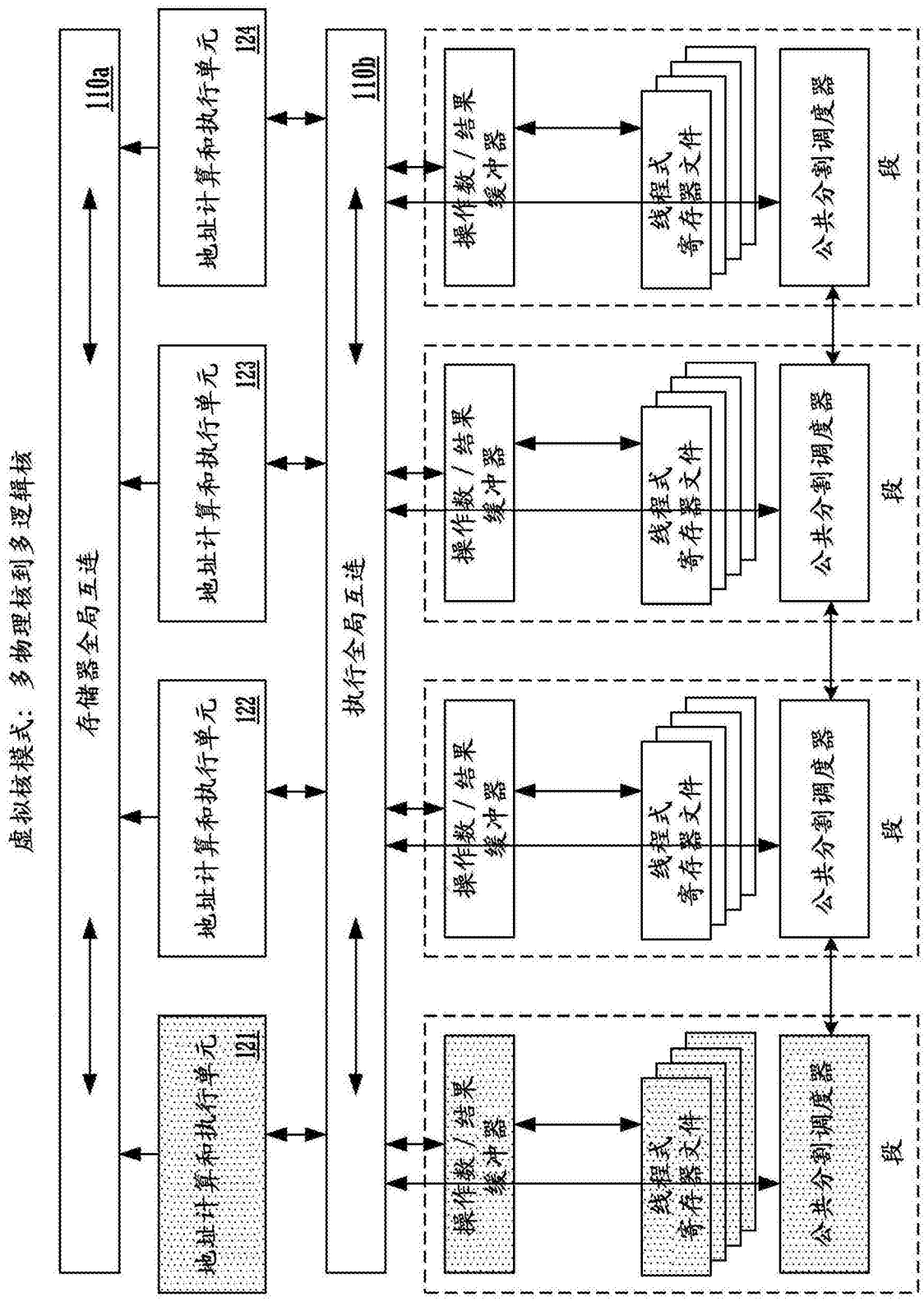

虚拟核模式：多物理核到多逻辑核
110a
存储器全局互连
地址计算和执行单元 121
地址计算和执行单元 122
地址计算和执行单元 123
地址计算和执行单元 124
110b
执行全局互连
操作数/结果缓冲器
线程式寄存器文件
公共分割调度器
段

图19

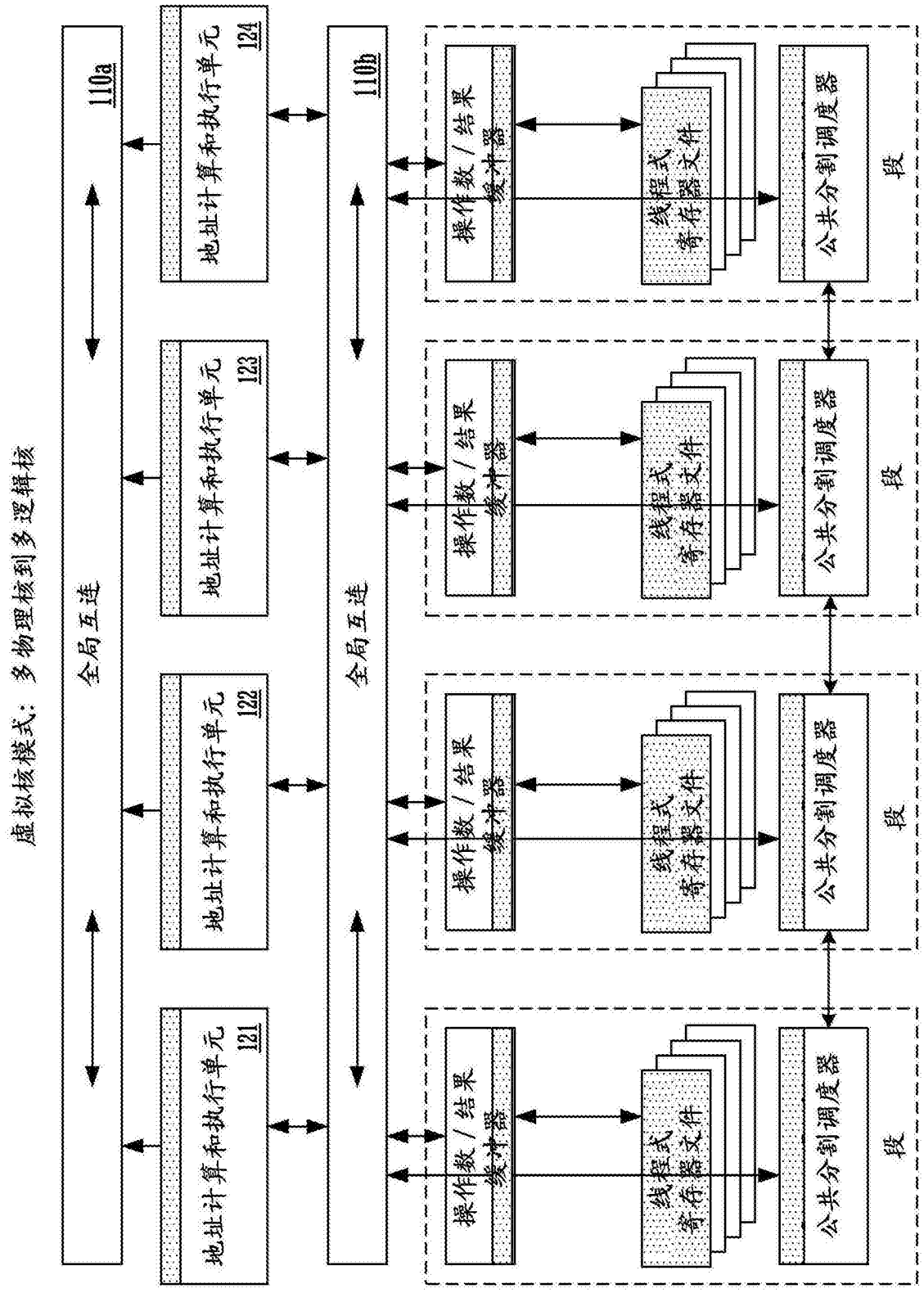

虚拟核模式：多物理核到多逻辑核
110a
全局互连
地址计算和执行单元 121
地址计算和执行单元 122
地址计算和执行单元 123
地址计算和执行单元 124
110b
全局互连
操作数 / 结果缓冲器
线程式寄存器文件
公共分割调度器
段

图20

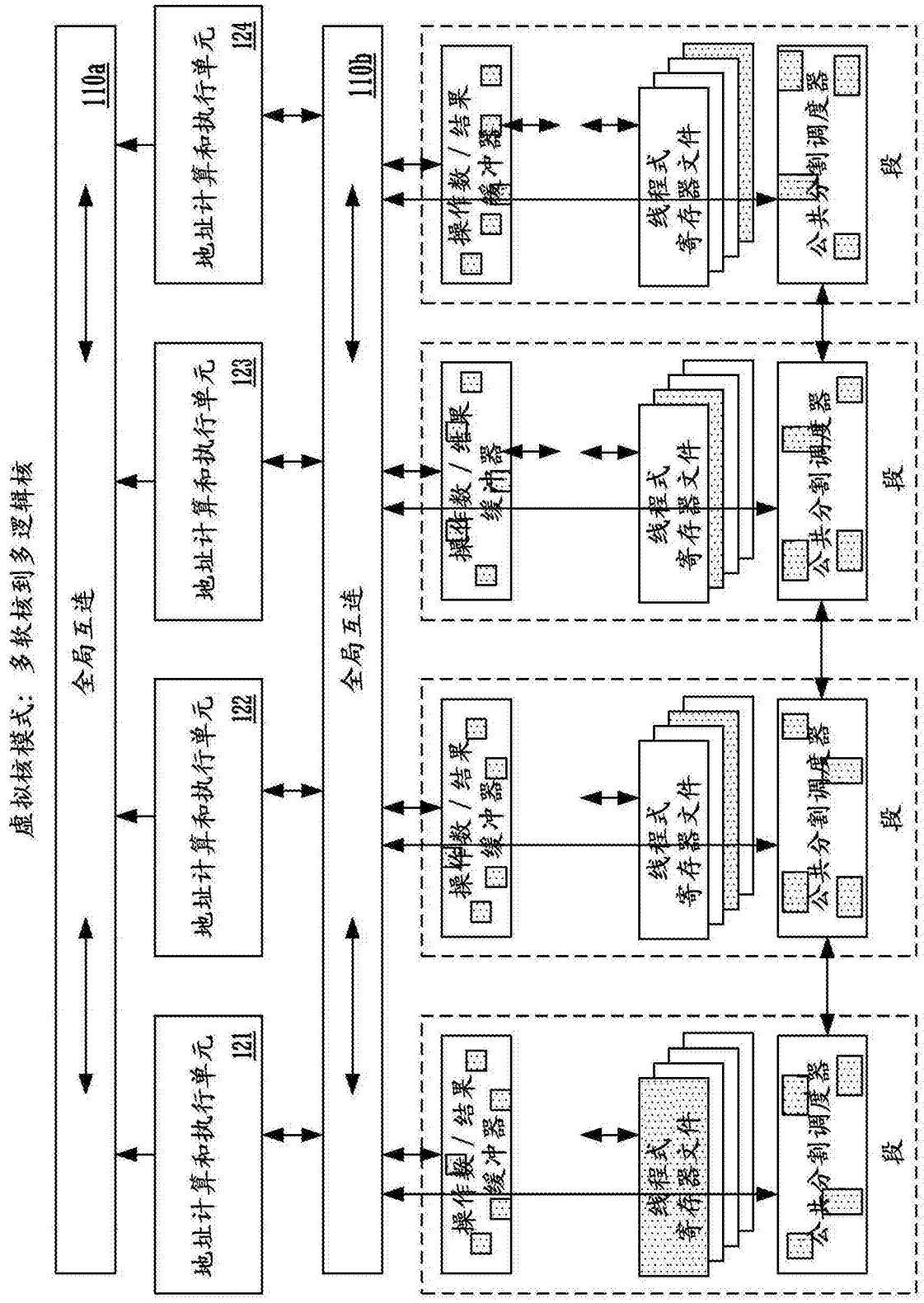

虚拟核模式：多软核到多逻辑核
全局互连
全局互连
110a
110b
地址计算和执行单元 121
地址计算和执行单元 122
地址计算和执行单元 123
地址计算和执行单元 124
操作数 / 结果缓冲器
线程式寄存器文件
公共分割调度器
段

图21

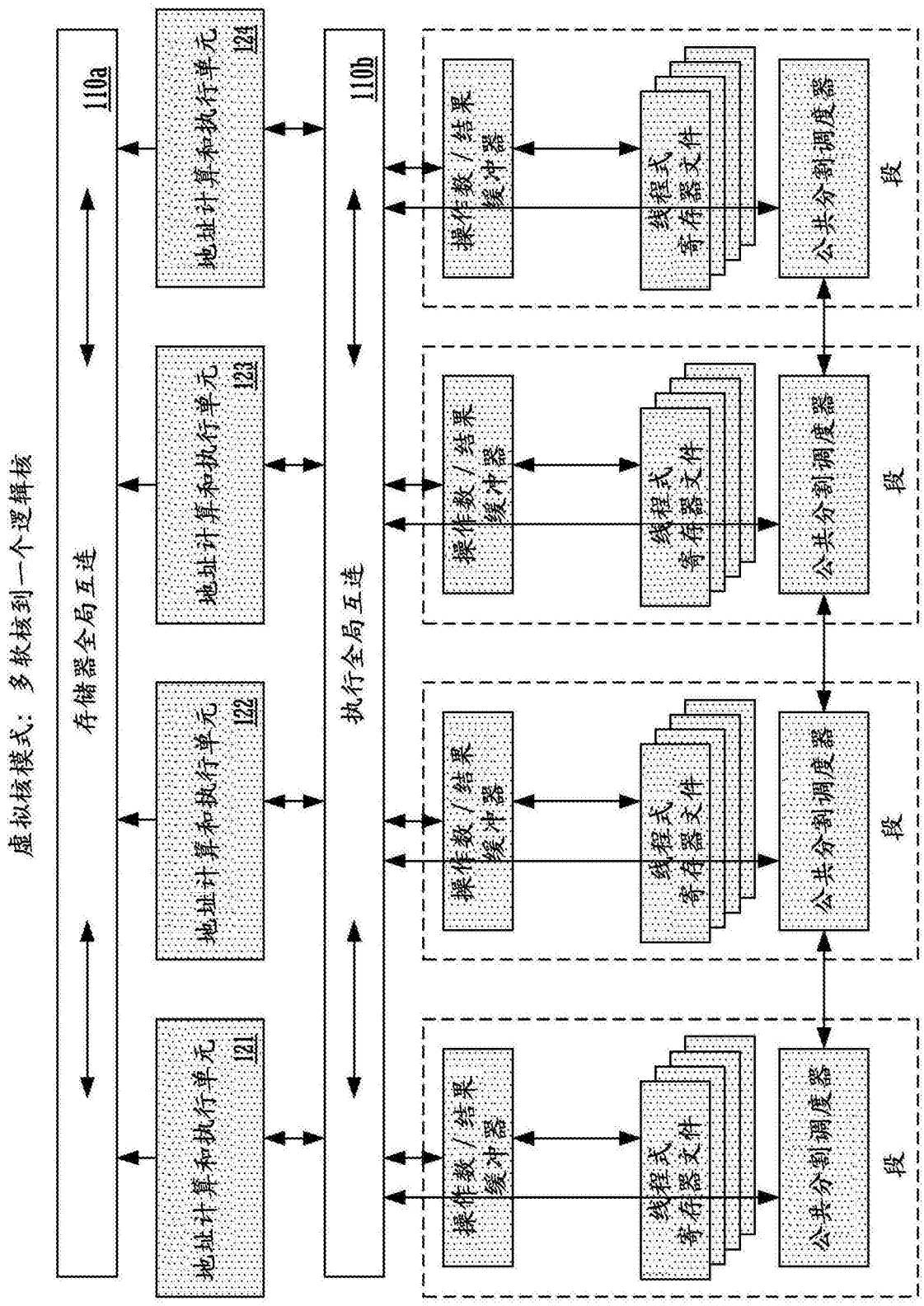

虚拟核模式：多软核到一个逻辑核
存储器全局互连
执行全局互连
110a
110b
地址计算和执行单元 121
地址计算和执行单元 122
地址计算和执行单元 123
地址计算和执行单元 124
操作数/结果缓冲器
线程式寄存器文件
公共分割调度器
段

图22

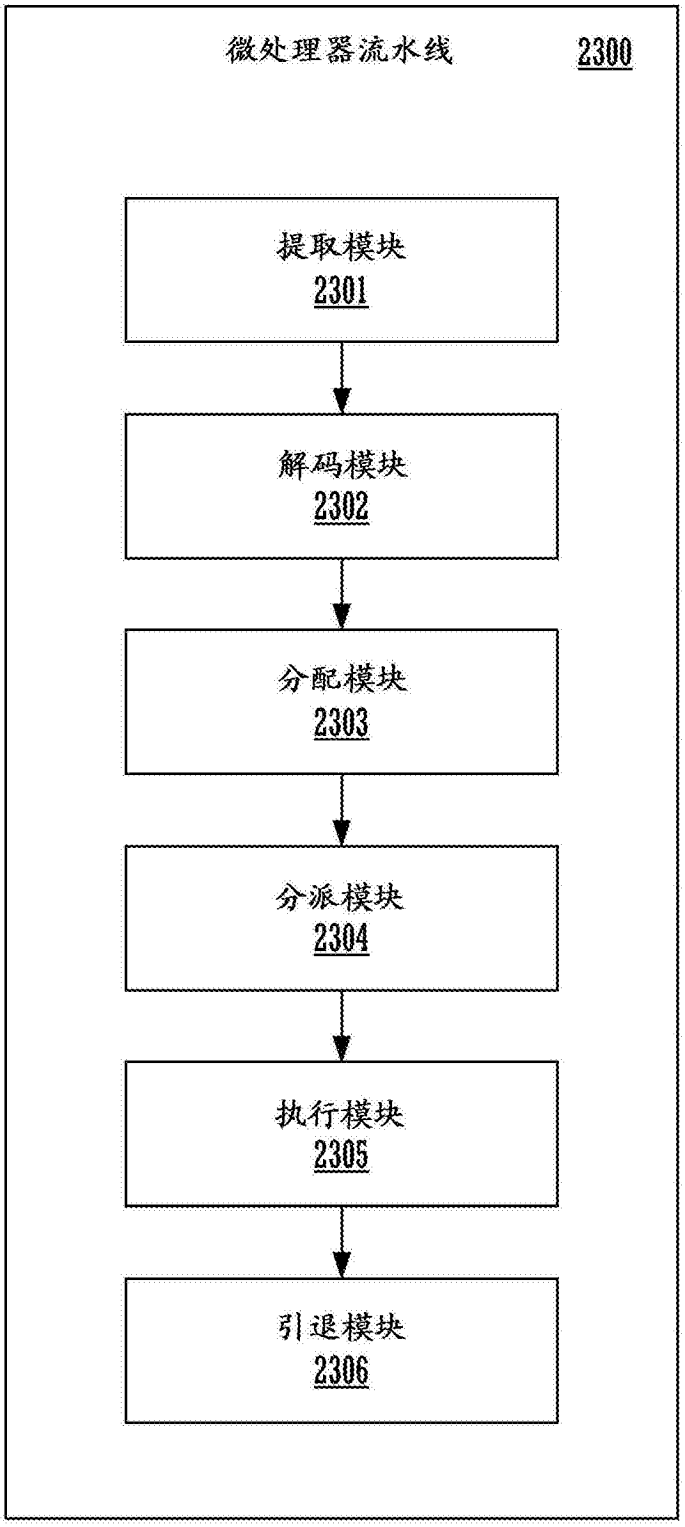

微处理器流水线
2300
提取模块
2301
解码模块
2302
分配模块
2303
分派模块
2304
执行模块
2305
引退模块
2306

图23