(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 107408021 A
(43)申请公布日 2017.11.28

(21)申请号 201680018780.X

(22)申请日 2016.02.26

(30)优先权数据
14/670526 2015.03.27 US

(85)PCT国际申请进入国家阶段日
2017.09.27

(86)PCT国际申请的申请数据
PCT/US2016/019679 2016.02.26

(87)PCT国际申请的公布数据
WO2016/160199 EN 2016.10.06

(71)申请人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 R.G.布兰肯希普

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 徐红燕 刘春元

(51)Int.Cl.
G06F 3/06(2006.01)
G06F 12/0802(2016.01)

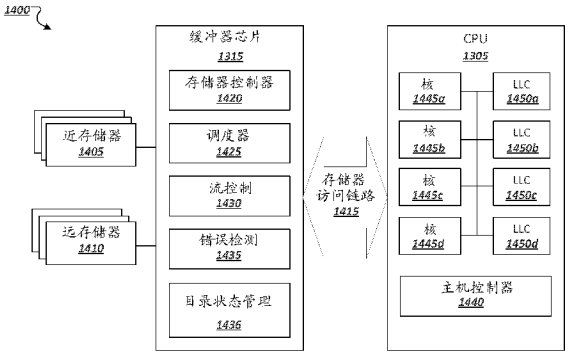
权利要求书3页 说明书30页 附图22页

(54)发明名称

隐式目录状态更新

(57)摘要

通过链路接收请求存储器中的特定行的请求。在存储器中识别目录状态记录，该目录状态记录识别该特定行的目录状态。根据该请求来识别请求类型。基于该特定行的目录状态和请求类型来确定该特定行的目录状态要从特定状态改为新状态。响应于该请求的接收，改变目录状态记录以反映新状态。响应于该请求而发送该特定行的副本。



CN 107408021 A

1.一种装置,包括:

存储器控制器,用于:

通过链路接收读取请求,其中,所述读取请求请求存储器中的特定行;

识别存储器中的目录状态记录,其中,所述目录状态记录识别所述特定行的目录状态;

根据所述读取请求来识别所述读取请求的类型;

确定所述特定行的目录状态要基于所述读取请求从所述特定状态改为新状态,其中,所述改变是根据所述特定行的目录状态和所述读取请求的类型确定的;以及

响应于所述读取请求的接收,改变所述目录状态记录以反映所述新状态;以及

发射器,用于响应于所述读取请求而发送所述特定行的副本。

2.根据权利要求1所述的装置,其中,所述读取请求是从与所述存储器控制器分离的主机设备接收的。

3.根据权利要求2所述的装置,其中,所述读取请求是通过缓冲存储器访问链路从所述主机设备接收的。

4.根据权利要求3所述的装置,其中,所述存储器控制器被提供在与所述主机设备分离的缓冲器芯片上。

5.根据权利要求1所述的装置,其中,所述读取请求源自特定缓存代理,并且所述改变是根据所述特定缓存代理的类型确定的。

6.根据权利要求5所述的装置,其中,所述特定缓存代理的类型是本地缓存代理或远程缓存代理中的一个。

7.根据权利要求5所述的装置,其中,所述特定缓存代理的类型是至少部分地基于所述读取请求中编码的一个或多个值来确定的。

8.根据权利要求1所述的装置,其中,所述目录状态记录包括两个或更多个位,并且所述位被包括在所述行中并且反映所述特定状态。

9.根据权利要求8所述的装置,其中,所述存储器控制器还用于将所述新状态直接写入到存储器中的所述特定行。

10.根据权利要求9所述的装置,其中,归属代理将所述目录状态更新为所述新状态,并且所述存储器控制器独立于所述归属代理地将所述目录状态更新为所述新状态。

11.根据权利要求8所述的装置,其中,所述位被包括在所述特定行的纠错码位中。

12.根据权利要求1所述的装置,其中,所述存储器控制器还用于确定与另一读取请求相关联的目录状态改变是不可预测的。

13.根据权利要求12所述的装置,其中,所述存储器控制器还用于:

从归属代理接收指示对与所述另一读取请求相对应的行的目录状态改变的通信;以及

基于所述通信,执行到对与所述另一读取请求相对应的行的写入,以更新与所述另一读取请求相对应的行的目录状态。

14.根据权利要求13所述的装置,其中,所述通信包括与所述另一读取请求相对应的行的写回,其中,被写回的行指示与所述另一读取请求相对应的行的所述目录状态。

15.根据权利要求14所述的装置,其中,与所述另一读取请求相对应的行的所述目录状态包括与所述另一读取请求相对应的行的目录状态记录。

16.根据权利要求1所述的装置,其中,所述目录状态是一组目录状态中的一个,并且该

组目录状态包括共享状态、任何状态和无效状态。

17.在其上存储有代码的至少一种机器可访问存储介质，所述代码当在机器上被执行时使得所述机器：

通过链路接收读取请求，其中，所述读取请求请求存储器中的特定行；

识别存储器中的目录状态记录，其中，所述目录状态记录识别所述特定行的目录状态；

根据所述读取请求来识别所述读取请求的类型；

确定所述特定行的目录状态要基于所述读取请求从所述特定状态改为新状态，其中，所述改变是根据所述特定行的目录状态和所述读取请求的类型确定的；以及

响应于所述读取请求的接收，改变所述目录状态记录以反映所述新状态；以及

响应于所述读取请求而发送所述特定行的副本。

18.一种装置，包括：

缓存代理逻辑，用于：

确定与存储器的特定行相对应的缓冲存储器读取请求，其中所述读取请求是多个读取请求类型中的特定读取请求类型；

通过缓冲存储器链路向缓冲器芯片发送所述读取请求；

响应于所述读取请求，接收存储器的所述特定行的副本；

根据所述特定行中包括的目录状态记录来确定所述特定行处于特定目录状态中；

确定对所述目录状态的改变；以及

基于所述特定目录状态和所述读取请求的特定读取请求类型，确定是否向所述缓冲器芯片传送所述改变。

19.根据权利要求18所述的装置，其中，如果所述改变可根据所述特定目录状态和所述读取请求的特定读取请求类型预测到，则所述改变不被传送至所述缓冲器芯片。

20.根据权利要求19所述的装置，其中，如果所述改变不可由所述缓冲器芯片根据所述特定目录状态和所述读取请求的特定读取请求类型预测到，则所述改变被传送至所述缓冲器芯片。

21.一种系统，包括：

缓冲器芯片；

存储器，该存储器通过所述缓冲器芯片而被访问；

处理器块，该处理器块通过存储器访问链路耦合至所述缓冲器芯片；

其中，所述缓冲器芯片包括存储器控制器以用于：

通过所述存储器访问链路从所述处理器块接收读取请求，其中，所述读取请求请求所述存储器中的特定行；

识别所述存储器中的目录状态记录，其中，所述目录状态记录识别所述特定行的目录状态；

根据所述读取请求来识别所述读取请求的类型；

确定所述特定行的目录状态要基于所述读取请求从所述特定状态改为新状态，其中，所述改变是根据所述特定行的目录状态和所述读取请求的类型确定的；以及

响应于所述读取请求的接收，改变所述目录状态记录以反映所述新状态。

22.根据权利要求21所述的系统，其中，所述处理器块用于通过通用输入/输出(GPIO)

互连链路与一个或多个其它设备对接，所述存储器访问链路不同于所述GPIO互连链路，并且所述存储器访问链路的物理层是基于所述GPIO互连链路的物理层。

23.一种方法，包括：

通过链路接收读取请求，其中，所述读取请求请求存储器中的特定行；

识别存储器中的目录状态记录，其中，所述目录状态记录识别所述特定行的目录状态；

根据所述读取请求来识别所述读取请求的类型；

确定所述特定行的目录状态要基于所述读取请求从所述特定状态改为新状态，其中，所述改变是根据所述特定行的目录状态和所述读取请求的类型确定的；以及

响应于所述读取请求的接收，改变所述目录状态记录以反映所述新状态；以及

响应于所述读取请求而发送所述特定行的副本。

24.一种系统，包括：

用于通过链路接收读取请求的部件，其中，所述读取请求请求存储器中的特定行；

用于识别存储器中的目录状态记录的部件，其中，所述目录状态记录识别所述特定行的目录状态；

用于根据所述读取请求来识别所述读取请求的类型的部件；

用于确定所述特定行的目录状态要基于所述读取请求从所述特定状态改为新状态的部件，其中，所述改变是根据所述特定行的目录状态和所述读取请求的类型确定的；以及

用于响应于所述读取请求的接收而改变所述目录状态记录以反映所述新状态的部件；以及

用于响应于所述读取请求而发送所述特定行的副本的部件。

25.一种方法，包括：

确定与存储器的特定行相对应的缓冲存储器读取请求，其中所述读取请求是多个读取请求类型中的特定读取请求类型；

通过缓冲存储器链路向缓冲器芯片发送所述读取请求；

响应于所述读取请求，接收存储器的所述特定行的副本；

根据所述特定行中包括的目录状态记录来确定所述特定行处于特定目录状态中；

确定对所述目录状态的改变；以及

基于所述特定目录状态和所述读取请求的特定读取请求类型，确定是否向所述缓冲器芯片传送所述改变。

26.一种系统，包括：

用于确定与存储器的特定行相对应的缓冲存储器读取请求的部件，其中所述读取请求是多个读取请求类型中的特定读取请求类型；

用于通过缓冲存储器链路向缓冲器芯片发送所述读取请求的部件；

用于响应于所述读取请求来接收存储器的所述特定行的副本的部件；

用于根据所述特定行中包括的目录状态记录来确定所述特定行处于特定目录状态中的部件；

用于确定对所述目录状态的改变的部件；以及

用于基于所述特定目录状态和所述读取请求的特定读取请求类型来确定是否向所述缓冲器芯片传送所述改变的部件。

隐式目录状态更新

[0001] 相关申请的交叉引用

本申请要求于2015年3月27日提交的题为“隐式目录状态更新(IMPLIED DIRECTORY STATE UPDATES)”的美国非临时专利申请号14/670,526的权益和优先权,其全部内容通过引用并入本文。

技术领域

[0002] 本公开一般涉及计算架构的领域,并且更具体地涉及缓冲存储器协议。

背景技术

[0003] 半导体处理和逻辑设计的进步已允许可存在于集成电路设备上的逻辑的数量上的增加。作为必然结果,计算机系统配置已从系统中的单个或多个集成电路演进为存在于各个集成电路上的多个核、多个硬件线程和多个逻辑处理器以及集成在此类处理器内的其它接口。处理器或集成电路通常包括单个物理处理器管芯,其中处理器管芯可包括任何数量的核、硬件线程、逻辑处理器、接口、存储器、控制器中枢等。

[0004] 由于具有更强的能力以在更小的封装中装配更多的处理功率,更小的计算设备的普及度增加。智能电话、平板、超薄笔记本以及其它用户设备已经呈指数级增长。然而,这些更小的设备依赖于用于超出形状因子的复杂处理和数据存储二者的服务器。因此,高性能计算市场(即,服务器空间)方面的需求也已经增加。例如,在现代服务器中,通常不仅存在具有多个核的单个处理器,还存在多个物理处理器(也称为多个插槽)以增加计算能力。但随着处理能力与计算系统中的设备数量一同增长,插槽和其它设备之间的通信变得更加关键。

[0005] 实际上,互连已从原来处理电通信的更传统的多点分支总线成长为促进快速通信的充分发展的互连架构。不幸的是,随着未来处理器以甚至更高速率消耗的需求,对于现存互连架构的能力寄托了对应的需求。

附图说明

[0006] 图1图示出根据一个实施例的系统的简化框图,该系统包括串行点对点互连以连接计算机系统中的I/O设备。

[0007] 图2图示出根据一个实施例的分层协议栈的简化框图。

[0008] 图3图示出事务描述符的实施例。

[0009] 图4图示出串行点对点链路的实施例。

[0010] 图5图示出潜在高性能互连(HPI)系统配置的实施例。

[0011] 图6图示出与HPI相关联的分层协议栈的实施例。

[0012] 图7图示出示例状态机的表示。

[0013] 图8图示出通过示例20通道数据链路发送的示例微片(flit)的表示。

[0014] 图9图示出通过示例8通道数据链路发送的示例微片的表示。

[0015] 图10图示出示例多槽(multi-slot)微片的表示。

[0016] 图11图示出通过示例8通道数据链路发送的示例微片的表示。

[0017] 图12图示出包括调试消息的示例多槽微片的表示。

[0018] 图13图示出采用缓冲存储器协议的系统的表示。

[0019] 图14图示出支持缓冲存储器协议的缓冲器设备的表示。

[0020] 图15A-15C图示出缓冲存储器协议中的示例微片的表示。

[0021] 图16A-16C图示出根据一些实现方式的缓冲器设备和主机设备之间的示例通信。

[0022] 图17图示出根据一些实现方式的缓存行的示例实施例的表示。

[0023] 图18图示出包括多个处理器插槽的计算系统的框图的实施例。

[0024] 图19图示出包括多核处理器的计算系统的框图的另一实施例。

[0025] 各图中类似的参考标号和标示指示类似的元件。

具体实施方式

[0026] 在以下描述中,阐述了众多具体细节,诸如以下示例:具体的处理器类型和系统配置、具体硬件结构、具体架构和微架构细节、具体寄存器配置、具体指令类型、具体系统组件、具体处理器管线级、具体互连层、具体分组/事务配置、具体事务名、具体协议交换、具体链路宽度、具体实现方式、以及操作等,以便提供对本发明的透彻理解。然而,可对本领域技术人员显而易见的是,不一定需要采用这些具体细节来实践本公开的主题。在其它实例中,已避免对已知的组件或方法(诸如具体的和替代的处理器架构、用于所描述的算法的具体逻辑电路/代码、具体固件代码、低级互连操作、具体逻辑配置、具体制造技术和材料、具体编译器实现方式、代码中算法的具体表达、具体掉电和选通技术/逻辑、以及计算机系统的其它具体操作细节)的非常详细的描述,以便避免不必要地模糊本公开。

[0027] 尽管可参考具体集成电路中(诸如在计算平台或微处理器中)的节能、能效、处理效率等来描述以下实施例,但其它实施例可应用于其它类型的集成电路和逻辑设备。本文描述的实施例的相似的技术和教导可被应用于同样可受益于此类特征的其它类型的电路或半导体设备。例如,公开的实施例不限于服务器计算机系统、桌面计算机系统、膝上型计算机、Ultrabook™(超级本),而是也可用于诸如手持设备、智能电话、平板、其它薄笔记本、片上系统(SOC)设备、和嵌入式应用之类的其它设备中。手持设备的一些示例包括蜂窝电话、互联网协议设备、数字相机、个人数字助理(PDA)和手持PC。这里,用于高性能互连的相似的技术可被应用于在低功率互连中增加性能(或甚至省电)。嵌入式应用通常包括微控制器、数字信号处理器(DSP)、片上系统、网络计算机(NetPC)、机顶盒、网络中枢、广域网(WAN)交换机或可以执行以下所教导的功能和操作的任何其它系统。此外,本文描述的装置、方法和系统不限于物理计算设备,而是也可涉及用于节能和能效的软件优化。如可在以下描述中变得容易地显而易见的,本文(无论是否参考硬件、固件、软件或其组合)描述的方法、装置和系统的实施例可被视为对与性能考虑相权衡的“绿色技术”未来是至关重要的。

[0028] 随着计算系统一直进步,其中的组件正变得更加复杂。耦合组件并在组件之间通信的互连架构也在复杂度上有所增加,以保证带宽需求被满足用于最佳组件操作。此外,不同的市场区段需要不同方面的互连架构以适应相应市场。例如,服务器需要更高性能,而移动生态系统有时能够牺牲总体性能以省电。但是,大多数构造的唯一目的是在最大省电的

情况下提供最高可能的性能。此外，各种不同的互连可以潜在地受益于本文描述的主题。

[0029] 快速外围组件互连(PCI)(PCIe)互连构造架构和快速路径互连(QPI)构造架构以及其它示例可以根据本文描述的一个或多个原理以及其它示例而潜在地被改进。例如，PCIe的首要目标是使来自不同供应商的组件和设备能够在开放的架构中相互操作，跨越多个市场区段；客户端(桌面和移动)、服务器(标准和企业)、以及嵌入式和通信设备。快速PCI是高性能、通用I/O互连，其被定义用于各种各样的未来计算和通信平台。一些PCI属性，诸如其使用模型、负载存储架构和软件接口，已通过其修订来维护，而先前的并行总线实现方式已被高度可缩放的、全串行接口所替代。快速PCI的较近版本利用点对点互连、基于交换机的技术、以及分组化协议中的进步，来传递新的性能水平和特征。功率管理、服务质量(QoS)、热插拔/热切换支持、数据完整性、以及错误处理在由快速PCI所支持的高级特征中的一些当中。尽管本文的主要讨论参考了新的高性能互连(HPI)架构，但本文描述的发明的各方面可被应用于其它互连架构，诸如顺从PCIe的架构、顺从QPI的架构、顺从MIPI的架构、高性能架构、或其它已知互连架构。

[0030] 参见图1，图示出由互连了一组组件的点对点链路组成的构造的实施例。系统100包括耦合于控制器中枢115的处理器105和系统存储器110。处理器105可以包括任何处理元件，诸如微处理器、主处理器、嵌入处理器、协处理器或其它处理器。处理器105通过前侧总线(FSB)106耦合于控制器中枢115。在一个实施例中，FSB 106为如下所述的串行点对点互连。在另一实施例中，链路106包括串行的、差分互连架构，其顺从于不同的互连标准。

[0031] 系统存储器110包括任何存储器设备，诸如随机存取存储器(RAM)、非易失性(NV)存储器、或可由系统100中的设备访问的其它存储器。系统存储器110通过存储器接口116耦合于控制器中枢115。存储器接口的示例包括双数据速率(DDR)存储器接口、双通道DDR存储器接口、和动态RAM(DRAM)存储器接口。

[0032] 在一个实施例中，控制器中枢115可以包括根中枢、根联合体、或根控制器，诸如在PCIe互连层级中。控制器中枢115的示例包括芯片组、存储器控制器中枢(MCH)、北桥、互连控制器中枢(ICH)、南桥和根控制器/中枢。术语芯片组常常指代两个物理分离的控制器中枢，例如，耦合于互连控制器中枢(ICH)的存储器控制器中枢(MCH)。注意，当前系统常常包括与处理器105相集成的MCH，而控制器115将与I/O设备以如下描述的类似方式进行通信。在一些实施例中，通过根联合体115可选地支持对等路由。

[0033] 这里，控制器中枢115通过串行链路119耦合于开关/桥120。输入/输出模块117和121，其还可称为接口/端口117和121，可以包括/实现分层协议栈，以提供控制器中枢115和开关120之间的通信。在一个实施例中，多个设备能够耦合于开关120。

[0034] 开关/桥120将分组/消息从设备125上游(即层级的上方)朝根联合体路由至控制器中枢115和下游(即远离根控制器的层级下方)，从处理器105或系统存储器110路由至设备125。在一个实施例中，开关120称为多个虚拟PCI至PCI桥设备的逻辑组装件。设备125包括要被耦合于电子系统的任何内部或外部设备或组件，诸如I/O设备、网络接口控制器(NIC)、扩充卡、音频处理器、网络处理器、硬驱动器、存储设备、CD/DVD ROM、监视器、打印机、鼠标、键盘、路由器、便携式存储设备、火线设备、通用串行总线(USB)设备、扫描仪和其它输入/输出设备。通常在PCIe专门语中，诸如设备，被称为端点。尽管没有具体地示出，但设备125可包括桥(例如，PCIe至PCI/PCI-X的桥)，以支持由此类设备支持的设备或互连构

造的旧有版本或其它版本。

[0035] 图形加速器130也可以通过串行链路132耦合于控制器中枢115。在一个实施例中，图形加速器130耦合于MCH，该MCH耦合于ICH。开关120，以及相应地I/O设备125于是耦合于ICH。I/O模块131和118也将实现分层协议栈，以在图形加速器130和控制器中枢115之间通信。类似于上述的MCH讨论，图形控制器或图形加速器130本身可被集成于处理器105中。

[0036] 转到图2，图示出分层协议栈的实施例。分层协议栈200可包括任何形式的分层通信栈，诸如QPI栈、PCIe栈、下一代高性能计算互连(HPI)栈、或其它分层栈。在一个实施例中，协议栈200可包括事务层205、链路层210和物理层220。诸如图1中的接口117、118、121、122、126和131之类的接口可被表示为通信协议栈200。作为通信协议栈的表示还可称为实现/包括协议栈的模块或接口。

[0037] 分组可用于在组件之间传送信息。分组可形成于事务层205和数据链路层210中，以将信息从传输组件输送至接收组件。随着所传输的分组流经其它层，它们被扩展有用于在那些层处理分组的附加信息。在接收侧发生逆过程并且将分组从其物理层220表示变换为数据链路层210表示，并最终(针对事务层分组)变换为可由接收设备的事务层205处理的形式。

[0038] 在一个实施例中，事务层205可提供设备的处理核和互连架构(诸如数据链路层210和物理层220)之间的接口。在此方面，事务层205的主要职责可包括分组(即，事务层分组或TLP)的组装和拆分。转换层205也可管理用于TLP的基于信用的流控制。在一些实现方式中，可利用分割事务，即具有按时间划分的请求和响应的事务，从而允许链路携带其它通信量，同时目标设备收集用于响应的数据，以及其它示例。

[0039] 基于信用的流控制可用于利用互连构造实现虚拟信道和网络。在一个示例中，设备可通告针对事务层205中的每个接收缓冲器的初始信用量。在链路的相对端处的外部设备，诸如图1中的控制器中枢115，可对由每个TLP消耗的信用的数量进行计数。如果事务未超过信用限度，则事务可被传输。在接收到响应时，信用量被恢复。这种信用方案的优点的一个示例是，假如信用限度未被计数，则信用返回的等待时间不影响性能，以及其它潜在优点。

[0040] 在一个实施例中，四个事务地址空间可包括配置地址空间、存储器地址空间、输入/输出地址空间和消息地址空间。存储器空间事务包括读取请求和写入请求中的一个或多个，以传输数据至存储器映射位置或从存储器映射位置传输数据。在一个实施例中，存储器空间事务能够使用两个不同的地址格式，例如，短地址格式(诸如32位地址)或长地址格式(诸如64位地址)。配置空间事务可用于访问连接到互连的各种设备的配置空间。对于配置空间的事务可包括读取请求和写入请求。消息空间事务(或简单地，消息)也可被定义以支持互连代理之间的带内通信。因此，在一个示例实施例中，事务层205可组装分组报头/有效载荷206。

[0041] 快速参考图3，图示出事务层分组描述符的示例实施例。在一个实施例中，事务描述符300可以是用于承载事务信息的机制。在此方面，事务描述符300支持系统中事务的识别。其它潜在的用途包括跟踪默认事务排序的修改和事务与信道的关联。例如，事务描述符300可包括全局标识符字段302、属性字段304和信道标识符字段306。在图示出的示例中，全局标识符字段302被描绘为包括本地事务标识符字段308和源标识符字段310。在一个实施

例中，全局事务标识符302对于所有未完成请求是独有的。

[0042] 根据一个实现方式，本地事务标识符字段308是由请求代理所生成的字段，并可对于要求针对该请求代理的完成的所有未完成请求是独有的。此外，在该示例中，源标识符310在互连层级内独有地识别请求方代理。由此，连同源ID 310，本地事务标识符308字段提供了在层级域内的事务的全局识别。

[0043] 属性字段304指定事务的特性和关系。在此方面，属性字段304潜在地被用于提供附加的信息，其允许对事务的默认处理的修改。在一个实施例中，属性字段304包括优先级字段312、保留字段314、排序字段316和无监听(no-snoop)字段318。这里，优先级子字段312可由启动器修改，以对事务分配优先级。保留属性字段314被保留用于将来使用或供应商定义的使用。使用优先级或安全属性的可能使用模型可使用保留属性字段来实现。

[0044] 在该示例中，排序属性字段316用于提供可选的信息，其传递可修改默认排序规则的排序的类型。根据一个示例实现方式，排序属性“0”表示应用默认排序规则，其中排序属性“1”表示不严格的排序，其中写入可在同一方向上传递写入，并且读取完成可在同一方向上传递写入。监听属性字段318被用于确定事务是否被监听。如图所示，信道ID字段306识别事务与之相关联的信道。

[0045] 返回到图2的讨论，链路层210(也称为数据链路层210)可充当事务层205和物理层220之间的中间级。在一个实施例中，数据链路层210的职责是提供可靠机构以用于在链路上的两个组件之间交换事务层分组(TLP)。数据链路层210的一侧接受由事务层205组装的TLP，应用分组序列标识符211(即识别号或分组号)，计算并应用错误检测代码(即CRC 212)，以及将已修改TLP呈递至物理层220以用于跨越物理至外部设备的传输。

[0046] 在一个示例中，物理层220包括逻辑子块221和电气子块222，以将分组物理地传输至外部设备。这里，逻辑子块221负责物理层221的“数字”功能。在此方面，逻辑子块可包括用以准备传出信息以便由物理子块222进行传输的传输区段，以及用以在将所接收的信息递送至链路层210之前对其进行识别和准备的接收器区段。

[0047] 物理块222包括发射器和接收器。发射器由逻辑子块221利用符号来提供，发射器串行化所述符号并将其传输至外部设备。接收器被提供有来自外部设备的串行化符号，并将接收到的信号变换成位流。该位流被去串行化并提供至逻辑子块221。在一个示例实施例中，采用了8b/10b传输代码，其中10位符号被传输/接收。这里，使用特殊符号来利用帧223来构建分组。此外，在一个示例中，接收器还提供从传入串行流恢复的符号时钟。

[0048] 如上所述，尽管参考协议栈(诸如PCIe协议栈)的具体实施例讨论了事务层205、链路层210和物理层220，但分层协议栈不限于此。实际上，任何分层协议可被包括/实现并采用本文讨论的特征。作为示例，表示为分层协议的端口/接口可包括：(1)用以组装分组的第一层，即事务层；用以排序分组的第二层，即链路层；以及用以传输分组的第三层，即物理层。作为具体示例，利用如本文描述的高性能互连分层协议。

[0049] 接下来参见图4，图示出串行点对点构造的示例实施例。串行点对点链路可包括用于传输串行数据的任何传输路径。在示出的实施例中，链路可包括两个低电压的差分驱动的信号对：传输对406/411和接收对412/407。由此，设备405包括传输逻辑406，其用以传输数据至设备410，以及接收逻辑407，其用以从设备410接收数据。换言之，在链路的一些实现方式中包括两个传输路径，即路径416和417，以及两个接收路径，即路径418和419。

[0050] 传输路径指代用于传输数据的任何路径，诸如传输线、铜线、光学线、无线通信信道、红外通信链路或其它通信路径。两个设备之间的连接，诸如设备405和设备410之间的连接，被称为链路，诸如链路415。链路可支持一个通道——每个通道表示一组差分信号对(一对用于传输，一对用于接收)。为了缩放带宽，链路可聚集多个通道，表示为xN，其中N为任何所支持的链路宽度，诸如1、2、4、8、12、16、20、24、32、64或更宽。

[0051] 差分对可指代两个传输路径，诸如线路416和417，以传输差分信号。作为示例，当线路416从低电压电平拨动(toggle)到高电压电平(即上升沿)时，线路417从高逻辑电平驱动至低逻辑电平(即下降沿)。差分信号潜在地表现出更好的电特性，诸如更好的信号完整性(即交叉耦合)、电压过冲/下冲、振铃，以及其它示例优点。这允许更好的定时窗口，其能够实现更快的传输频率。

[0052] 在一个实施例中，提供了新的高性能互连(HPI)。HPI可包括下一代缓存一致的、基于链路的互连。作为一个示例，HPI可被用在高性能计算平台中，诸如工作站或服务器，其包括在其中PCIe或另一互连协议通常被用于连接处理器、加速器、I/O设备等的系统中。然而，HPI不限于此。替代地，HPI可被用于本文描述的系统或平台中的任意中。此外，所开发的各个构思可被应用于其它互连和平台，诸如PCIe、MIPI、以太网、USB、QPI等。

[0053] 为支持多个设备，在一个示例实现方式中，HPI可包括指令集架构(ISA)不可知(即，HPI能够被实现于多个不同的设备中)。在另一情形中，HPI还可被用于连接高性能I/O设备，而不仅是处理器或加速器。例如，高性能PCIe设备可通过合适的转换桥(即HPI至PCIe)耦合于HPI。此外，HPI链路可以按各种方式(例如星形、环形、网状等)由许多基于HPI的设备(诸如处理器)所利用。图5图示出多个潜在的多插槽配置的示例实现方式。如所描绘的双插槽配置505可包括两个HPI链路；然而，在其它实现方式中，可利用一个HPI链路。对于更大的拓扑结构，可利用任何配置，只要标识符(ID)是可分配的并且存在某种形式的虚拟路径，以及其它附加的或替代的特征。如所示的，在一个示例中，四插槽配置510具有从每个处理器到另一个处理器的HPI链路。但在配置515中所示出的八插槽实现方式中，不是每个插槽都通过HPI链路相互直接连接。然而，如果虚拟路径或信道存在于处理器之间，则支持该配置。所支持的处理器的范围在原生域中包括2-32个处理器。除了其它示例之外，可通过使用多个域或节点控制器之间的其它互连来实现更高的处理器数量。

[0054] HPI架构包括分层协议架构的定义，在一些示例中包括协议层(一致、非一致、以及可选地，其它基于存储器的协议)、路由层、链路层和物理层。此外，HPI可还包括与功率管理器(诸如功率控制单元(PCU))有关的增强、针对测试和调试的设计(DFT)、错误处理、寄存器、安全性，以及其它示例。图5图示出示例HPI分层协议栈的实施例。在一些实现方式中，图5中图示出的层中的至少一些层可以是可选的。每个层处理其自身级别的信息粒度或信息份额(具有分组630的协议层605a、b，具有微片635的链路层610a、b，以及具有物理微片(phit)640的物理层605a、b)。注意，分组在一些实施例中可基于实现方式而包括部分微片、单个微片或多个微片。

[0055] 作为第一示例，物理微片640的宽度包括链路宽度到位的1对1映射(例如20位链路宽度包括20位的物理微片等)。微片可具有更大的尺寸，诸如184、192或200个位。注意，如果物理微片640为20位宽度并且微片635大小为184位，则采用物理微片640的分数来传输一个微片635(例如采用20位下的9.2个物理微片传输184位微片635或采用20位下的9.6个物理

微片传输192位微片，以及其它示例）。注意，基础链路的宽度在物理层可变化。例如，每个方向的通道的数量可包括2、4、6、8、10、12、14、16、18、20、22、24等。在一个实施例中，链路层610a、b能够在单个微片中嵌入多片不同的事务，并且一个或多个报头（例如1、2、3、4）可被嵌入在微片内。在一个示例中，HPI将报头划分到对应的槽中以使得能够实现微片中的去往不同节点的多个消息。

[0056] 在一个实施例中，物理层605a、b可负责在物理介质（电的或光的，等等）上快速传输信息。物理链路可以是两个链路层实体（诸如层605a和605b）之间的点对点。链路层610a、b可从上层抽取物理层605a、b并提供能力来可靠地传输数据（以及请求）并管理两个直接连接的实体之间的流控制。链路层还可负责将物理信道虚拟化成多个虚拟信道和消息类。协议层620a、b依赖于链路层610a、b来在将协议消息处理至物理层605a、b以用于跨越物理链路进行传输之前将协议消息映射至合适的消息类和虚拟信道中。链路层610a、b可支持多个消息，诸如请求、监听、响应、写回、非一致数据，以及其它示例。

[0057] HPI的物理层605a、b（或PHY）可被实现在电气层（即，连接两个组件的电导体）之上和链路层610a、b之下，如图6中图示出的那样。物理层和对应逻辑可驻留于每个代理上并连接相互分离的两个代理（A和B）上（例如在链路任一侧上的设备上）的链路层。本地电气层和远程电气层通过物理介质连接（例如导线、导体、光学等）。在一个实施例中，物理层605a、b具有两个主要阶段，初始化和操作。在初始化期间，连接对链路层不透明，并且信令可涉及定时状态和握手事件的组合。在操作期间，连接对链路层透明并且信令是在一定速度下，其中所有通道一起操作为单个链路。在操作阶段期间，物理层将微片从代理A传输至代理B并从代理B传输至代理A。连接也称为链路，并从链路层抽取一些物理方面，包括介质、宽度和速度，同时与链路层交换微片和当前配置的控制/状态（例如宽度）。初始化阶段包括次要阶段，例如轮询、配置。操作阶段也包括次要阶段（例如链路功率管理状态）。

[0058] 在一个实施例中，可实现链路层610a、b，以便在两个协议或路由实体之间提供可靠的数据传输。链路层可从协议层620a、b抽取物理层605a、b，并可负责两个协议代理（A、B）之间的流控制，并提供虚拟信道服务给协议层（消息类）和路由层（虚拟网络）。协议层620a、b和链路层610a、b之间的接口可通常处于分组级。在一个实施例中，在链路层处的最小传输单元称为微片（flit），其是指定数量的位，诸如192位或某其它额度。链路层610a、b依赖于物理层605a、b来将物理层的605a、b传输单元（物理微片）构建为链路层的610a、b传输单元（微片）。此外，链路层610a、b可被逻辑上分为两个部分，即发送器和接收器。一个实体上的发送器/接收器对可被连接于另一实体上的接收器/发送器对。流控制通常以微片和分组两者为基础来执行。错误检测和校正也潜在地以微片级为基础来执行。

[0059] 在一个实施例中，路由层615a、b可提供灵活和分布式的方法，以将HPI事务从源路由至目的地。该方案是灵活的，因为用于多个拓扑结构的路由算法可通过在每个路由器处的可编程路由表来指定（在一个实施例中，编程由固件、软件或其组合来执行）。路由功能性可以是分布式的；路由可通过一系列路由步骤来完成，其中每个路由步骤通过对源、中间或目的地路由器中任一处的表的查找来定义。在源处的查找可被用于将HPI分组注入至HPI构造。在中间路由器处的查找可被用于将HPI分组从输入端口路由至输出端口。在目的地端口处的查找可被用于将目的地HPI协议代理作为目标。注意，在一些实现方式中，路由层可能较薄，因为路由表以及因此的路由算法并未按规范被具体定义。这允许灵活性和各种使用

模型，包括要由系统实现方式来定义的灵活的平台架构的拓扑结构。路由层615a、b依赖于链路层610a、b，以便提供多达三个(或更多个)虚拟网络(VN)的使用——在一个示例中，两个免死锁VN，即VN0和VN1，具有在每个虚拟网络中定义的若干消息类。共享的自适应虚拟网络(VNA)可被定义于链路层中，但该自适应网络可能不直接暴露在路由概念中，因为每个消息类和虚拟网络可具有专用的资源和有保证的前向进展，以及其它特征和示例。

[0060] 在一个实施例中，HPI可以包括一致性协议层620a、b以支持从存储器缓存数据行的代理。希望缓存存储器数据的代理可以使用一致性协议来读取数据行以加载到其缓存中。希望在其缓存中修改数据行的代理可以在修改数据之前使用一致性协议获取行的所有权。在修改了行之后，代理可以遵循以下协议要求：将其保持在其缓存中，直到将行写回存储器或者将行包含在对外部请求的响应中。最后，代理可以满足外部请求以使其缓存中的行无效。协议通过规定所有缓存代理可遵循的规则来确保数据的一致性。它也为没有缓存的代理提供一致地读写存储器数据的手段。

[0061] 可以施行两个条件来支持利用HPI一致性协议的事务。首先，协议可以保持(例如，在每个地址的基础上)在代理缓存中的数据当中以及那些数据和存储器中的数据之间的数据一致性。非正式地，数据一致性可以指代理缓存中表示数据的最新值的每个有效数据行，并且在一致性协议分组中传输的数据可以表示发送数据时数据的最新值。当缓存或传输中没有有效的数据副本存在时，协议可以确保数据的最新值驻存在存储器中。其次，协议可以为请求提供良好定义的提交点。读取的提交点可以指示数据何时可用；并且对于写入，它们可以指示被写入的数据何时是全局可见的以及何时将被后续读取加载。协议可以支持针对一致性存储器空间中的可缓存和不可缓存(UC)请求的这些提交点。

[0062] 在一些实现方式中，HPI可利用嵌入时钟。时钟信号可被嵌入在使用互连来传输的数据中。利用嵌入在数据中的时钟信号，不同的和专用的时钟通道可被省略。这可以是有用的，例如，由于其可以允许更多的设备引脚专用于数据传输，特别是在用于引脚的空间非常珍贵的系统中。

[0063] 链路可被建立在互连的任一侧上的两个代理之间。发送数据的代理可以是本地代理并且接收数据的代理可以是远程代理。状态机可以被两个代理用来管理链路的各个方面。在一个实施例中，物理层数据路径可以将微片从链路层传输至电气前端。在一个实现方式中，控制路径包括状态机(也称为链路训练状态机或类似物)。状态机的动作和从状态的退出可取决于内部信号、定时器、外部信号或其它信息。实际上，一些状态，诸如少量的初始化状态，可具有定时器以提供超时值来退出状态。注意，在一些实施例中，检测指代检测通道的两个支路上的事件；但不一定同时。然而，在其它实施例中，检测指代通过参考代理来检测事件。作为一个示例，防反跳(debounce)指代对信号的持续断言。在一个实施例中，HPI支持非功能通道的事件中的操作。这里，通道可落在特定状态处。

[0064] 状态机中定义的状态可包括重置状态、初始化状态和操作状态，以及其它分类和子分类。在一个示例中，一些初始化状态可具有辅定时器，其用于超时时退出该状态(实质上由于未能在状态中取得进展而中止)。中止可包括诸如状态寄存器之类的寄存器的更新。一些状态还可具有(一个或多个)主定时器，其用于对状态中的主功能进行定时。其它状态可被定义为使得内部或外部信号(诸如握手协议)驱动从该状态到另一状态的转变，以及其它示例。

[0065] 状态机还可支持通过单个步骤的调试、对初始化中止的冻结以及测试器的使用。这里，状态退出可被延迟/保持，直到调试软件就绪。在一些实例中，退出可被延迟/保持直到辅助超时。在一个实施例中，动作和退出可基于训练序列的交换。在一个实施例中，链路状态机将运行在本地代理时钟域中并从一个状态转变至下一状态以与发射器训练序列边界相符合。状态寄存器可被用来反映当前状态。

[0066] 图7图示出在HPI的一个示例实现方式中由代理使用的状态机的至少一部分的表示。应当领会的是，包括于图7的状态表中的状态包括可能状态的非穷举列表。例如，一些转变被省略以简化图示。并且，一些状态可被组合、划分或省略，而其它状态可被添加。此类状态可包括：

事件重置状态：在温重置或冷重置事件时进入。恢复默认值。初始化计数器(例如，同步计数器)。可退出至另一状态，诸如另一重置状态。

[0067] 定时重置状态：用于带内重置的定时状态。可驱动预定义的电气有序集合(EOS)，因此远程接收器能够检测EOS并也进入定时重置。接收器具有保持电气设置的通道。可退出至代理以校准重置状态。

[0068] 校准重置状态：在没有通道上的信令(例如接收器校准状态)或关闭驱动器的情况下的校准。可以是基于定时器的状态中的预定时间量。可设置操作速度。可充当端口未被启用时的等待状态。可包括最小驻留时间。接收器调节或错开(stagger off)可基于设计来进行。可在超时和/或完成校准之后退出至接收器检测状态。

[0069] 接收器检测状态：检测(一个或多个)通道上的接收器的存在。可寻找接收器终止(例如接收器下拉插入)。可在指定值被设置时或在另一指定值未被设置时退出至校准重置状态。如果接收器被检测到或达到超时，则可退出至发射器校准状态。

[0070] 发射器校准状态：用于发射器校准。可以是被分配用于发射器校准的定时状态。可包括通道上的信令。可连续地驱动EOS，诸如电气空闲退出有序集合(EIEIOS)。当完成校准或定时器期满时可退出至顺从状态。如果计数器已经期满或已经出现辅助超时，则可退出至发射器检测状态。

[0071] 发射器检测状态：限定有效信令。可以是握手状态，其中代理基于远程代理信令完成动作并退出至下一状态。接收器可限定来自发射器的有效信令。在一个实施例中，接收器寻找唤醒检测，并且如果在一个或多个通道上防反跳，则在其它通道上寻找。发射器驱动检测信号。响应于为所有通道完成的防反跳和/或超时或如果未完成所有通道上的防反跳并且存在超时，则可退出至轮询状态。这里，一个或多个监视通道可保持唤醒，以对唤醒信号防反跳。并且如果被防反跳，则潜在地对其它通道防反跳。这可以使得能够实现低功率状态中的节电。

[0072] 轮询状态：接收器适配，(例如，通过初始化漂移缓冲器)来锁定位，(例如，通过识别符号边界)来锁定字节，以及(例如，通过锁定到电气有序集合(EOS)和/或训练序列报头)来锁定训练序列边界。然后可以对通道去扭斜。也可以完成握手以退出轮询至若干潜在状态中的一个。例如，可(通过代理)导致到链路宽度状态、顺从状态、环回标记状态或重置状态中的任何一个的退出。握手可包括各种定义的接收器动作或条件的完成以及应答消息(ACK)的发送以触发退出到与动作的完成的集合以及ACK对应的状态。对于轮询去扭斜，可针对最高速度以第一长度以及针对慢速度以第二长度覆盖远程发射器处的通道到通道的

扭斜。去扭斜可在慢速模式以及操作模式中执行。接收器可具有对通道到通道的扭斜进行去扭斜的特定最大值，诸如8、16或32的扭斜间隔。在一些实现方式中，接收器动作也可包括等待时间修复。在一个实施例中，接收器动作可在有效通道映射的成功去扭斜上完成。在一个示例中，可在接收器已经完成其动作之后在利用应答接收多个连续训练序列报头并且具有应答的多个训练序列被传输时实现成功的握手。

[0073] 顺从状态：从轮询状态进入。代理可以做为顺从主设备或从设备以用于验证目的。设备(用作主设备)可以将顺从模式发送到另一设备(从设备)，并且从设备可以在将其重定时到其本地时钟之后(例如，在不撤销任何极性逆转或通道反转的情况下)来环回模式。当环回不起作用或不满意时，顺从模式可用于表征通道的一些子集上的模拟前端的操作。例如，在成功的字节锁定、TS锁定、去扭斜、等待时间测试以及依赖于几个数字逻辑的正确运转的其它测试上，可以预先调节进入环回。在完成了位锁定的情况下，可以进入顺从，并且顺从可用于抖动或噪声调查、调试、探索链路、以及其它调节。换句话说，如果轮询的结果不允许直接进入环回，则顺从可以作为轮询的替代退出。顺从状态可以利用来自主设备的发射器驱动超序列。接收器在监视器通道上寻找唤醒、对唤醒防反跳、丢弃坏通道、适配以及位锁定等。从设备发射器可以驱动顺从模式，直到其接收器动作完成。于是环回是重新定时的和未经去扭斜的。从设备接收器进行类似的监视和防反跳等动作。退出可以是到重置状态(诸如定时重置)或到环回模式状态以开始测试，以及其它示例。在退出到环回模式状态的情况下，可以将主设备(例如，由软件控制器)发送到环回模式状态以尝试更专用的模式。

[0074] 代理环回标记状态：环回标记是代理状态，但与其它代理状态不同，主设备和从设备的动作和退出可以不同。环回从设备可以撤消任何极性逆转和/或通道反转，但不可解扰或重新加扰已环回的位。应答交换不可适用于从设备，因为它正在进行环回。由于从设备可在环回到符号边界上之前去扭斜，因此主设备可不被强制重新进行字节锁定或重新去扭斜，但主设备可重新锁定训练序列，以避免锁定到一些别名。这样做的手段可包括LFSR的补种(re-seed)，比较TS和/或EIEOS或这些的某组合。SDS的结束标志着环回设置的结束以及模式生成、检查和计数的开始。

[0075] 代理环回模式状态(或阻断链路状态)：从环回标记状态或顺从状态进入。在环回模式状态下，替代控制模式，主设备发射器可以发送附加的专用模式以补充顺从或环回模式状态中的那些环回。接收器可以接收环回模式中的专用模式，并检查接收到的模式中的错误。对于发射器适配，两个代理可以都是主设备。在预定时段内，发射器可以发送模式，并且远程接收器可以比较此模式并确定记录在诸如寄存器之类的存储元件中的接收到的模式的度量或价值符号。比较方法和度量可以取决于设计(例如，具有抖动注入的BER)。在这段时间结束时，两个代理都可以退出到重置以便返回信道检查度量并设置发射器适配的下一次迭代。

[0076] 链路宽度状态：代理与到远程发射器的最终通道映射进行通信。接收器接收信息并解码。接收器可在第二结构中的先前通道映射值的检查点之后记录结构中的配置通道映射。接收器还可利用应答(“ACK”)进行响应。可发起带内重置。作为一个示例，第一状态发起带内重置。在一个实施例中，响应于ACK来执行退出至下一状态，诸如微片配置状态。此外，在进入低功率状态之前，如果唤醒检测信号出现的频率降至指定值以下(例如每单位间隔(UI)数量1个，诸如4K UI)，则重置信号也可被生成。接收器可保持当前的和先前的通道映

射。基于具有不同值的训练序列，发射器可使用不同的通道组。在一些实施例中，通道映射可不修改一些状态寄存器。

[0077] 微片锁定配置状态：当发射器和接收器已均退出至阻断链路状态或其它链路状态时，由发射器进入但该状态被视为退出(即辅助超时假设情况)。在一个实施例中，至链路状态的发射器退出包括在接收到有轨迹的对齐信号之后的训练序列(TS)边界和数据序列(SDS)的开始。这里，接收器退出可基于从远程发射器接收SDS。该状态可以是从代理到链路状态的桥。接收器识别SDS。如果SDS在解扰器被初始化之后被接收，则接收器可退出至阻断链路状态(BLS)(或控制窗口)。如果超时发生，则退出可以是到重置状态。发射器利用配置信号驱动通道。基于条件或超时，发射器退出可以是到重置、BLS或其它状态。

[0078] 传输链路状态：链路状态。微片被发送至远程代理。可从阻断链路状态进入并在诸如超时的事件上返回到阻断链路状态。发射器传输微片。接收器接收微片。还可退出至低功率链路状态。在一些实现方式中，传输链路状态(TLS)可以被称为L0状态。

[0079] 阻断链路状态：链路状态。发射器和接收器以统一的方式进行操作。可以是定时状态，在此期间链路层微片被拖延而物理层信息被传送至远程代理。可退出至低功率链路状态(或基于设计的其它链路状态)。在一个实施例中，阻断链路状态(BLS)周期性地出现。该周期被称为BLS间隔并且可被定时，以及可在慢速度和操作速度之间有所不同。注意，可周期性地阻断链路层发送微片，使得诸如在传输链路状态或部分宽度传输链路状态期间可发送一定长度的物理层控制序列。在一些实现方式中，阻断链路状态(BLS)可称为L0控制(或L0c)状态。

[0080] 部分宽度传输链路状态：链路状态。可通过进入部分宽度状态而省电。在一个实施例中，非对称部分宽度指代具有不同的宽度的双向链路的每个方向，其可在一些设计中被支持。在图9的示例中示出启动器的示例，诸如发射器，其发送部分宽度指示以进入部分宽度传输链路状态。这里，部分宽度指示被发送，同时利用第一宽度在链路上传输，以将链路进行转变以在新的第二宽度下进行传输。失配可导致重置。注意，可不改变速度但可改变宽度。因此，微片潜在地以不同的宽度被发送。可类似于以逻辑方式传输链路状态；但由于存在更小的宽度，因此可花费较长时间传输微片。可退出至其它链路状态，诸如基于特定的接收和发送消息的低功率链路状态或基于其它事件的部分宽度传输链路状态或链路阻断状态的退出。在一个实施例中，如时序图中所示，发射器端口可以交错的方式将空闲通道关闭以提供更好的信号完整性(即，噪声减轻)。这里，非可重试微片，诸如空微片，可在链路宽度正改变的时段期间被利用。对应接收器可丢弃这些空微片并以交错方式关闭空闲通道，以及在一个或多个结构中记录当前的和先前的通道映射。注意，状态和相关联的状态寄存器可保持不变。在一些实现方式中，部分宽度传输链路状态可称为部分L0(或L0p)状态。

[0081] 退出部分宽度传输链路状态：退出部分宽度状态。在一些实现方式中，可使用或可不使用阻断链路状态。在一个实施例中，发射器通过在空闲通道上发送部分宽度退出模式以对其进行训练和去扭斜来发起退出。作为一个示例，退出模式开始于EIEOS，其被检测并被防反跳至通道准备好开始进入完全传输链路状态的信号，并可结束于空闲通道上的SDS或快速训练序列(FTS)。在退出序列期间的任何失败(接收器动作，诸如在超时之前未完成去扭斜)停止微片传输至链路层并断言重置，其通过在下一阻断链路状态出现时重置链路来处理。SDS还可将通道上的加扰器/解扰器初始化为合适的值。

[0082] 低功率链路状态：为更低功率状态。在一个实施例中，其是比部分宽度链路状态更低的功率，因为在该实施例中的信令在所有通道和双方向上被停止。发射器可使用阻断链路状态以用于请求低功率链路状态。这里，接收器可解码请求并利用ACK或NAK进行响应；否则重置可被触发。在一些实现方式中，低功率链路状态可称为L1状态。

[0083] 在一些实现方式中，可促进状态转变以允许状态被绕过，例如，当状态的状态动作(诸如某些校准和配置)已被完成时。链路的先前状态结果和配置可被存储并重用于链路的后续初始化和配置中。与重复这种配置和状态动作相反，对应状态可被绕过。然而，实现状态绕过的传统系统常常实现复杂设计和昂贵的验证逸出。与使用传统绕过相反，在一个示例中，HPI可利用某些状态中的短定时器，诸如在状态动作不需要被重复的情况下。这可以潜在地允许更多一致和同步的状态机转变，以及其它潜在的优点。

[0084] 在一个示例中，基于软件的控制器(如，通过用于物理层的外部控制点)可以使得能够实现用于一个或多个特定状态的短定时器。例如，对于动作已被执行和存储的状态，该状态可被短定时，以促进从该状态至下一状态的快速退出。然而，如果先前状态动作失败或无法在短定时器持续时间内被应用，则状态退出可被执行。此外，控制器可禁用短定时器，例如，当状态动作应当被重新执行时。长的或默认的定时器可被设置用于每个相应状态。如果在该状态下的配置动作无法在长定时器内被完成，则状态退出可以发生。长定时器可被设置为合理的持续时间以便允许状态动作的完成。相反，短定时器可能要短得多，从而使在一些情况下不可能在没有参考回到先前执行的状态动作的情况下执行所述状态动作，以及其它示例。

[0085] 在一些实例中，在链路的初始化(或重新初始化)期间，当代理通过状态机朝向操作链路状态前进时，可发生导致状态重置(例如，重置到重置状态或其它状态)的一个或多个故障或状态退出。实际上，链路的初始化可以在不完成初始化和进入链路状态的情况下循环通过一个或多个状态。在一个示例中，计数可以维持在链路的初始化内的状态转变中的非生产性循环的数量。例如，每当初始化返回到重置状态而未达到链路状态时，可以使计数器增量。一旦链路成功进入链路状态，用于链路的计数器就可以被重置。这种计数器可以由链路两侧的代理维持。此外，可以例如通过利用一个或多个外部控制点的基于软件的控制器来设置阈值。当非生产性循环的计数满足(或超过)所定义的阈值时，链路的初始化可以被暂停(例如，在重置状态或之前设置和保持)。在一些实现方式中，为了重新开始初始化并从暂停状态释放初始化，基于软件的控制器可以触发链路的重启或重新初始化。在一些实例中，基于软件的工具可以分析暂停的初始化的性质，并执行诊断，设置寄存器值和执行其它操作，从而防止初始化的进一步循环。实际上，在一些实现方式中，控制器可以与重启暂停的链路初始化相关联地设置更高的计数器阈值或甚至推翻计数器，以及其它示例。

[0086] 在HPI的一些实现方式中，超序列可被定义，每个超序列对应于相应状态或进入到相应状态/从相应状态进入/退出至相应状态/从相应状态退出。超序列可包括数据集合和符号的重复序列。在一些实例中，序列可重复，直到状态或状态转变的完成或对应事件的传送为止，以及其它示例。在一些实例中，超序列的重复序列可根据定义的频率(诸如定义的单位间隔(UI)的数量)进行重复。单位间隔(UI)可对应于用于在链路或系统的通道上传输单个位的时间间隔。在一些实现方式中，重复序列可始于电气有序集合(EOS)。因此，EOS的实例可预期依据预定义的频率进行重复。这种有序集合可被实现为定义的16字节代码，其

可按十六进制格式来表示，以及其它示例。在一个示例中，超序列的EOS可以是EIEIOS。在一个示例中，EIEOS可类似低频率时钟信号（例如，预定义数量的重复FF00或FFF000十六进制符号，等等）。预定义的数据集合可遵循EOS，诸如预定义数量的训练序列或其它数据。这种超序列可被用于状态转变中，包括链路状态转变和初始化，以及其它示例。

[0087] 在互连的一些实现方式中（诸如在QPI中），可以促成和实现串行数据链路的终止，诸如当链路被重置或初始化时。这种方法可能将复杂性和时间引入到链路的初始化中。在HPI的一些实现方式中，链路的终止可以被维持，包括在链路的重置或重新初始化期间。此外，HPI可以允许设备的热插拔。当（通过热插拔或其它方式）引入另一设备时，在其上添加新的远程代理的通道的电压特性将改变。本地代理可以感测到通道电压中的这些改变，以检测远程代理的存在并提示链路的初始化。可以在状态机中定义状态机状态和定时器，以在不终止的情况下协调链路的检测、配置和初始化。

[0088] 在一个实现方式中，HPI可以支持带内重置的重新初始化，而不会通过由接收代理针对传入信令对通道的筛选来改变终止值。信令可用于识别良好的通道。作为示例，可以针对要由发射器设备发送的预定义信号的集合中的任何一个来筛选通道，以促进链路的发现和配置。在一个示例中，可以定义与一个或多个初始化或重新初始化任务对应的超序列。预定义的序列可以包括电气空闲退出有序集合（EIEOS），随后是附加的序列数据。在一些实例中，当通道任一侧的每个设备变为活动时，设备可以开始发送与特定初始化状态等对应的超序列。在一个实施例中，可以支持两种类型的引脚重置；上电（或“冷”）重置和温重置。由软件发起的重置或在一个代理上发起（在物理层或另一层中）的重置可以在带内传送到另一代理。然而，由于使用嵌入式时钟，可以通过使用诸如特定电气有序集合或EIOS之类的有序集合的与另一代理的通信来处理带内重置。

[0089] 可以在初始化期间发送有序集合，并且可以在初始化之后发送PHY控制序列（或“阻断链路状态”）。阻断链路状态可以阻断链路层发送微片。作为另一示例，链路层通信量可被阻断以发送可在接收器处丢弃的几个NULL（空）微片。

[0090] 如以上所介绍的，在一个实施例中，初始化可最初以慢速完成，之后是以快速的初始化。慢速下的初始化将默认值用于寄存器和定时器。然后软件使用慢速链路来设立寄存器、定时器和电气参数，并清除校准信号（semaphore）来为快速初始化铺平道路。作为一个示例，初始化可包括诸如重置、检测、轮询、顺从、和配置之类的状态或任务以及潜在的其它状态或任务。

[0091] 在一个示例中，链路层阻断控制序列（即阻断链路状态（BLS）或L0c状态）可包括定时状态，在此期间链路层微片被拖延，同时PHY信息被传送至远程代理。这里，发射器和接收器可启动阻断控制序列定时器。并且在定时器期满时，发射器和接收器可退出阻断状态并可采取其它动作，诸如退出至重置，退出至不同的链路状态（或其它状态），包括允许跨越链路发送微片的状态。

[0092] 在一个实施例中，链路训练可被提供并包括诸如与所定义的超序列有关地发送加扰训练序列、有序集合以及控制序列中的一个或多个。训练序列符号可包括以下中的一个或多个：报头、保留部分、目标等待时间、对数、物理通道映射代码参考通道或通道组、以及初始化状态。在一个实施例中，可发送具有ACK或NAK的报头，以及其它示例。作为示例，训练序列可作为超序列的一部分被发送并可被加扰。

[0093] 在一个实施例中，有序集合以及控制序列不被加扰或交错并在所有通道上同样、同时且完整地传输。有序集合的有效接收可包括检查有序集合的至少一部分(或针对部分有序集合的整个有序集合)。有序集合可包括电气有序集合(EOS)，诸如电气空闲有序集合(EIOS)或EIEOS。超序列可包括数据序列(SDS)或快速训练序列(FTS)的开始。这种集合和控制超序列可被预定义并可具有任何模式或十六进制表示以及任何长度。例如，有序集合和超序列可以是8字节、16字节或32字节等的长度。作为示例，FTS可附加地被用于在部分宽度传输链路状态的退出期间的快速位锁定。注意，FTS定义可以是按通道的并可利用FTS的轮转版本。

[0094] 在一个实施例中，超序列可包括在训练序列流中的EOS(诸如EIEOS)的插入。在一个实现方式中，当信令开始时，通道以交错形式上电。然而，这可能导致最初的超序列在一些通道上的接收器处看起来被截断。然而，超序列可在短间隔(例如近似一千个单位间隔(或～1KUI))上重复。训练超序列可附加地被用于以下中的一个或多个：去扭斜、配置、以及用于传送初始化目标、通道映射等。EIEOS可用于以下中的一个或多个：通道从非活动到活动状态的转变、筛选良好的通道、识别符号和TS边界，以及其它示例。

[0095] 在一个实施例中，时钟可被嵌入在数据中，因此没有单独的时钟通道。通过通道发送的微片可被加扰，以促进时钟恢复。作为一个示例，接收器时钟恢复单元可将采样时钟递送至接收器(即接收器从数据恢复时钟并将其用于对传入数据进行采样)。接收器在一些实现方式中连续地适应于传入位流。通过对时钟进行嵌入，可潜在地减少引脚分配(pinout)。然而，在带内数据中嵌入时钟可改变对待带内重置的方式。在一个实施例中，可在初始化之后利用阻断链路状态(BLS)。并且，电气有序集合超序列可在初始化期间被用来促进重置(例如，如上所述)以及其它考虑。嵌入时钟可以在链路上的设备之间是公共的，并且公共操作时钟可在链路的校准和配置期间被设置。例如，HPI链路可参考具有漂移缓冲的公共时钟。这种实现方式可相比非公共参考时钟中所使用的弹性缓冲实现更低的等待时间，以及其它潜在的优点。此外，参考时钟分布区段可被匹配到指定限度内。

[0096] 在一些实现方式中，HPI可支持具有在一些情况下并非标称通道宽度的倍数的宽度的微片(例如，作为纯例证性示例，使用192位的微片宽度和20个通道)。实际上，在允许部分宽度传输状态的实现方式中，通过其传输微片的通道的数量可波动(甚至在链路寿命期间)。例如，在一些实例中，微片宽度在一个瞬间可能是活动通道数量的倍数，但在另一瞬间可能不是活动通道数量的倍数(例如，随着链路改变状态和通道宽度)。在通道数量不是当前通道宽度的倍数的实例中(例如，20个通道上的192位的微片宽度的示例)，在一些实施例中，连续微片可被配置为被传输以在通道上重叠，从而保留带宽(例如，传输在20个通道上重叠的五个连续192位微片)。

[0097] 图8图示出在多个通道上重叠的连续微片的传输的表示。例如，图8示出了在20个通道链路(由列0-19表示的通道)上发送的五个重叠192位微片的表示。图8的每个单元格表示包括在通过4UI跨度发送的微片中的相应“半字节”或四个位的分组(例如，位4n+3:4n)。例如，192位微片可被划分成48个四位半字节。在一个示例中，半字节0包括位0-3，半字节1包括位4-7等。半字节中的位可被发送以便重叠或被交织(例如，“拌和”)，使得微片的更高优先级字段被更早地呈现，错误检测属性(例如，CRC)被保持，以及其它考虑。实际上，拌和方案还可提供的是一些半字节(以及其相应位)被无序发送(例如，如图8和9的示例中那

样)。在一些实现方式中,拌和方案可取决于链路层的架构和链路层中使用的微片的格式。

[0098] 具有并非活动通道倍数的长度的微片的位(或半字节)可被拌和,诸如根据图8的示例。例如,在第一4UI期间,半字节1、3、5、7、9、12、14、17、19、22、24、27、29、32、34、37、39、42、44和47可被发送。半字节0、2、4、6、8、11、13、16、18、21、23、26、28、31、33、36、38、41、43和46可在下一4UI期间被发送。在UI 8-11中,仅8个半字节保留于第一微片。第一微片的这些最后的半字节(即,10、15、20、25、30、40、45)可与第二微片的第一半字节(即,半字节2、4、7、9、12、16、20、25、30、35、40、45)同时发送,使得第一和第二微片重叠或拌和。利用这种技术,在本示例中,五个完整的微片可在48UI中被发送,其中每个微片通过分数的9.6个UI时段被发送。

[0099] 在一些实例中,拌和可产生周期性的"清洁的"微片边界。例如,在图8的示例中,开始的5个微片边界(第一微片的顶行)还可称为清洁的微片边界,因为所有通道均传输来自同一微片的起始的半字节。代理链路层逻辑可被配置为识别通道的拌和并可从被拌和的位重构微片。另外,物理层逻辑可包括用于基于当时使用的通道数量来识别何时以及如何拌和微片数据流的功能性。实际上,在从一个链路宽度状态至另一个的转变中,代理可将自身配置为识别将如何采用数据流的拌和。

[0100] 如上所述,链路可在通道宽度之间转变,在一些实例中,以原始或完整的宽度操作,并稍后转变为利用更少通道的部分宽度(以及从其转变)。在一些实例中,微片的定义宽度可以是可被通道数量除尽的。例如,图9的示例图示出这样的一个示例,其中先前示例的192位微片通过8通道链路来传输。如图9所表示的,192位微片的4位半字节可在8通道上均匀地分布和传输(即,由于192是8的倍数)。实际上,当以8通道部分宽度操作时,单个微片可在24个UI上发送。此外,在图9的示例中每个微片边界可以是清洁的。尽管清洁的微片边界可简化状态转变、确定性和其它特征,但是允许拌和和偶然的参差的微片边界可允许链路上浪费带宽的最小化。

[0101] 另外,尽管图9的示例将通道0-7示为在部分宽度状态中保持活动的通道,但可以潜在地使用8通道的任何集合。还要注意,上述示例仅出于例证性目的。微片可以潜在地被定义为具有任何宽度。链路还可以潜在地具有任何链路宽度。此外,系统的拌和方案可根据微片的格式和字段、系统中优选的通道宽度(以及其它考虑和示例)来灵活地构建。

[0102] 在等待时间未在链路层处产生等待时间修复错误或超时以及其它考虑的情况下,HPI PHY逻辑层的操作可独立于底层传输介质。

[0103] 链路层

链路层可以从协议层中抽象出物理层,处理两个协议代理之间的流控制,并提供虚拟信道服务给协议层(消息类)和路由层(虚拟网络)。在一些实现方式中,链路层可以处理称为微片的固定的信息份额。在一个示例中,微片可以被定义为192位长度。然而,可以在不同的变型中使用诸如81-256(或更多)的位的任何范围。诸如192位的大的微片大小可以包括格式、循环冗余校验(CRC)、纠错码(ECC)和其它特征。例如,较大的微片长度也可以允许CRC字段被扩展(例如,至16位)以处理较大的微片有效载荷。用于传送单个微片的物理微片或单元间隔(UI)的数量(例如,用于传送单个位或者物理微片等的时间)可以随着链路宽度而变化。例如,除了其它潜在示例之外,20通道或位链路宽度可以潜在地在9.6个UI中传送单个192位微片,而8通道链路宽度在24个UI中传送相同的微片。也可以基于微片进行链路层

信用化和协议分组化。

[0104] 图10图示出用于8通道链路宽度的一般化微片的表示1000。表示1000的每列可以象征链路通道，并且每行可以象征相应的UI。在一些实现方式中，单个微片可以被细分到两个或更多个槽中。每个槽中可以包含不同的消息或链路层报头，以允许在单个微片中发送与潜在不同的事务对应的多个不同的(并且在一些情况下独立的)消息。此外，除了其它示例之外，包括在单个微片的槽中的多个消息也可以去往不同的目的地节点。例如，图10的示例图示出具有三个槽的微片格式。阴影部分可以表示微片的被包括在相应槽中的部分。

[0105] 在图10的示例中，提供三个槽，槽0、1和2。槽0可以被提供有72位的微片空间，其中22位专用于消息报头字段并且50位用于消息有效载荷空间。槽1可以被提供有70位的微片空间，其中20位专用于消息报头字段并且50位用于消息有效载荷空间。消息报头字段空间之间的差异可被优化成提供某些消息类型将被指定为包括在槽0中(例如，其中使用更多消息报头编码)。可以提供第三槽(槽2)，其占用比槽0和1小得多的空间，在这种情况下使用18位的微片空间。槽2可以被优化成处理那些不采用较大消息有效载荷的消息，诸如应答、信用返回等。此外，可以提供浮动的有效载荷字段，其允许替代地应用附加的11位来补充槽0或槽1的有效载荷字段。

[0106] 继续图10的具体示例，其它字段对于微片可以是全局的，(即，跨微片应用而不是应用于特定槽)。例如，报头位可以与可用于指定诸如微片的虚拟网络之类的信息、识别微片要如何进行编码以及其它示例的4位微片控制字段一起被提供。此外，可以诸如通过16位循环CRC字段以及其它潜在示例来提供错误控制功能性。

[0107] 可以定义微片格式以便优化链路层上的消息的吞吐量。一些传统协议已经使用了无槽的较小的微片。例如，在QPI中，使用了80位微片。虽然较大(例如，192位微片)的微片吞吐量可能较低，但可以通过优化微片数据的使用来增加消息或分组吞吐量。例如，在QPI的一些实现方式中，无论消息大小或类型如何，都使用整个80位的微片空间。通过将较大的微片细分成预定长度和字段的槽，即使在有时不使用可用槽中的一个或多个的情况下，也可以优化192微片长度，从而实现更高的效率。实际上，可以假设链路层通信量包括许多不同类型的消息和通信量，包括具有变化的报头长度和字段的消息和分组。可以定义在微片中定义的槽的相应长度和组织，以便与各种消息的统计或预期频率以及这些消息的需求相对应。例如，可以为每个小槽定义两个较大的槽，以适应使用这些较大消息类型和报头长度的消息传递的预期统计频率以及其它示例。此外，还可以诸如通过浮动的有效载荷字段提供灵活性，以进一步适应变化的通信量，如图10的示例中那样。在一些实例中，微片格式可以是固定的(包括专用于微片中的特定槽的位)。

[0108] 在图10的示例中，一般可以为微片提供“Hdr”字段，并且该字段表示微片的报头指示。在一些实例中，Hdr字段可以指示微片是报头微片还是数据微片。在数据微片中，微片仍然可以保持被分槽，但省略某些字段的使用或用有效载荷数据来替换某些字段的使用。在一些情况下，数据字段可以包括操作码和有效载荷数据。在报头微片的情况下，可以提供各种报头字段。在图10的示例中，可以为每个槽提供“Oc”字段，Oc字段表示操作码。类似地，一个或多个槽可以具有对应的“msg”字段，其表示要包括在槽中的对应分组的消息类型(在该槽被设计为处理这种分组类型等的情况下)。“DNID”字段可以表示目的地节点ID，“TID”字段可以表示事务或跟踪器ID，“RHTID”字段可以表示请求器节点ID或归属跟踪器ID，以及其

它潜在字段。此外，一个或多个槽可以被提供有有效载荷字段。此外，除了其它示例之外，可以在微片中包括CRC字段以提供微片的CRC值。

[0109] 在一些实现方式中，链路宽度可以在链路的寿命期间变化。例如，物理层可以在链路宽度状态之间转变，诸如从或向完整或原始通道宽度以及不同或部分通道宽度。例如，在一些实现方式中，可以将链路初始化成在20个通道上传输数据。之后，链路可以转变到仅有8个通道被活动地使用的部分宽度传输状态，以及许多其它的潜在示例。这样的通道宽度转变可以例如与由一个或多个功率控制单元(PCU)管控的电力管理任务结合地利用，以及其它示例。

[0110] 如上所述，链路宽度可以影响微片吞吐率。图11是通过8通道链路发送的示例192位微片的表示，得出在24UI处的微片的吞吐量。此外，如图11的示例所示，在一些实例中，可以乱序地发送微片的位，例如以在传输中更早地发送时间更敏感的字段(例如，微片类型字段(例如，数据或报头微片)、操作码等)，以保留或促进特定错误检测或在微片中体现的其它功能性，以及其它示例。例如，在图11的示例中，位191、167、143、119、95、71、47和23在传送的第一个UI(即，UI0)期间并行地在通道L7至L0上被发送，而位168、144、120、96、72、48、24和0在微片传送的第24个(或最后的)UI(即，UI23)期间被发送。应当领会到，在其它实现方式和示例中可以使用其它排序方案、微片长度、通道宽度等。

[0111] 在一些实例中，微片的长度可以是活动通道数量的倍数。在此类实例中，微片可以在所有活动通道上均匀地传输，并且微片的传送可基本上同时以清洁(即，不重叠)的边界结束。例如，如图8的表示所示，可以认为以4位的连续分组或“半字节”的方式传输微片的位。在此示例中，一个192位的微片将通过8通道链路进行传送。由于192是8的倍数，所以可以在24个UI中通过8通道链路干净地传送整个微片。在其它实例中，微片宽度可能不是活动通道数量的倍数。例如，图9示出了在20个通道上传送的示例192位的另一表示。由于192不能被20均匀地除尽，所以完整微片的传送将需要非整数个间隔(例如，9.6个UI)。在这种情况下，与浪费在传送的第10个UI期间未被使用的“额外”通道相反，可以用前一个微片的最后的位来传送第二个重叠的微片。在一些实现方式中，微片的这种重叠或拌和可能导致参差的微片边界和微片位的乱序发送。用于传送的模式可以被配置为允许微片中时间更敏感的字段在微片中被更早地传送、错误检测和校正的保留，以及其它考虑。可以在物理层和链路层中的一个或两个中提供逻辑，以根据这样的模式传送微片位，并且基于当前链路宽度在模式之间动态地改变。可以提供其它的逻辑来重排和重构来自这种拌和或有序位流的微片，以及其它示例。

[0112] 在一些实现方式中，微片可以被表征为报头微片(例如，承载分组报头数据)或数据微片(例如，承载分组有效载荷数据)。回到图10，可以定义包括三(3)个不同槽(例如，0、1和2)的微片格式，允许多达三个报头在单个微片中传送(例如，每个槽中的一个报头)。在图10的示例中，提供三个槽，槽0、1和2。槽0可以被提供有72位的微片空间，其中22位专用于消息报头字段并且50位用于消息有效载荷空间。槽1可以被提供有70位的微片空间，其中20位专用于消息报头字段并且50位用于消息有效载荷空间。消息报头字段空间之间的差异可被优化成提供某些消息类型将被指定为包括在槽0中(例如，其中使用更多消息报头编码)。可以提供第三槽(槽2)，其占用比槽0和1小得多的空间，在这种情况下使用18位的微片空间。槽2可以被优化成处理那些不采用较大消息有效载荷的消息，诸如应答、信用返回等。此外，

可以提供浮动的有效载荷字段，其允许替代地应用附加的11位来补充槽0或槽1的有效载荷字段。

[0113] 在一些实现方式中，通过允许字段在两个槽之间浮动，可以根据需要为某些消息提供额外的位，同时仍然保持在预定义的微片长度(例如，192位)内并最大化带宽的利用。转到图19的示例，在8通道数据链路上示出了示例192位微片的两个实例1905、1910。在一个实例中，微片(例如，1905)可以包括三个槽，槽0、1和2。槽0和1中的每一个可以包括50位有效载荷字段。可以提供浮动字段以替换地将任一个槽0或槽1的有效载荷字段扩展浮动字段的字段长度(例如，11位)。使用浮动字段还可以扩展通过所定义的多槽微片格式提供的效率增益。微片内的槽的尺寸确定以及可以放置在每个槽中的消息的类型可以潜在地提供增加的带宽(即使在微片比率降低的情况下)。

[0114] 在图10的特定示例中，可以对可以使用槽1和2的消息进行优化，以减少为编码这些槽的操作码而留出的位数。当具有槽0可以提供的更多位的报头进入链路层时，可以提供分槽算法以允许其接管槽1有效载荷位来实现附加空间。也可以提供特殊控制(例如LLCTRL)微片，其消耗针对其需要的位值得的所有三个槽。在链路部分地繁忙的情况下，也可以存在分槽算法以允许使用各个槽同时其它槽不携带信息。

[0115] 在一个实施例中，链路层可附加地定义可用于例如调试消息和其它用途的特殊控制微片。在一个示例中，当设置启用调试控制字段时，可以由链路层发送LLCTRL-DEBUG微片。当该位未设置时，可能不会在链路上传输LLCTRL-DEBUG微片。调试分组对于暴露通过HPI连接的(否则不可访问的)设备的内部状态可以是重要的。调试分组的内容也可以是实现方式特定的。内容可以包括分支信息(源IP和目标IP)、时间戳、内部事件触发的指示等之类的内容。暴露的数据可以是例如通过监视诸如用于后处理和故障分析的逻辑分析器的设备。图12中图示出调试消息类型的示例微片编码。

[0116] 在一些实现方式中，可以利用诸如HPI的通用I/O互连(GPIO)架构的原理和协议来实现缓冲存储器接口和对应的协议。例如，上面列出的物理层和链路层定义也可以在缓冲存储器协议中实现。实际上，用于支持GPIO协议的物理层和链路层的逻辑可以在支持缓冲存储器协议的接口处重复使用。缓冲存储器协议还可以共享消息类，诸如请求、响应和写回消息类，以及其它示例。虽然缓冲存储器协议消息内的操作码值可以不同于HPI(或另一GPIO协议)中的那样来解释，但是在缓冲存储器协议和它被构建于其上的GPIO互连中都可以使用相同的通用分组格式。实际上，附加的分组和微片字段可以对于缓冲存储器协议和GPIO互连协议二者是独有的。

[0117] 在一些实现方式中，利用HPI的物理层和链路层的缓冲存储器协议可以是事务接口和协议。这可以允许将数据乱序返回到一系列接收到的请求(例如，读取请求)。缓冲存储器接口可用于在一些实现方式中将缓冲器芯片互连到CPU。在一个实现方式中，虽然一些协议跨缓冲存储器总线发送诸如激活、读取、写入和刷新之类的DRAM命令，但是在本文讨论的缓冲存储器协议的一些实现方式中，读取或写入命令可以简单地用地址发送。然后，缓冲器芯片可以解码特定的秩和组，并将请求分解为DRAM命令。也可以支持推测性读取和需求读取。

[0118] 转到图13，示出了简化框图1300，其图示出包括通过顺从HPI的链路互连的CPU设备1305、1310的计算系统的示例拓扑结构。每个CPU 1305、1310可以使用对应的缓冲存储器

协议链路(“MemLink”)同样地连接到一个或多个相应的缓冲器设备1315a-1。如上所述,在一些实现方式中,缓冲存储器协议互连可以是基于GPIO协议,因为缓冲存储器协议的物理层和链路层是基于GPIO协议(例如, HPI)的相同物理和链路层定义。尽管图13中未示出,但是CPU 1305、1310可以进一步使用GPIO协议连接到一个或多个下游设备。

[0119] 如图13的示例中进一步示出的,缓冲器设备1315a-1可以连接到诸如双列直插存储器模块(DIMM)设备之类的存储器设备。对应于每个缓冲器设备的存储器可以被视为对该缓冲器设备连接到的CPU(例如,1305、1301)来说是本地的。然而,其它设备(包括其它CPU)可以使用顺从GPIO协议的链路通过其它插槽访问存储器。在一些实现方式中,运行缓冲存储器协议的端口可以仅支持用于与存储器通信的命令,并且仅支持缓冲存储器协议(即,并非GPIO协议和缓冲存储器协议)。另外,在一些实现方式中,GPIO互连协议可以支持路由并且指示诸如请求节点标识符和目的地节点标识符这样的信息(例如,在其分组中)。另一方面,缓冲存储器协议可以是不利用路由的点对点接口。因此,在使用缓冲存储器接口发送的分组中,GPIO协议中使用的一些字段可能会被省略。替代地,可以指定供将地址解码信息主机携带到缓冲器使用的字段,以及其它示例。

[0120] 在其它实现方式中,缓冲器设备1315a-1可以支持两级存储器拓扑结构,其中一定量的快速存储器(例如,DRAM)用作针对更大、更慢的存储器(例如,非易失性存储器)的缓存。在一个这样的实现方式中,除了其它示例之外,缓冲器设备1315a-1中的一个或多个可以将DDR用作近的、快速存储器并且将事务DDR DIMM用作较大的“远”存储器。事务DIMM可以使用协议(例如,DDR-事务(DDR-T))来使用事务性命令向易失性存储器单列直插存储器模块(SIMM)通信。

[0121] 转向图14,呈现了简化框图1400,其示出了连接到近存储器模块和远存储器模块(例如,1405、1410)的缓冲器芯片1315的更详细的表示。如上述示例所示,缓冲器芯片1315可以通过使用定义的缓冲存储器协议的缓冲存储器协议互连链路1415通信地耦合到CPU设备或其它处理设备1305。缓冲器芯片1315可以包括执行附加功能的逻辑。例如,缓冲器芯片1315的实现可以包括存储器控制器1420、调度器1425、流控制逻辑1430、错误检测逻辑1435、以及目录状态管理逻辑1436。逻辑可以使用硬件电路、固件和/或软件来实现。

[0122] 在一些实现方式中,存储器控制器1420可以包括将请求转换成其存储器设备的存储器特定协议(例如,DDR4)的逻辑。存储器控制器1420可以从使用链路1415与缓冲器芯片1315通信的CPU 1305(或另一主机)的主机控制器逻辑1440中抽取这些存储器特定协议的细节。调度器1425可以包括对请求进行重排序和仲裁响应的逻辑以求达到更高的性能。缓冲器芯片1315可以附加地提供诸如流控制和错误检测之类的特征。在一些实现方式中,流控制逻辑1430可以在调度器1425中实施或以其它方式与调度器1425进行互操作,以确保缓冲器芯片1315的更高的效率和更高的性能。错误检测逻辑1435可以包括支持纠错码(ECC)检测方案的逻辑以及发起校正或处理检测到的错误的逻辑。目录状态管理逻辑1436可以维护存储器中每一行的目录状态。响应于一些事务,对应行的目录状态可以改变。目录状态管理逻辑1436可以更新识别存储器中每行的当前目录状态的记录。在一些情况下,目录状态管理逻辑1436可以从归属代理(例如,在主机1305处)接收数据以指示对存储器的行的目录状态的改变。在其它实例中,目录状态管理逻辑1436可以基于对应请求的性质改变包括逻辑来自动更新目录状态(即,没有主机来指示改变)。这可以节省带宽以及其它示例优点,因

为用于指示目录状态改变的至少一部分的带宽可以被保留，在目录状态管理逻辑1436允许直接在存储器控制器处(例如，1315处的1420)识别这些目录状态改变的一部分的情况下。

[0123] 在一些实现方式中，CPU 1305可以包括可以将CPU(或主机)地址转译为存储器地址的主机控制器逻辑1440，包括跨信道和插槽的排列以及其它功能性。主机1305可以包括可以允许多个进程在主机1305上并行执行的多个处理核1445a-d。另外，主机可以使用一个或多个缓存来缓存来自存储器(例如，1405、1410)的先前访问的行，以使得主机1305可以在不从缓冲器芯片1315重新请求数据的情况下重新访问这些行。这可以帮助解决缓冲存储器架构中通过缓冲器芯片1315引入的较高等待时间。缓存可以包括本地缓存和中间缓存，以及末级缓存(LLC)1450a-d。缓存(包括LLC)可以在多个核1445a-d之间被共享。实际上，各种缓存架构可以与采用一个或多个处理核的主机一起使用。

[0124] 如上所述，缓冲存储器协议的链路层可以是基于GPIO协议的链路层。例如，可以在缓冲存储器协议中使用(例如，在图10中描述和示出的)HPI协议的192位、3槽微片。缓冲存储器协议中的分组可以实施整个命令。分组可以被实现为一个或多个微片。图15A-15C可以表示缓冲存储器协议中的微片的实现。在一些实现方式中，读取数据可以经由带报头和无报头的存储器数据(MemData)分组的组合从缓冲器设备发送回主机。整个缓存行可以在三个无报头微片(而不是五个微片，如带报头的微片那样)中传送。因此，无报头分组可以在可能是接口的最受限制的部分的内容上提供更高的吞吐量。为了适应三个微片，无报头分组移除了一些字段，并对其值进行了假设。大多数所做的假设可能是真实的，但是所有读取返回，如果任何假设不是真的，那么将使用包含所有字段的带报头分组。在一些实现方式中，带报头的分组可以开始读取返回序列，以允许早期识别跟踪器标识符(例如，HTID，RTID等)，以用于将读取返回映射到事务缓冲存储器协议中的读取请求。

[0125] 在一些实现方式中，无报头分组可以包含表1中列出的信息：

表1：无报头分组字段

字段	Qty	描述
数据	512	64字节缓存行数据
目录	2	两个目录位，其存储有一般作为ECC位的一部分的数据
HTID	11	请求事务ID。由于早期HTID，分组包括用于下一读取数据分组的HTID。
CRC	48	每个微片16位的CRC
HIB	3	报头指示位。每个微片一个。此位对于无报头分组的每个微片一直为0。

[0126] 此外，图15A至15C图示出可用于在读取返回中传输整个缓存行的三个微片(例如，微片0(图15A处)、微片1(图15B处)以及微片2(图15C处))的序列。表2提供了在无报头分组中采用的三个微片的示例的总结：

表2：无报头分组

微片	内容
数据微片0	来自缓存行的前32字节的数据
数据微片1	前32字节的剩余部分加上第二32字节的第一部分
数据微片2	第二32字节的剩余部分

[0127] 可以设置HIB位以指示分组是带报头还是无报头。可以将HIB设置为仅对于跟随在报头微片之后的数据微片指示无报头的分组。这允许在分组中间交织某些其它分组。当交

织的分组完成时，原始带报头的分组的数据部分可以以HIB = 0在微片中发送。这种技术也可以用于通过插入空的微片来延迟数据分组。在一些实现方式中，当正常地将会需要报头时，无报头分组的开始由HIB位为0来进行信号通知。所有其它分组(包括空的微片)可包含已设置了HIB位的报头。

[0128] 如上所述，带报头的分组可以是总共五个微片长度，并且可以包括一个报头微片和4个数据微片。为了减少空闲等待时间，带报头的微片可以发送两次数据微片1。第一次只用来自前32字节的数据发送，被称为微片1a。这允许在数据的第二个32字节从存储器设备可用之前发送微片。结果是，关键的32字节块的数据在空闲情况下较早到达主机。微片1然后可以第二次被发送，这次完成。它将包含前32字节数据部分的重复加上第二个32字节部分。表3描述了一带报头分组的五个微片：

表3:带报头的分组

微片	内容
报头	分组报头
数据微片0	来自缓存行的前32字节的数据
数据微片1a	仅前32字节的剩余部分。第二个32字节部分被预留。其它微片有效载荷位(RTID[6:3])与数据微片1相同
数据微片1	完成微片1。前32字节部分被重发。也包括第二个32字节部分
数据微片2	第二个32字节的剩余部分。

在一些实现方式中，带报头的MemData分组可能不会被取消，并且将随后是数据微片。在知道也可以发送数据之前，不发送报头。在发送报头之前，可能未在缓冲器中累积完整的缓存行。

[0129] 在传统的系统中，各种主机上的归属代理可以被指派有在每次读取之后显式地更新存储在存储器中的对应行的目录状态的任务。明确地说，更新目录状态可以涉及到写回完整的缓存行加上目录状态。这样的实现可以导致用于更新目录状态的大量带宽，特别是在归属代理和存储器控制器不紧密耦合的缓冲存储器架构中。缓冲存储器链路可用于互连归属代理和存储器控制器。因此，该缓冲存储器链路可以由存储器控制器使用以接收主机的请求以及接收写回二者，其唯一目的在一些情况下可以是简单地指定基于读取的对给定行的目录状态的改变。此外，在目录状态在缓存行本身内指示的情况下，目录状态改变可以涉及到通过缓冲存储器链路将缓存行的整个副本从归属代理传输到存储器控制器，导致这种更新的高带宽成本。

[0130] 这些是其它问题，可以使用改进的架构来解决，该改进的架构利用新的命令类型以允许存储器控制器在没有来自归属代理或存储器控制器之外的其它逻辑的指导的情况下显式地或隐式地指导存储器控制器在可能的情况下更新目录状态。因此，可以节省归属代理和存储器控制器之间的接口上的显著的写入带宽。此接口可以是缓冲存储器链路接口。其它架构也可以利用这样的模型，诸如在与归属代理相同的片上系统(SoC)上实现存储器控制器的内部网格以及其它示例。

[0131] 图16A-16C是图示出在涉及主机1305和缓冲器芯片1315的缓冲存储器互连上的示例读取事务的简图1600a-c。尽管此示例特定地利用了读取事务，但是应当领会的是，图16A至16C的示例中所示的原理也可以应用于其它类型的请求(包括存储器无效请求)。回到图16A至图16C中的图示，存储器读取1610可以通过缓冲存储器互连从主机1305发送到缓冲器芯片1315。存储器读取或读取请求1610可以源自本地主机1305或来自(例如，经由HPI链路

以及其它示例)与本地主机1305连接的另一个远程主机。缓冲器芯片1315的存储器控制器可以接收读取请求1610并且确定对应的存储器读取操作1615以响应于该请求执行从存储器1605中检索对应的行。

[0132] 读取请求1610可以是根据缓冲存储器协议的请求。缓冲存储器协议读取请求1610可以对应于通用输入/输出(GPIO)互连协议的读取请求1650。GPIO协议也可以支持若干不同的请求类型,包括读取和存储器无效请求。在一个示例中,GPIO协议请求类型可以包括:

·RdCur:请求特定缓存行的不可缓存的“快照”副本

·RdCode:请求缓存行的共享(或“S”)副本

·RdData:请求缓存行的独占(或“E”)或S副本

·RdDataMig:请求缓存行的修改(或“M”)、E或S副本

·RdInvOwn:请求缓存行的M或E副本

·RdInv:从归属代理请求缓存行的E副本;在接收数据之前任何修改的副本都被提交给存储器

·InvItoE:请求E状态的缓存行的所有权,而无需数据响应

·InvXtoI:使所有缓存无效,包括请求代理的缓存

·InvItoM:请求M空状态的缓存行的所有权,无需数据响应,其中不久之后写回已修改行的意图

·NonSnpRd:请求来自存储器的缓存行的不可缓存的“快照”副本

如上述示例所示,可以为GPIO协议维护一组一致性状态,包括共享(“S”)、独占(“E”)、修改(“M”)、和无效(“I”)状态,以及其它潜在示例。

[0133] 缓冲存储器协议本身可以定义多个不同的请求类型。缓冲存储器协议还可以维护每个缓存行的目录状态,包括任何(“A”)(例如,M、E、S或I一致性状态中的任何一个)、共享(“S”)(例如,任何S或I一致性状态中的任何一个)、独占(“E”)、无效(“I”)(例如,保证处于I一致性状态)或未知,以及其它示例。在一个示例中,缓冲存储器协议的读取请求类型可以包括:

·MemRd:正常存储器读取。目录信息保留未修改

·MemRdXtoI:存储器读取,目录结果无效。如果目录指示除I之外的任何东西,则它被写回为I

·MemInvXtoI:存储器无效。目录结果无效。没有读取数据被返回。存储器控制器读取缓存行,并且如果它尚未将目录设置为I,则通过将目录设置为I来对其进行重写

·MemRdXtoS:存储器读取,目录结果是共享。如果目录指示S以外的任何东西,则它被写回为S

·MemRdData:存储器读取,使用如下目录进行重写(如有必要):I到A;S没有变化;A没有变化;如果要求从A进行改变,则主机必须专门对其进行改变

·MemRdXtoA:存储器读取,目录结果为任何。如果目录指示除A之外的任何内容,则将其写回为A

·MemInvXtoA:存储器无效,目录结果为任何。没有读取数据被返回。存储器控制器读取缓存行,并且如果它尚未将目录设置为A,则通过将目录设置为A来对其进行重写。

[0134] 缓冲存储器协议读取请求类型可以对应于GPIO协议读取请求类型。缓冲器或存储

器控制器有时可以仅覆盖缓存代理的子集，因此，归属逻辑可以负责根据请求的缓存代理是否被目录覆盖来发布正确的存储器命令。在一个实现方式中，缓存代理可以在本地和远程缓存代理之间分组(诸如在表4和5的示例中)，但是可以在其它实现方式中使用其它替代分组。在表4和表5的示例中，“本地”请求方是未被目录覆盖的代理，并且“远程”缓存代理被目录覆盖。表4和表5列出了GPIO协议读取请求类型和缓冲存储器(MEM)协议请求类型之间的关联。表4列出了来自“本地”主机或缓存代理的读取请求。表5列出了源自“远程”主机或缓存代理的读取请求。

[0135] 表4:对于本地缓存代理的读取请求的目录状态转变

GPIO请求类型	MEM请求类型	当前目录状态	新目录状态
RdCur、RdCode、RdData、RdDataMig	MemRd	I/S	-
		A	*
RdInvOwn	MemRdXtoI	I	-
		S/A	I
InvItoE、InvXtoI	MemInvXtoI	I/S/A	I
NonSnpRd	MemRd	I/S/A	-

[0136] 表5:对于远程缓存代理的读取请求的目录状态转变

GPIO请求类型	MEM请求类型	当前目录状态	新目录状态
RdCur	MemRd	I/S	-
		A	*
RdCode	MemRdXtoS	I	S
		S	-
		A	S
RdData、RdDataMig	MemRdData	I	A
		S	-
		A	*
RdInvOwn	MemRdXtoA	I/S	A
		A	-
InvItoE	MemInvXtoA	I/S	A
		A	-
InvXtoI	MemInvXtoI	I	-
		S/A	I
NonSnpRd	MemRd	I/S/A	-

[0137] 如图16A-16C所示，在一些实现方式中，缓存行1630可以包括被编码以识别缓存行1630的目录状态的字段或位(例如，1635)。响应于读取请求，缓存行1630的副本可以返回到请求主机(例如，1305)，该副本包括编码的目录状态位1635。转向图16B的示例，在一些情况下，读取事务可导致对对应缓存行的目录状态的改变。实际上，表4和表5列出了基于读取请求的类型和所请求的缓存行在读取请求时的目录状态可以发生的状态转变。此外，在一些实现方式中，目录状态转变还可以取决于读取请求是源自本地缓存代理还是远程缓存代理。例如，实现方式可以使用目录状态来首要地跟踪远程代理(例如，远程代理是否具有共享副本、独占副本等)。在一些实现方式中，归属代理只能知道目录覆盖的代理(例如，远程代理)或不被目录覆盖的代理(例如，本地代理)，这导致这些确定仅是针对远程代理或本地代理中的一个或另一个做出的。目录位可以反映远程插槽中的目录状态，并可用于确定是否应执行远程监听。在一些实现方式中，本地代理可以拥有缓存行的副本，并且来自本地代

理的读取请求通常可导致目录状态转变为无效(“I”)或“清洁”状态。

[0138] 作为示例,在图16B的图示中,读取请求1610可以导致副本1630由缓冲器芯片1315存储器控制器(在1620处)检索并(在1640处)(通过缓冲存储器链路)转发到主机1305。在此示例中,缓存行1630的目录状态在接收到请求1610时可以是“A”。目录状态可以被编码在目录状态位1635中。在接收到缓存行1630时,主机1305可以改变目录状态。在一些情况下,该改变可以是基于由主机的归属代理执行的监听的结果。例如,当请求是“RdCur”读取请求时,主机1305可以与请求1610相关联地监听远程缓存代理。在图16B的示例中,基于监听的结果,主机确定目录状态应该被更新为“S”(如位1635a所反映的那样)。为了允许将此目录更新传送到缓冲存储器,主机可以写回1650具有更新的目录状态位1635a的缓存行1630。缓冲器芯片的存储器控制器然后可以将具有更新的目录状态位1635的更新的缓存行写入1655到存储器。

[0139] 在一些情况下,缓冲器芯片1315的存储器控制器可以在改变未(例如,通过写回(例如,1650))传送回缓冲器芯片的情况下推断目录状态的改变。例如,如表4和5所示,读取请求类型和当前目录状态的一些组合将总是导致相同的目录改变(或没有改变)。在表4和表5中,列“新目录状态”显示当缓存行具有特定目录状态时(并且取决于请求是源于本地代理还是远程代理)特定类型的读取请求所导致的目录状态。作为示例,如表5所示,当读取请求来自远程代理并且是“RdData”请求同时目录状态处于“I”状态时,可以可靠地预测到目录状态到“A”状态的改变,以及包括表4和表5所示的那些的其它示例。

[0140] 在表4和5中,当“新目录状态”被指示为“-”时,可以预测将不会基于读取请求导致对目录状态的改变。当“新目录状态”被指示为“*”时,改变是可能的,但不可预测。例如,任何目录状态改变可取决于请求主机执行的监听的结果。作为示例,在表5中,当读取请求来自远程代理并且是“RdData”请求同时目录状态处于“A”状态时,可以由归属代理执行监听,导致对缓存行的目录状态的潜在不可预测的改变。因此,在由“*”指示的情况下,目录状态有时可以仅由存储器控制器通过来自归属代理的写回或其它显式通信来更新。在新的目录状态可预测的情况下,存储器控制器可以直接更新目录状态,而不需要本地代理的干预。

[0141] 为了例证,在图16C中,示出了当缓存行处于“A”状态(如目录状态位1635中所记录的那样)时接收到读取请求1610的示例。在此示例中,基于读取请求1610的类型和缓存行处于“A”状态(并且基于请求的来源),存储器控制器可以确定归属代理将会将目录状态更新为“I”,如图所示。存储器控制器可以基于读取请求的字段来识别读取请求的类型和请求的来源,并根据此信息得出目录状态将改为什么。存储器控制器可以通过将具有新的目录状态位(1635b)的缓存行的新副本(在1645处)写入到存储器(在1665处)来抢先改变存储器中的缓存行副本的目录状态位,以反映归属代理将(在1660处)对目录状态做出的改变。因此,缓冲器芯片1315的存储器控制器可以装配有用于确定是否可以根据当缓存行处于特定目录状态时来自特定缓存代理的特定读取请求类型来预测出目录状态改变的逻辑。存储器控制器可以同样地确定它是否应当等待并且依赖于来自归属代理的通信来识别对目录状态的不可预测的改变。类似地,归属代理可以包括这样的逻辑,其用于确定何时应该将更新的目录状态位写回(或以其它方式通信)以促进记录在存储器中的缓存行的目录状态的更新、或者归属代理是否可以基于分离的存储器控制器将直接执行这些更新的理解来放弃任何这样的通信。存储器控制器和归属代理二者可以包括或访问具有体现描述目录状态改变的

组合的信息(诸如在示例表4和5以及其它示例中描述的信息)的机器可读数据的数据结构。

[0142] 图17是根据一些示例的缓存行1630的简化表示1700。在此特定示例中,缓存行可以包括有效载荷数据1705以及纠错码(ECC)数据(例如,在ECC字段1710中)。在此示例中,目录状态位1635可以被包括在如存储在DRAM中的ECC位1710中或在ECC位1710中被推出。例如,在一个示例中,目录位可以是存储在64字节缓存行内的两个目录位。目录状态位1635可以由主机设备在写入时设置,由存储器控制器在一些读取事务中(例如,如图16C的示例中)直接更新,或者是基于在其它读取事务中(例如,如图16B的示例中)的由主机的通信。在一个示例中,可以根据下面的表6的示例来实现两位目录状态位实现:

表6:目录状态编码

Dir [1]	Dir [0]	状态	描述	远程插槽缓存	本地插槽缓存
0	0	I	无效状态。缓存行未被缓存在远程插槽缓存中。在本地插槽缓存中可能处于I、S或F。	未缓存	任何状态
0	1	-	未使用		
1	1	S	共享。缓存行以S或F状态进行缓存,或根本未被缓存	I、S或F	I、S或F
1	0	A	任何。缓存行在远程插槽中可处于任何缓存状态(M、E、F、S或I)。在本地插槽中也可以是任何状态	任何状态(M、E、F、S或I)	任何状态(M、E、F、S或I)

其它实现方式可以支持附加的或替代的目录状态。此外,其它实现方式可以使用附加的目录状态位。此外,虽然表4和表5列出了各种读取请求类型和由这些读取请求产生的预测出的目录状态改变,但是其它实现方式可以支持读取请求类型的替代集合以及基于读取请求类型、使用的目录状态以及读取请求的来源的对于目录状态的预测影响,以及其它替代和示例。

[0143] HPI和相关联的缓冲存储器访问协议可并入到任何种类的计算设备和系统中,包括大型机、服务器系统、个人计算机、移动计算机(诸如平板、智能电话、个人数字系统等)、智能电器、游戏或娱乐控制台和机顶盒,以及其它示例。例如,参考图18,其示出了根据本发明的实施例的第二系统1800的框图。如图18所示,多处理器系统1800是点对点互连系统,并且包括经由点对点互连1850耦合的第一处理器1870和第二处理器1880。处理器1870和1880中的每一个可以是某版本的处理器。在一个实施例中,1852和1854是串行点对点一致性互连结构(诸如高性能架构)的一部分。结果,本发明可以在QPI架构内实现。

[0144] 尽管示出有仅两个处理器1870、1880,但是应当理解,本发明的范围不限于此。在其它实施例中,一个或多个附加处理器可以存在于给定的处理器中。处理器1870和1880被示出为分别包括集成的存储器控制器单元1872和1882。处理器1870还包括点对点(P-P)接口1876和1878作为其总线控制器单元的一部分;类似地,第二处理器1880包括P-P接口1886和1888。处理器1870、1880可以使用P-P接口电路1878、1888经由点对点(P-P)接口1850来交换信息。如图18所示,IMC 1872和1882将处理器耦合到相应的存储器,即存储器1832和存储器1834,存储器1832和存储器1834可以是本地地附接于相应处理器的主存储器的一部分。

[0145] 处理器1870、1880各自通过使用点对点接口电路1876、1894、1886、1898经由单独的P-P接口1852、1854与芯片组1890交换信息。芯片组1890还沿着高性能图形互连1839经由接口电路1892与高性能图形电路1838交换信息。

[0146] 共享缓存(未示出)可以被包括在任一处理器中或两个处理器外部;但是通过P-P互连与处理器连接,以使得如果处理器被置于低功率模式,则任一个或两个处理器的本地缓存信息可以被存储在共享缓存中。

[0147] 芯片组1890可以经由接口1896耦合到第一总线1816。在一个实施例中,第一总线

1816可以是外围组件互连(PCI)总线或诸如快速PCI总线或另一第三代I/O互连总线之类的总线，但是本发明的范围不限于此。

[0148] 如图18所示，各种I/O设备1814连同总线桥1818一起耦合到第一总线1816，总线桥1818将第一总线1816耦合到第二总线1820。在一个实施例中，第二总线1820包括低引脚数(LPC)总线。各种设备耦合到第二总线1820，所述设备包括例如键盘和/或鼠标1822，通信设备1827和诸如盘驱动器或其它大容量存储设备之类的存储单元1828，其通常在一个实施例中包括指令/代码和数据1830。此外，音频I/O 1824被示出为耦合到第二总线1820。注意，其中所包括的组件和互连架构变化的其它架构是可能的。例如，代替图18的点对点架构，系统可以实现多点分支总线或其它此类架构。

[0149] 现在参考图19，示出了多核处理器的实施例的框图。如图19的实施例中所示，处理器1900包括多个域。具体地，核域1930包括多个核1930A-1930N，图形域1960包括具有媒体引擎1965的一个或多个图形引擎，以及系统代理域1910。

[0150] 在各种实施例中，系统代理域1910处理功率控制事件和功率管理，使得域1930和1960的各单元(例如核和/或图形引擎)是可独立控制的，以根据在给定单元中进行的活动(或不活动)在适当的功率模式/水平下(例如活动、超频(turbo)、睡眠、休眠、深度睡眠或其它类似于高级配置功率接口的状态)动态地操作。域1930和1960中的每一个可以在不同的电压和/或功率下操作，并且此外，域内的各单元各自潜在地在独立的频率和电压下操作。要指出，虽然仅示出有三个域，但应理解本发明的范围在这方面不受限制，并且在其它实施例中可以存在附加域。

[0151] 如所示，每个核1930除各种执行单元和附加处理元件之外还包括低级缓存。在这里，各种核被相互耦合并耦合到共享缓存存储器，其由末级缓存(LLC)的多个单元或切片1940A-1940N形成；这些LLC常常包括存储和缓存控制器功能性，并且在核之间以及潜在地也在图形引擎之间被共享。

[0152] 如看到的，环形互连1950将核耦合在一起，并且经由多个环间断(stop)1952A-1952N(每个在核与LLC切片之间的耦合处)提供核域1930、图形域1960 和系统代理电路1910之间的互连。如在图19中看到的，互连1950被用来载送各种信息，包括地址信息、数据信息、应答信息以及监听/无效信息。虽然图示出环形互连，但可以利用任何已知的管芯上互连或结构。作为例证性示例，可以以类似方式利用上文所讨论的结构中的某些(例如，另一管芯上互连、英特尔片上系统结构(IOSF)、高级微控制器总线架构(AMBA)互连、多维网状结构或其它已知互连架构)。

[0153] 如进一步描绘的，系统代理域1910包括显示引擎1912，其将提供关联显示器的控制和到关联显示器的接口。系统代理域1910可以包括其它单元，诸如：集成存储器控制器1920，其提供到系统存储器(例如，用多个DIMM实现的DRAM)的接口；用以执行存储器一致性操作的一致性逻辑1922。可以存在多个接口以使得能够实现处理器与其它电路之间的互连。例如，在一个实施例中，提供了至少一个直接媒体接口(DMI)1916接口以及一个或多个PCIe™接口1914。显示引擎和这些接口通常经由PCIe™桥接器1918耦合到存储器。此外，为了提供其它代理(诸如附加处理器或其它电路)之间的通信，可以提供一个或多个其它接口。

[0154] 虽然已经相对于有限数量的实施例描述了本发明，但是本领域技术人员将认识到

源自于其的许多修改和变化。所附权利要求旨在涵盖落入本发明的真实精神和范围内的所有这些修改和变化。

[0155] 设计可以经历从创建到模拟到制造的各个阶段。代表设计的数据可以以多种方式表示设计。首先，如在仿真中有用的，硬件可以使用硬件描述语言(HDL)或另一功能性描述语言来表示。此外，可以在设计过程的某一阶段产生具有逻辑和/或晶体管门的电路级模型。此外，大多数设计在某一阶段达到表示硬件模型中各种设备的物理放置的数据级别。在使用常规半导体制造技术的情况下，表示硬件模型的数据可以是指定用于产生集成电路的掩模的不同掩模层上存在或不存在各种特征的数据。在一些实现方式中，这样的数据可以以诸如图形数据系统II(GDS II)、开放艺术品系统互换标准(OASIS)或类似格式的数据库文件格式存储。

[0156] 在一些实现方式中，基于软件的硬件模型以及HDL和其它功能性描述语言对象可以包括寄存器传送语言(RTL)文件，以及其它示例。这样的对象可以是机器可解析的，使得设计工具可以接受HDL对象(或模型)，针对所描述的硬件的属性解析HDL对象，并且从对象确定物理电路和/或片上布局。设计工具的输出可用于制造物理设备。例如，设计工具可以确定来自HDL对象的各种硬件和/或固件元件的配置，诸如总线宽度、寄存器(包括大小和类型)、存储器块、物理链路路径、结构拓扑、以及将被实施从而实现HDL对象中建模的系统的其它属性。设计工具可以包括用于确定片上系统(SoC)和其它硬件设备的拓扑和结构配置的工具。在一些实例中，HDL对象可以用作开发可以由制造设备用于制造所描述的硬件的模型和设计文件的基础。实际上，HDL对象本身可以被提供作为制造系统软件的输入以促成所描述的硬件。

[0157] 在设计的任何表示中，数据可被存储于任何形式的机器可读介质中。存储器或磁存储或光存储，诸如磁盘，可以是机器可读介质，以存储经由光或电波传输的信息，所述光或电波被调制或以其它方式生成为传输这种信息。当指示或携带代码或设计的电载波被传输时，在电信号的复制、缓冲或重传输被执行的程度上做出新的副本。因此，通信供应商或网络供应商可在有形的、机器可读介质上至少暂时存储诸如编码到载波中的信息的制品，从而实施本公开的实施例的技术。

[0158] 如本文所使用的模块指代硬件、软件和/或固件的任何组合。例如，模块包括硬件，诸如微控制器，其关联于非暂时性介质，以存储被适配成由微控制器执行的代码，因此，在一个实施例中，对模块的引用指代硬件，其特别地被配置为识别和/或执行将保持在非暂时性介质上的代码。此外，在另一实施例中，模块的使用指代包括代码的非暂时性介质，其特别地被适配成由微控制器执行，以实行预定操作。并且如可以推断的，在又一实施例中，术语模块(在该示例中)可指代微控制器和非暂时性介质的组合。通常，说明为分离的模块边界通常发生变化并且潜在地重叠。例如，第一和第二模块可共享硬件、软件、固件或其组合，同时潜在地保留一些独立硬件、软件或固件。在一个实施例中，术语逻辑的使用包括硬件，诸如晶体管、寄存器或其它硬件，诸如可编程逻辑器件。

[0159] 在一个实施例中，短语“配置为”的使用指代布置、放一起、制造，许诺销售、进口和/或设计装置、硬件、逻辑或元件，以执行指定的或确定的任务。在该示例中，如果被设计、耦合和/或互连以执行所述指定的任务，则没有正在操作的装置或其元件仍“配置为”执行指定的任务。仅作为说明性的示例，逻辑门可在操作期间提供0或1。但逻辑门“配置为”提供

启用信号至时钟，不包括可提供1或0的每个潜在的逻辑门。相反，逻辑门是以在操作期间1或0输出是用于启用时钟的某种方式耦合的逻辑门。再次注意，术语“配置为”的使用不需要操作，但相反集中于装置、硬件和/或元件的潜在状态，其中装置、硬件和/或元件的潜在状态被设计为当装置、硬件和/或元件正操作时执行特定任务。

[0160] 此外，在一个实施例中，短语“用以”、“能够”和/或“可操作以”的使用指代以使能以指定方式使用装置、逻辑、硬件和/或元件的这种方式设计的一些装置、逻辑、硬件和/或元件。如以上注意的，在一个实施例中，用以、能够、或可操作以的使用，指代装置、逻辑、硬件和/或元件的潜在状态，其中装置、逻辑、硬件和/或元件没有正在操作但以使能以指定方式使用装置的这种方式来被设计。

[0161] 如本文所使用的值，包括数字、状态、逻辑状态或二进制逻辑状态的任何已知表示。通常，逻辑电平、逻辑值或逻辑上的值的使用也称为1和0，其仅表示二进制逻辑状态。例如，1指代高逻辑电平而0指代低逻辑电平。在一个实施例中，存储单元，诸如晶体管或闪存单元，可以能够保持单个逻辑值或多个逻辑值。然而，使用了计算机系统中的值的其它表示。例如十进制数10还可被表示为1010的二进制值和十六进制的字母A。因此，值包括能够被保持在计算机系统中的信息的任何表示。

[0162] 此外，状态可由值或值的部分来表示。例如，第一值，诸如逻辑1，可表示默认或初始状态，而第二值，诸如逻辑0，可表示非默认状态。此外，在一个实施例中，术语重置和设置，分别指代默认和更新的值或状态。例如，默认值潜在地包括高逻辑值，即重置，而更新值潜在地包括低逻辑值，即设置。注意，值的任何组合可被用于表示任何数量的状态。

[0163] 上文阐述的方法、硬件、软件、固件或代码的实施例可经由可由处理元件执行的存储于机器可访问的、机器可读的、计算机可访问的或计算机可读的介质上的指令或代码来实现。非暂时性机器可访问/可读介质包括提供(即，存储和/或传输)以诸如计算机或电子系统的机器可读的形式的信息的任何机构。例如，非暂时性机器可访问介质包括随机存取存储器(RAM)，诸如静态RAM(SRAM)或动态RAM(DRAM)；ROM；磁或光存储介质；闪存设备；电存储设备；光存储设备；声学存储设备；用于保持接收自暂时性(传播)信号(例如，载波、红外信号、数字信号)的信息的其它形式的存储设备；等等，其将区别于可从其中接收信息的非暂时性介质。

[0164] 用于将逻辑编程以执行本发明的实施例的指令可存储于系统的存储器内，存储器诸如是DRAM、缓存、闪存或其它存储装置。此外，指令可经由网络或借助于其它计算机可读介质来分发。因此机器可读介质可包括用于以由机器(例如，计算机)可读的形式存储或传输信息的任何机构，但不限于，软磁盘、光盘、紧凑盘只读存储器(CD-ROM)、以及磁光盘、只读存储器(ROM)、随机存取存储器(RAM)、可擦可编程只读存储器(EPROM)、电可擦可编程只读存储器(EEPROM)、磁卡或光学卡、闪存、或用于经由电、光、声学或其它形式的传播信号(例如，载波、红外信号、数字信号等)通过互联网传输信息的有形的、机器可读存储装置。因此，计算机可读介质包括任何类型的有形的机器可读介质，其适用于以由机器(例如，计算机)可读的形式存储或传输电子指令或信息。

[0165] 以下示例属于根据本说明书的实施例。一个或多个实施例可以提供用于以下各项的装置、系统、机器可读存储、机器可读介质、基于硬件和/或基于软件的逻辑、以及方法：通过链路接收请求，该请求请求存储器中的特定行；识别存储器中的目录状态记录，该目录状

态记录识别所述特定行的目录状态；识别所述请求的类型；基于所述特定行的目录状态和所述请求的类型，确定所述特定行的目录状态要从所述特定状态改为新状态；以及响应于所述请求的接收，改变所述目录状态记录以反映所述新状态。可以响应于所述请求而发送所述特定行的副本。

[0166] 在一个示例中，该方法至少部分地由存储器控制器执行。

[0167] 在一个示例中，要从与存储器控制器分离的主机设备接收请求。

[0168] 在一个示例中，将通过缓冲存储器访问链路从主机设备接收请求。

[0169] 在一个示例中，存储器控制器被提供在与主机设备分离的缓冲器芯片上。

[0170] 在一个示例中，请求源自特定的缓存代理，并且所述改变将根据所述特定缓存代理的类型来确定。

[0171] 在一个示例中，特定缓存代理的类型是本地缓存代理或远程缓存代理中的一个。

[0172] 在一个示例中，请求包括读取请求和存储器无效请求中的一个。

[0173] 在一个示例中，目录状态记录包括两个或多个位，并且所述位将被包括在所述行中并反映所述特定状态。

[0174] 在一个示例中，存储器控制器直接将新状态写入到存储器中的特定行。

[0175] 在一个示例中，归属代理将目录状态更新为新状态，并且存储器控制器独立于归属代理地将目录状态更新为新状态。

[0176] 在一个示例中，所述位被包括在特定行的纠错码位中。

[0177] 在一个示例中，存储器控制器确定与另一请求相关联的目录状态改变是不可预测的。

[0178] 在一个示例中，存储器控制器还用于接收来自归属代理的通信，该通信指示对与所述另一请求相对应的行的目录状态改变，并基于该通信来执行对与所述另一请求相对应的行的写入以更新与所述另一请求相对应的行的目录状态。

[0179] 在一个示例中，所述通信包括与所述另一请求相对应的行的写回，其中经写回的行指示与所述另一请求相对应的行的目录状态。

[0180] 在一个示例中，与所述另一请求相对应的行的目录状态包括与所述另一请求相对应的行的目录状态记录。

[0181] 在一个示例中，目录状态是一组目录状态中的一个，并且该组目录状态包括共享状态、任何状态和无效状态。

[0182] 一个或多个实施例可以提供用于以下各项的装置、系统、机器可读存储、机器可读介质、方法以及基于硬件和/或基于软件的逻辑(例如，实现缓存代理逻辑)：确定对应于存储器的特定行的并且具有多个请求类型中的特定请求类型的缓冲存储器请求；通过缓冲存储器链路将所述请求发送到缓冲器芯片；响应于所述请求接收存储器的所述特定行的副本；根据所述特定行中包括的目录状态记录来确定所述特定行处于特定目录状态中；确定对所述目录状态的改变；以及基于所述特定目录状态和所述请求的特定请求类型来确定是否将所述改变传送到所述缓冲器芯片。

[0183] 在一个示例中，如果可以根据特定目录状态和请求的特定请求类型来预测该改变，则不将改变传送到缓冲器芯片。

[0184] 在一个示例中，如果不可由缓冲器芯片根据特定目录状态和请求的特定请求类型

来预测到该改变，则将改变传送到缓冲器芯片。

[0185] 一个或多个实施例可以提供一种系统，包括：缓冲器芯片；通过缓冲器芯片访问的存储器；和通过存储器访问链路耦合到缓冲器芯片的处理器块。缓冲器芯片包括存储器控制器，用于：通过存储器访问链路从处理器块接收与存储器中的特定行相关的请求；识别存储器中的目录状态记录，所述目录状态记录识别所述特定行的目录状态；根据请求识别该请求的类型；基于特定行的目录状态以及请求的类型来确定特定行的目录状态要从特定状态改为新状态；以及响应于请求的接收，改变目录状态记录以反映新状态。

[0186] 在一个示例中，处理器块通过通用输入/输出(GPIO)互连链路与一个或多个其它设备对接，存储器访问链路不同于GPIO互连链路，并且存储器访问链路的物理层是基于GPIO互连链路的物理层。

[0187] 遍及本说明书对“一个实施例”或“实施例”的引用意指结合所述实施例描述的特定特征、结构或特性被包括于本发明的至少一个实施例中。因此，遍及本说明书在各个地方出现短语“在一个实施例中”或“在实施例中”不一定全部指代相同的实施例。此外，特定特征、结构或特性可以在一个或多个实施例中以任何合适的方式被组合。

[0188] 在前述说明书中，已参考特定示例性实施例给出详细描述。然而，将显而易见的是，可对其进行各种修改和改变而不脱离如所附权利要求中阐述的本发明的更广泛的精神和范围。因此，应以例证性意义而非限制性意义来看待说明书和附图。此外，实施例的前述用途和其它示例性语言不一定指代相同实施例或相同示例，而是可指代不同的和有区别的实施例以及潜在地指代相同的实施例。

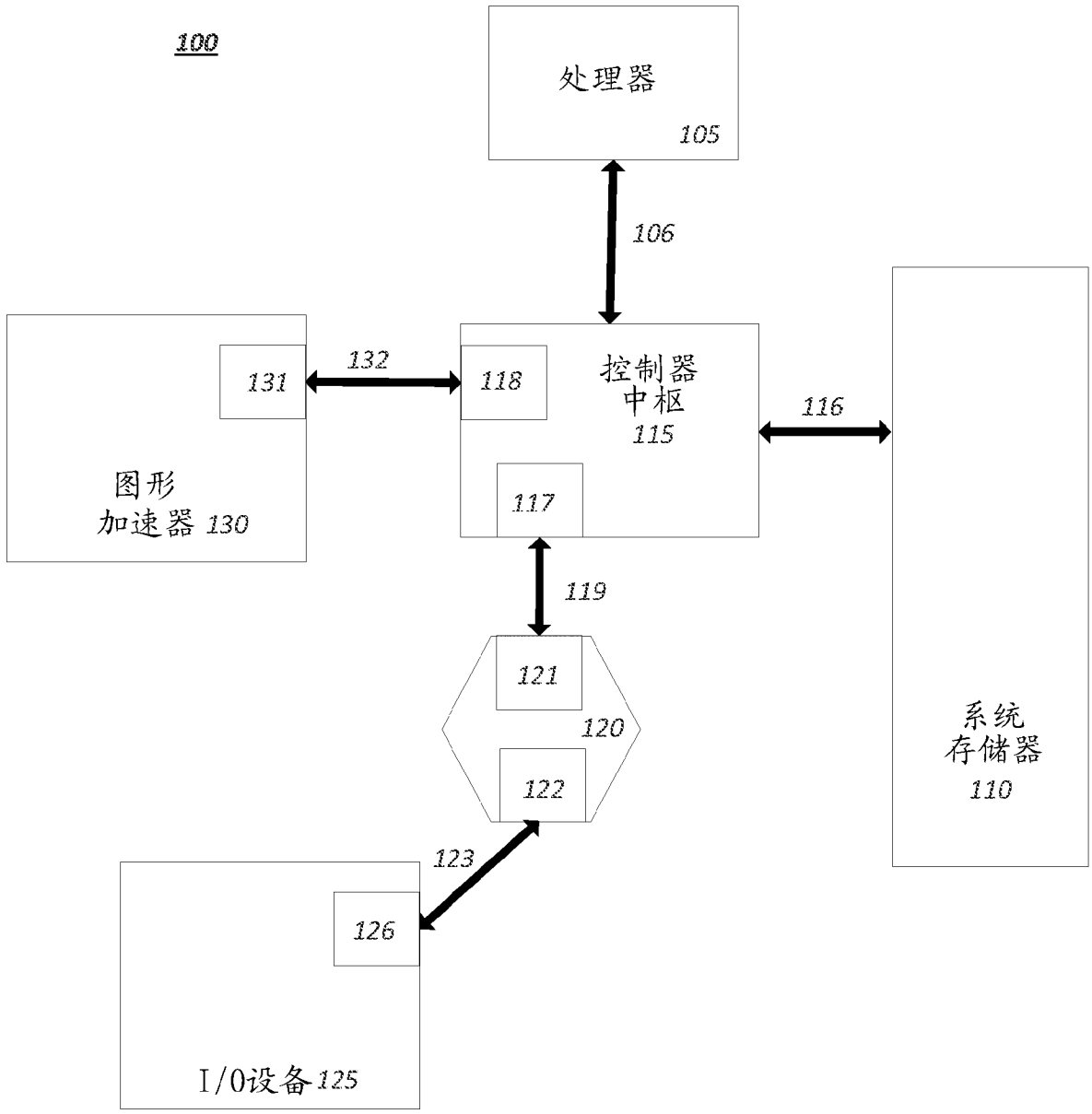

100
处理器
105
106
131
132
118
控制器
中枢
115
116
图形
加速器 130
117
119
121
120
122
系统
存储器
110
123
126
I/O设备 125

图 1

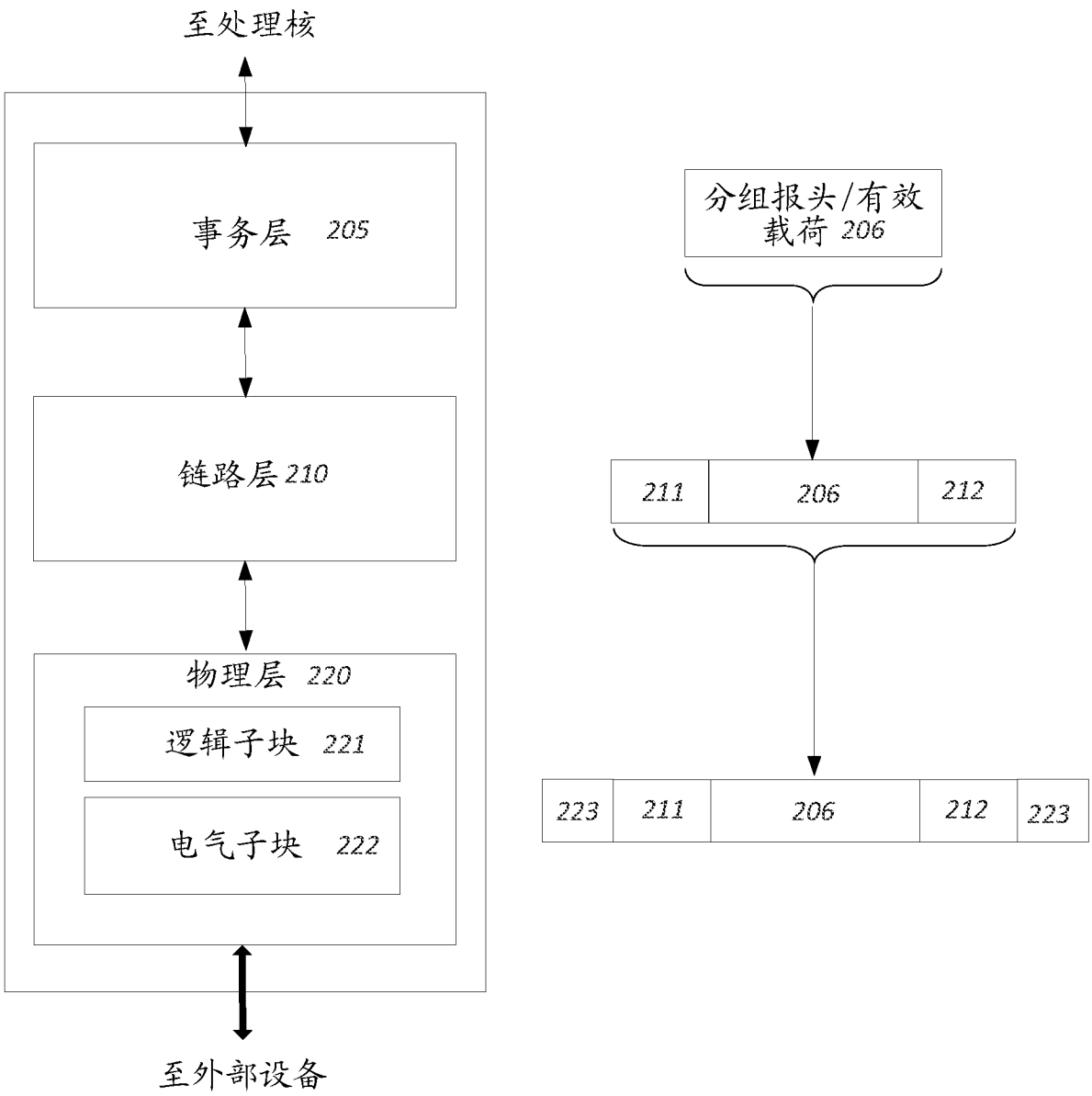

分层协议栈200
至处理核
事务层 205
链路层 210
物理层 220
逻辑子块 221
电气子块 222
至外部设备
分组报头/有效载荷 206
211
206
212
223
211
206
212
223

图 2

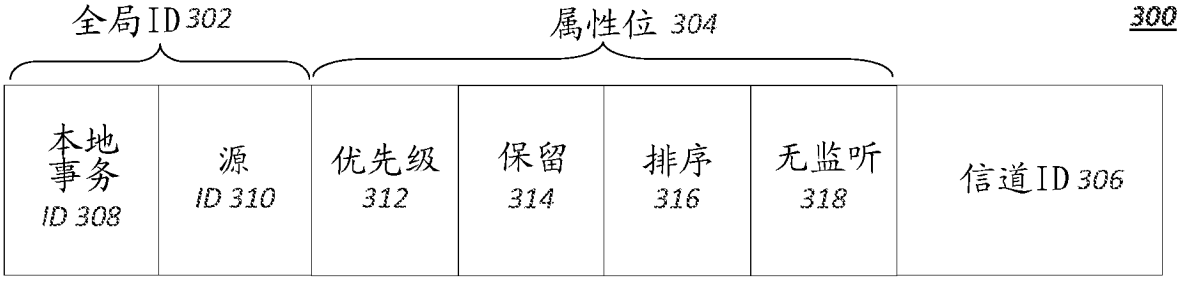

全局ID 302
属性位 304
300
本地事务 ID 308
源 ID 310
优先级 312
保留 314
排序 316
无监听 318
信道ID 306

图 3

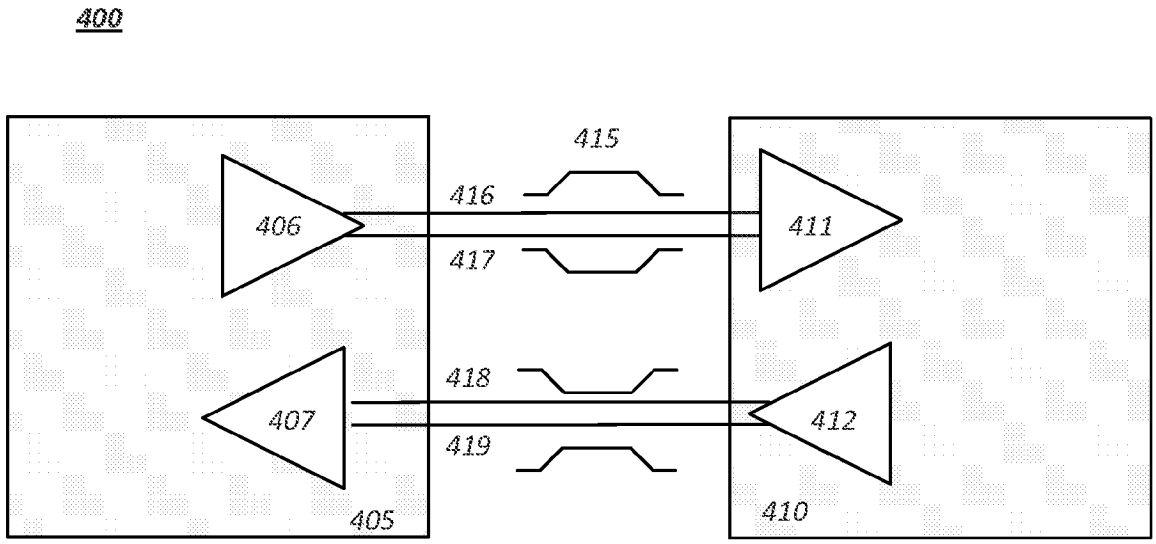


图 4

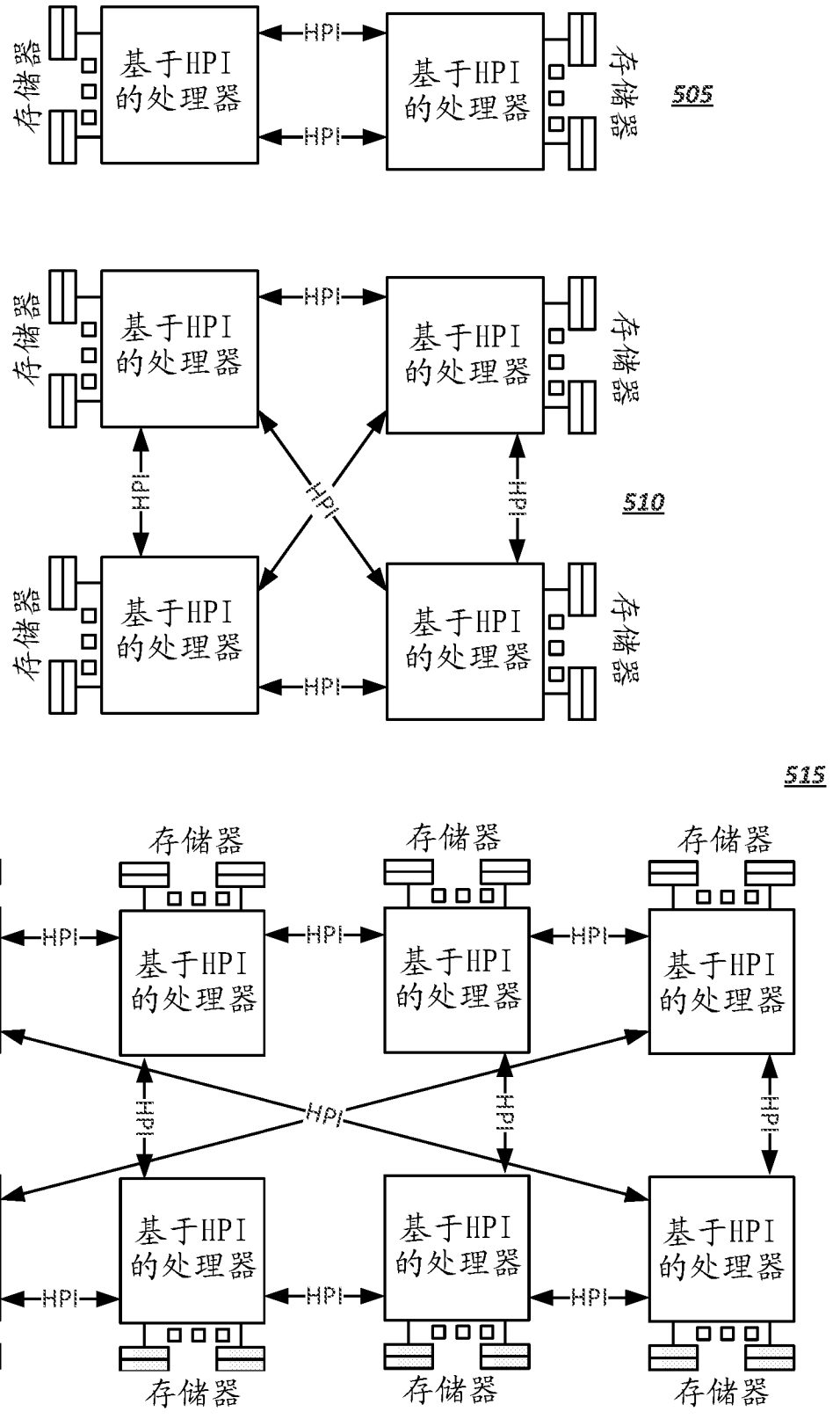


图 5

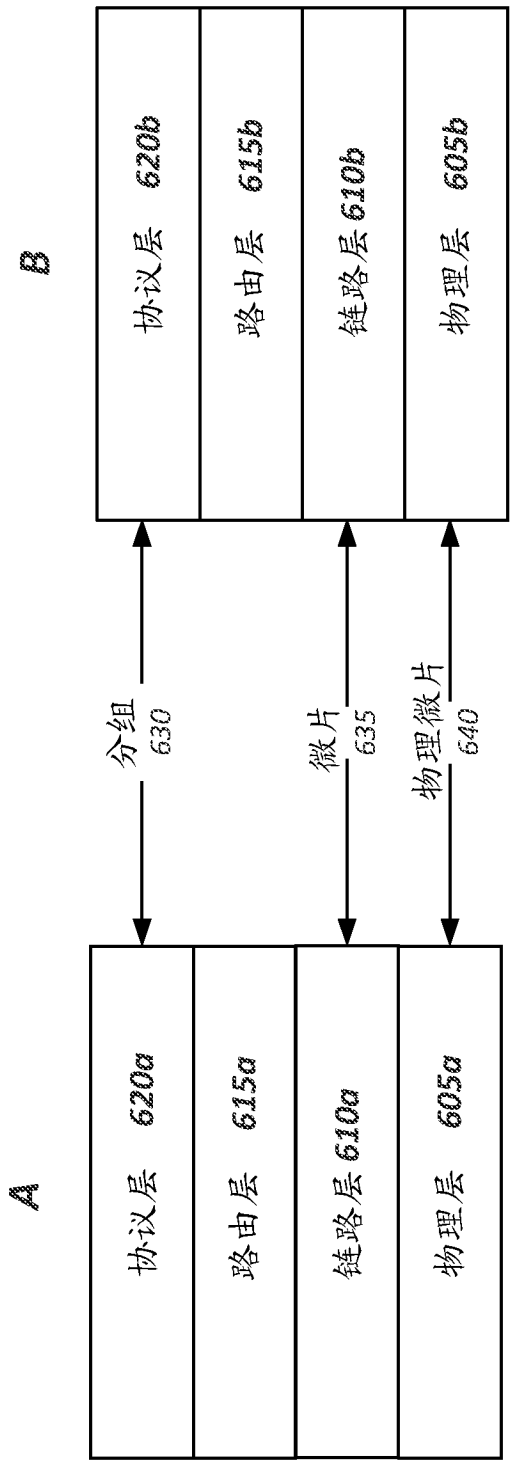

B
协议层 620b
路由层 615b
链路层 610b
物理层 605b
分组 630
微片 635
物理微片 640
A
协议层 620a
路由层 615a
链路层 610a
物理层 605a

图 6

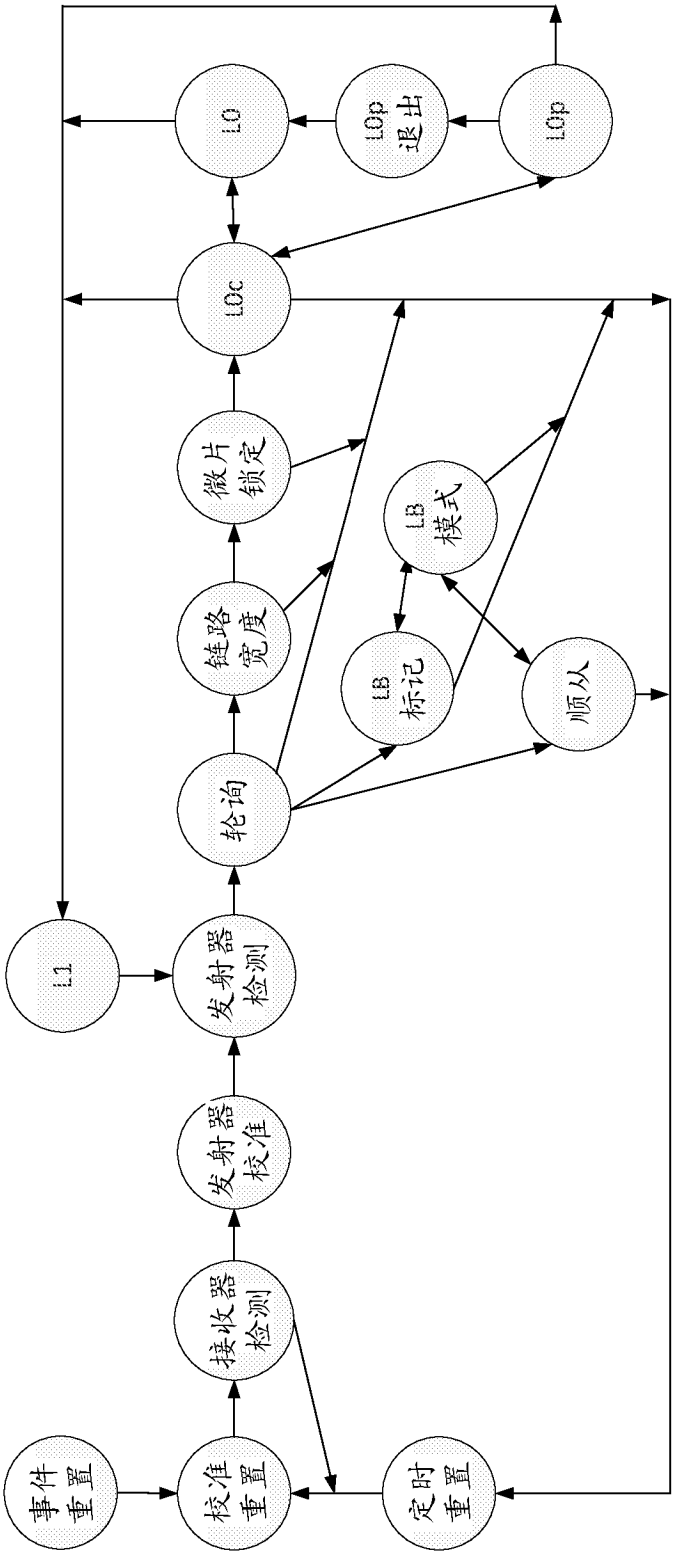
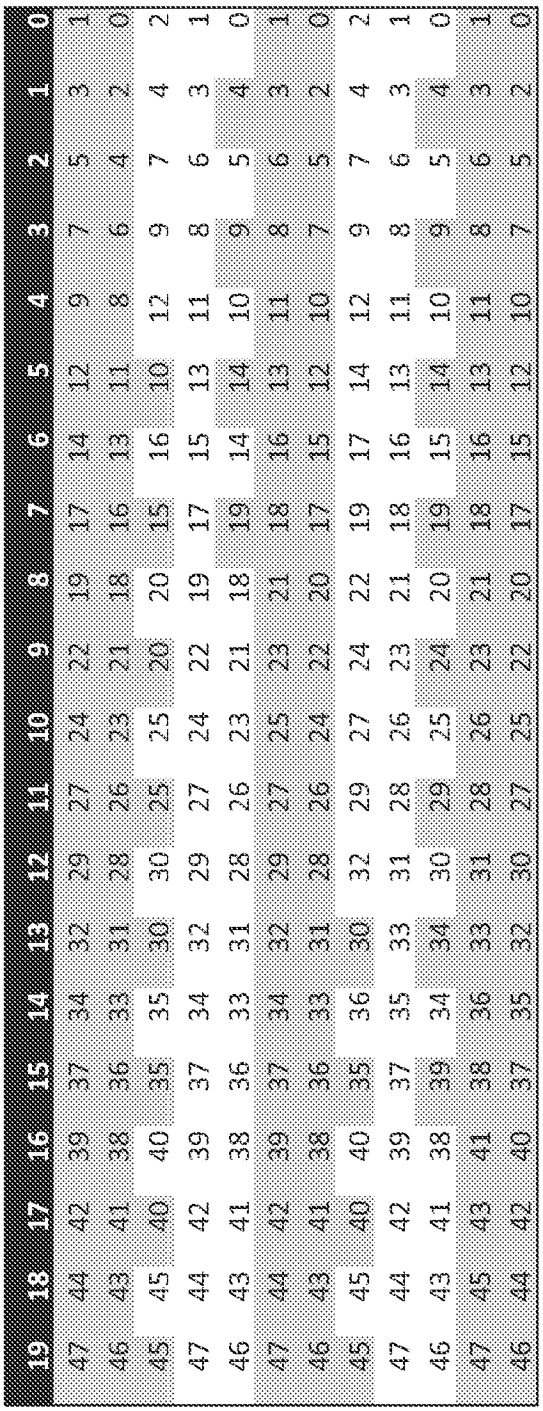

700
L0
L0p 退出
L0p
L0c
微片锁定
链路宽度
轮询
L1
发射器检测
发射器校准
接收器检测
事件重置
校准重置
定时重置
LB 模式
LB 标记
顺从

图 7



Ui	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0, 1, 2, 3	47	44	42	39	37	34	32	29	27	24	22	19	17	14	12	9	7	5	3	1
4, 5, 6, 7	46	43	41	38	36	33	31	28	26	23	21	18	16	13	11	8	6	4	2	0
8, 9, 10, 11	45	45	40	40	35	35	30	30	25	25	20	20	15	16	10	12	9	7	4	2
12, 13, 14, 15	47	44	42	39	37	34	32	29	27	24	22	19	17	15	13	11	8	6	3	1
16, 17, 18, 19	46	43	41	38	36	33	31	28	26	23	21	18	19	14	14	10	9	5	4	0
20, 21, 22, 23	47	44	42	39	37	34	32	29	27	25	23	21	18	16	13	11	8	6	3	1
24, 25, 26, 27	46	43	41	38	36	33	31	28	26	24	22	20	17	15	12	10	7	5	2	0
28, 29, 30, 31	45	45	40	40	35	36	30	32	29	27	24	22	19	17	14	12	9	7	4	2
32, 33, 34, 35	47	44	42	39	37	35	33	31	28	26	23	21	18	16	13	11	8	6	3	1
36, 37, 38, 39	46	43	41	38	39	34	34	30	29	25	24	20	19	15	14	10	9	5	4	0
40, 41, 42, 43	47	45	43	41	38	36	33	31	28	26	23	21	18	16	13	11	8	6	3	1
44, 45, 46, 47	46	44	42	40	37	35	32	30	27	25	22	20	17	15	12	10	7	5	2	0

800

图 8

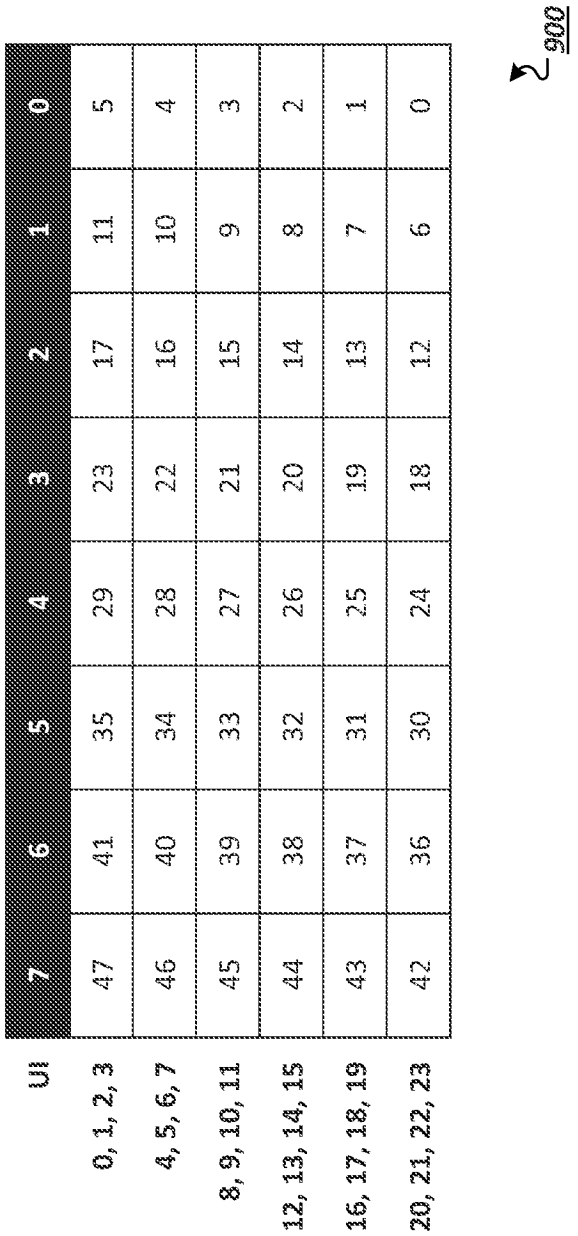

900
UI
7
6
5
4
3
2
1
0
0, 1, 2, 3
47 41 35 29 23 17 11 5
4, 5, 6, 7
46 40 34 28 22 16 10 4
8, 9, 10, 11
45 39 33 27 21 15 9 3
12, 13, 14, 15
44 38 32 26 20 14 8 2
16, 17, 18, 19
43 37 31 25 19 13 7 1
20, 21, 22, 23
42 36 30 24 18 12 6 0

图 9

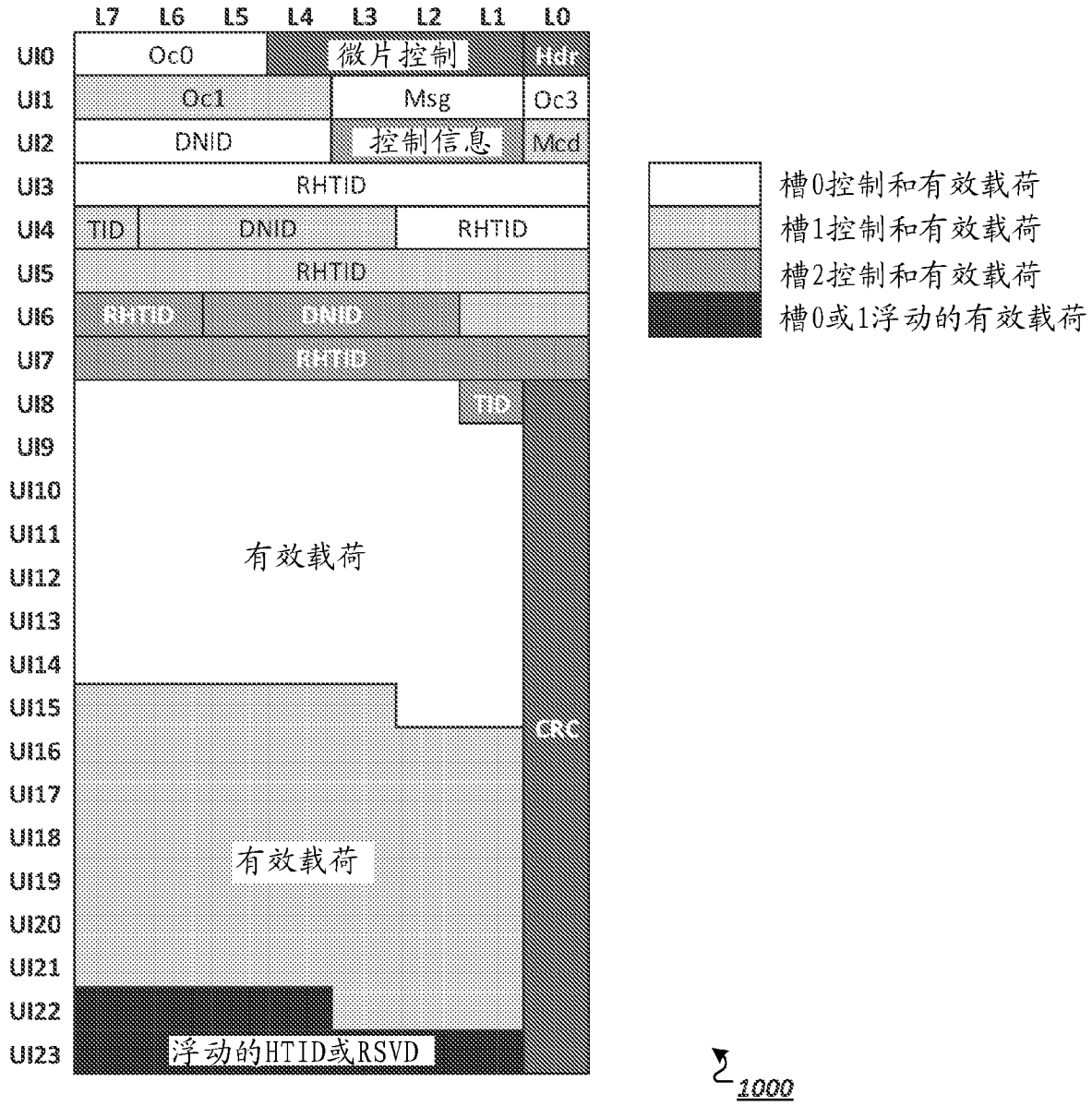

L7 L6 L5 L4 L3 L2 L1 L0
UI0
UI1
UI2
UI3
UI4
UI5
UI6
UI7
UI8
UI9
UI10
UI11
UI12
UI13
UI14
UI15
UI16
UI17
UI18
UI19
UI20
UI21
UI22
UI23
Oc0
微片控制
Hdr
Oc1
Msg
Oc3
DNID
控制信息
Mcd
RHTID
TID
DNID
RHTID
RHTID
DNID
RHTID
TID
有效载荷
CRC
有效载荷
浮动的HTID或RSVD
槽0控制和有效载荷
槽1控制和有效载荷
槽2控制和有效载荷
槽0或1浮动的有效载荷
1000

图 10

1100

	L7	L6	L5	L4	L3	L2	L1	L0
UI0	191	167	143	119	95	71	47	23
UI1	190	166	142	118	94	70	46	22
UI2	189	165	141	117	93	69	45	21
UI3	188	164	140	116	92	68	44	20
UI4	187	163	139	115	91	67	43	19
UI5	186	162	138	114	90	66	42	18
UI6	185	161	137	113	89	65	41	17
UI7	184	160	136	112	88	64	40	16
UI8	183	159	135	111	87	63	39	15
UI9	182	158	134	110	86	62	38	14
UI10	181	157	133	109	85	61	37	13
UI11	180	156	132	108	84	60	36	12
UI12	179	155	131	107	83	59	35	11
UI13	178	154	130	106	82	58	34	10
UI14	177	153	129	105	81	57	33	9
UI15	176	152	128	104	80	56	32	8
UI16	175	151	127	103	79	55	31	7
UI17	174	150	126	102	78	54	30	6
UI18	173	149	125	101	77	53	29	5
UI19	172	148	124	100	76	52	28	4
UI20	171	147	123	99	75	51	27	3
UI21	170	146	122	98	74	50	26	2
UI22	169	145	121	97	73	49	25	1
UI23	168	144	120	96	72	48	24	0

图 11

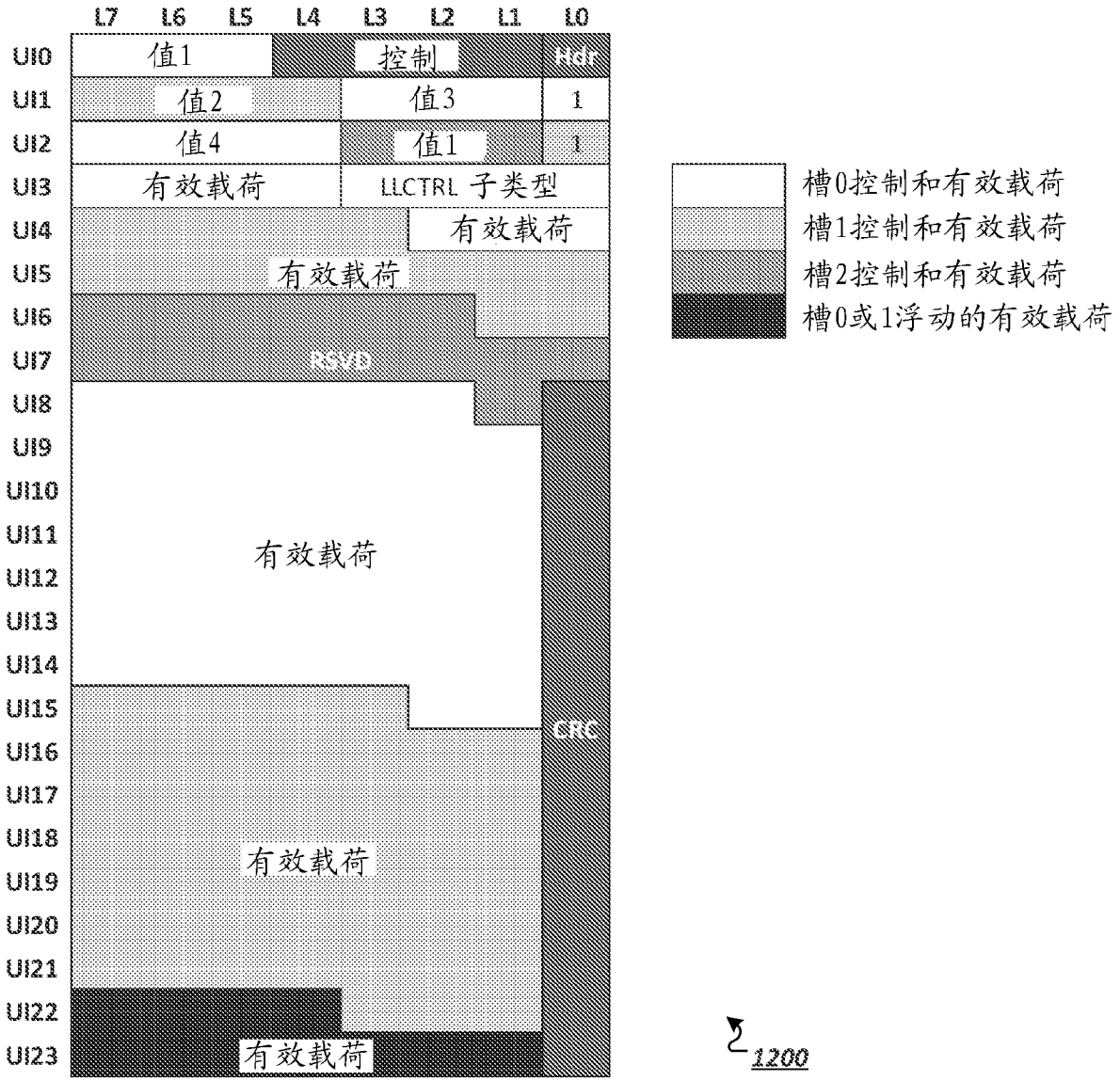

L7
L6
L5
L4
L3
L2
L1
L0
UI0
UI1
UI2
UI3
UI4
UI5
UI6
UI7
UI8
UI9
UI10
UI11
UI12
UI13
UI14
UI15
UI16
UI17
UI18
UI19
UI20
UI21
UI22
UI23
值1
控制
Hdr
值2
值3
1
值4
值1
1
有效载荷
LLCTRL 子类型
有效载荷
有效载荷
RSVD
有效载荷
CRC
有效载荷
有效载荷
槽0控制和有效载荷
槽1控制和有效载荷
槽2控制和有效载荷
槽0或1浮动的有效载荷
1200

图 12

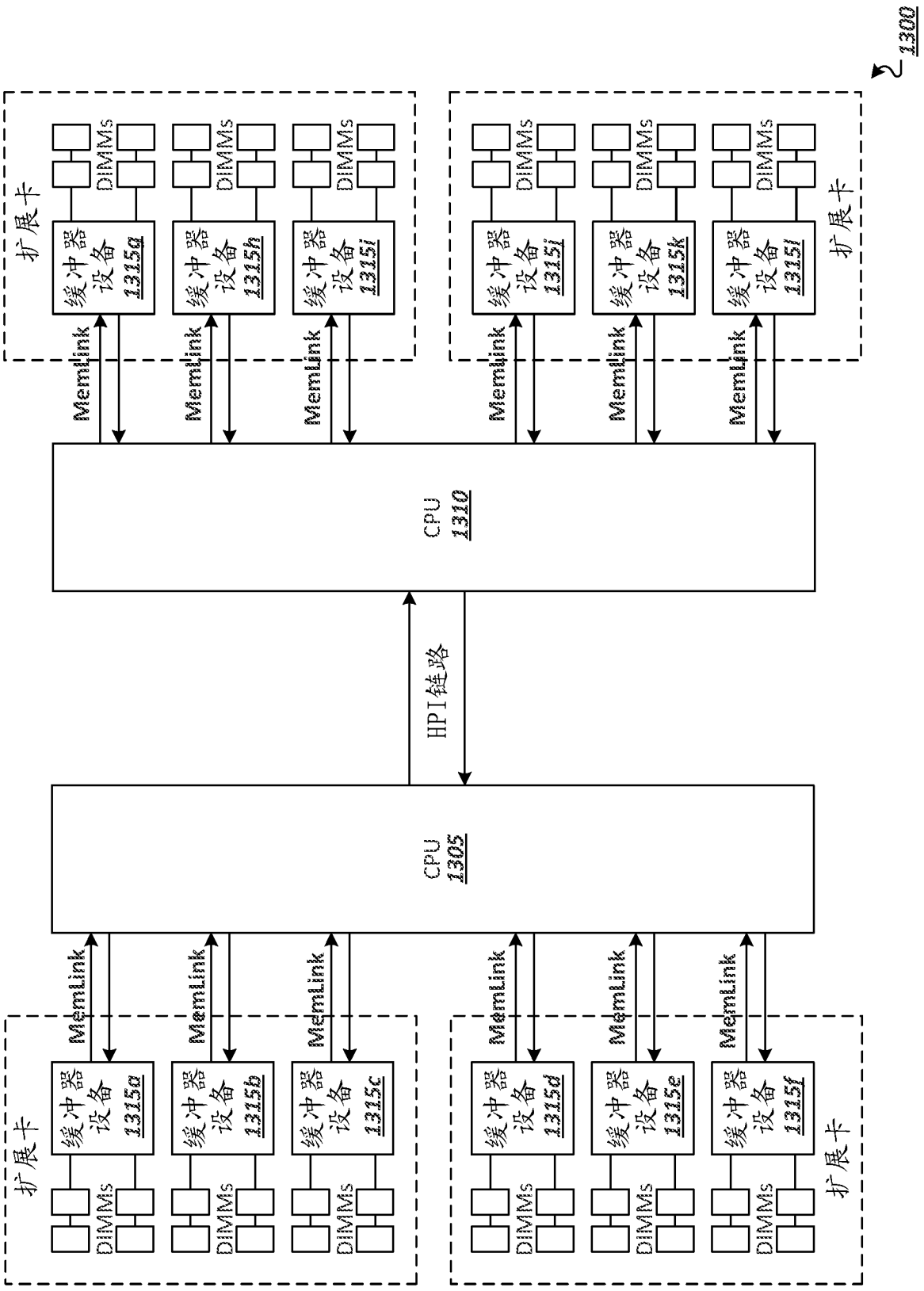

1300
扩展卡
缓冲器设备 1315a
缓冲器设备 1315b
缓冲器设备 1315c
缓冲器设备 1315d
缓冲器设备 1315e
缓冲器设备 1315f
缓冲器设备 1315g
缓冲器设备 1315h
缓冲器设备 1315i
缓冲器设备 1315j
缓冲器设备 1315k
缓冲器设备 1315l
DIMMs
MemLink
CPU 1310
CPU 1305
HPI链路

图 13

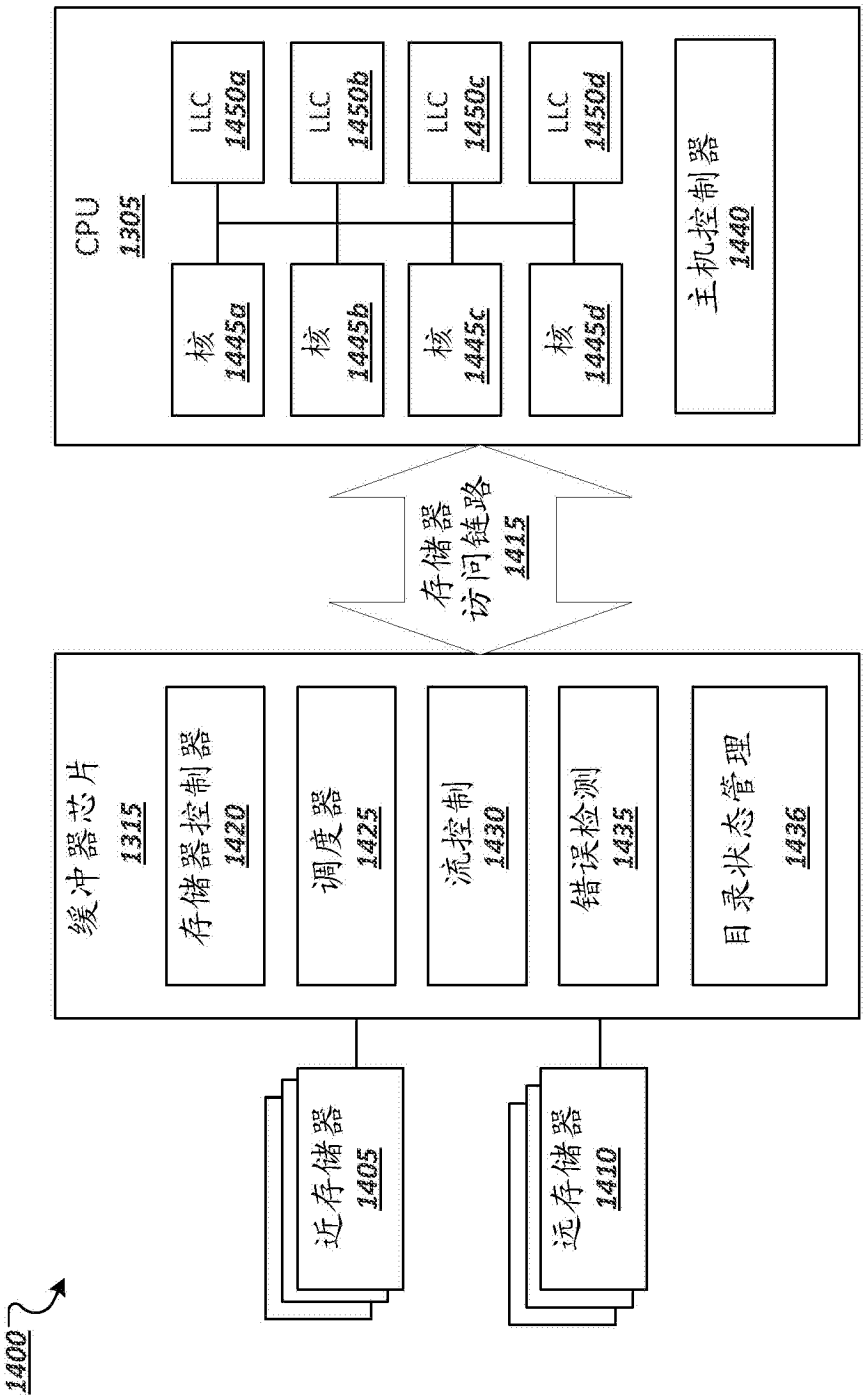

1400
近存储器
1405
远存储器
1410
缓冲器芯片
1315
存储器控制器
1420
调度器
1425
流控制
1430
错误检测
1435
目录状态管理
1436
存储器访问链路
1415
CPU
1305
核
1445a
核
1445b
核
1445c
核
1445d
LLC
1450a
LLC
1450b
LLC
1450c
LLC
1450d
主机控制器
1440

图 14

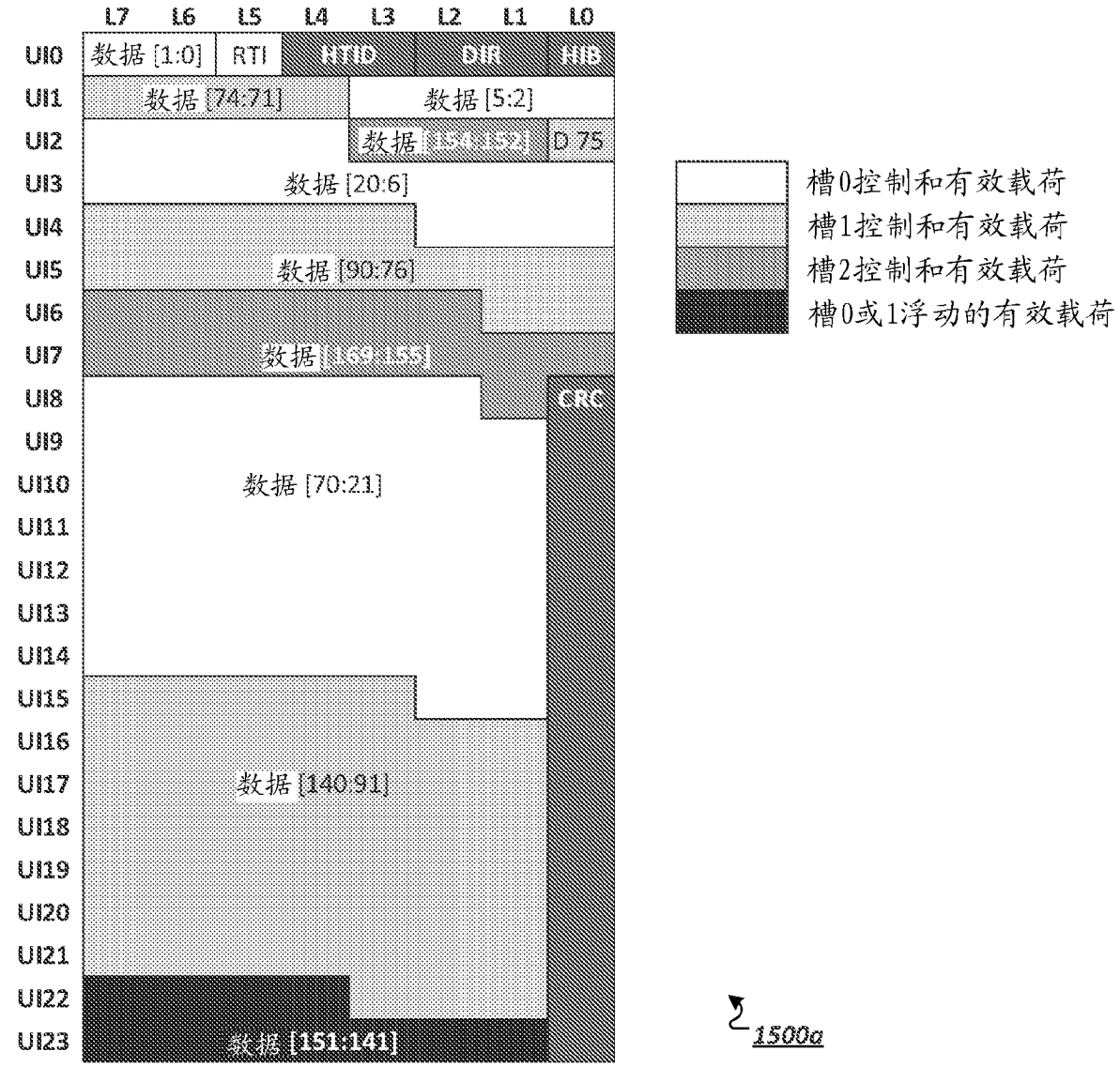

L7 L6 L5 L4 L3 L2 L1 L0
UI0 数据 [1:0] RTI HTID DIR HIB
UI1 数据 [74:71] 数据 [5:2]
UI2 数据 [154:152] D 75
UI3 数据 [20:6]
UI4
UI5 数据 [90:76]
UI6
UI7 数据 [169:155]
UI8 CRC
UI9
UI10 数据 [70:21]
UI11
UI12
UI13
UI14
UI15
UI16
UI17 数据 [140:91]
UI18
UI19
UI20
UI21
UI22
UI23 数据 [151:141]
槽0控制和有效载荷
槽1控制和有效载荷
槽2控制和有效载荷
槽0或1浮动的有效载荷
1500a

图 15A

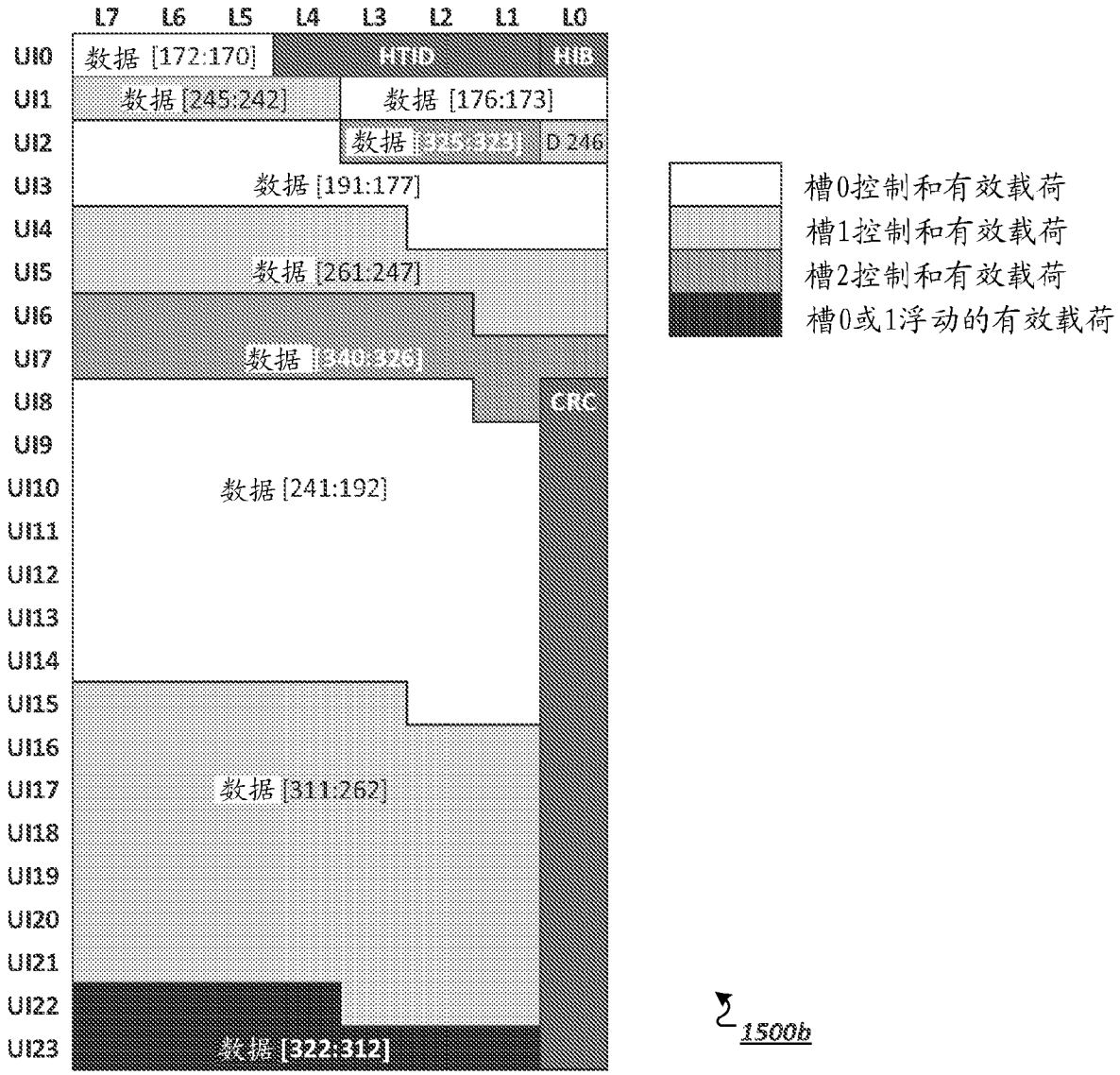

L7
L6
L5
L4
L3
L2
L1
L0
UI0
UI1
UI2
UI3
UI4
UI5
UI6
UI7
UI8
UI9
UI10
UI11
UI12
UI13
UI14
UI15
UI16
UI17
UI18
UI19
UI20
UI21
UI22
UI23
数据 [172:170]
HTID
HIB
数据 [245:242]
数据 [176:173]
数据 [275:262]
D 246
数据 [191:177]
数据 [261:247]
数据 [340:326]
CRC
数据 [241:192]
数据 [311:262]
数据 [322:312]
槽0控制和有效载荷
槽1控制和有效载荷
槽2控制和有效载荷
槽0或1浮动的有效载荷
1500b

图 15B

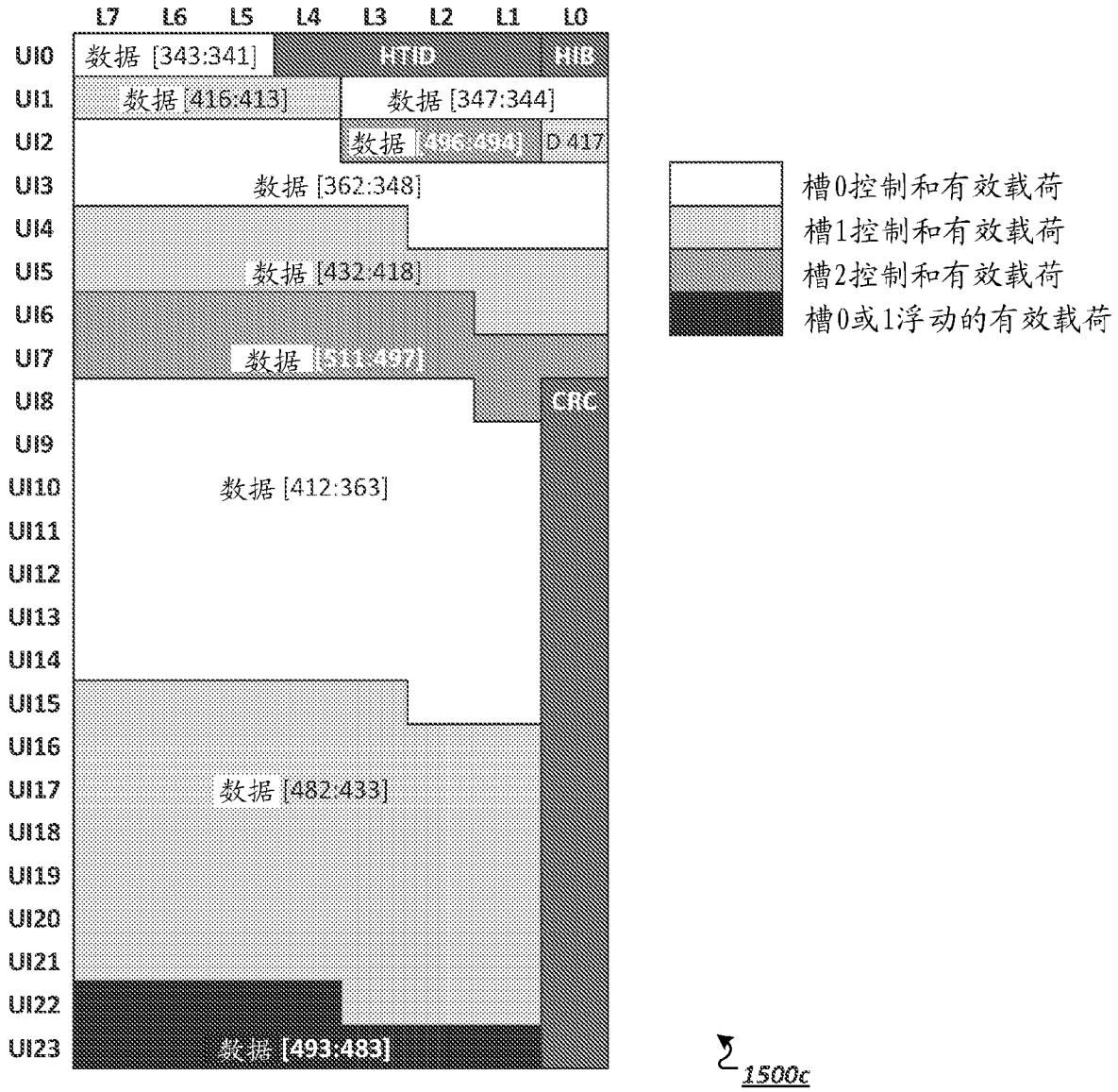

L7
L6
L5
L4
L3
L2
L1
L0
UI0
UI1
UI2
UI3
UI4
UI5
UI6
UI7
UI8
UI9
UI10
UI11
UI12
UI13
UI14
UI15
UI16
UI17
UI18
UI19
UI20
UI21
UI22
UI23
数据 [343:341]
HTID
HIB
数据 [416:413]
数据 [347:344]
数据
D 417
数据 [362:348]
数据 [432:418]
数据
CRC
数据 [412:363]
数据 [482:433]
数据 [493:483]
槽0控制和有效载荷
槽1控制和有效载荷
槽2控制和有效载荷
槽0或1浮动的有效载荷
1500c

图 15C

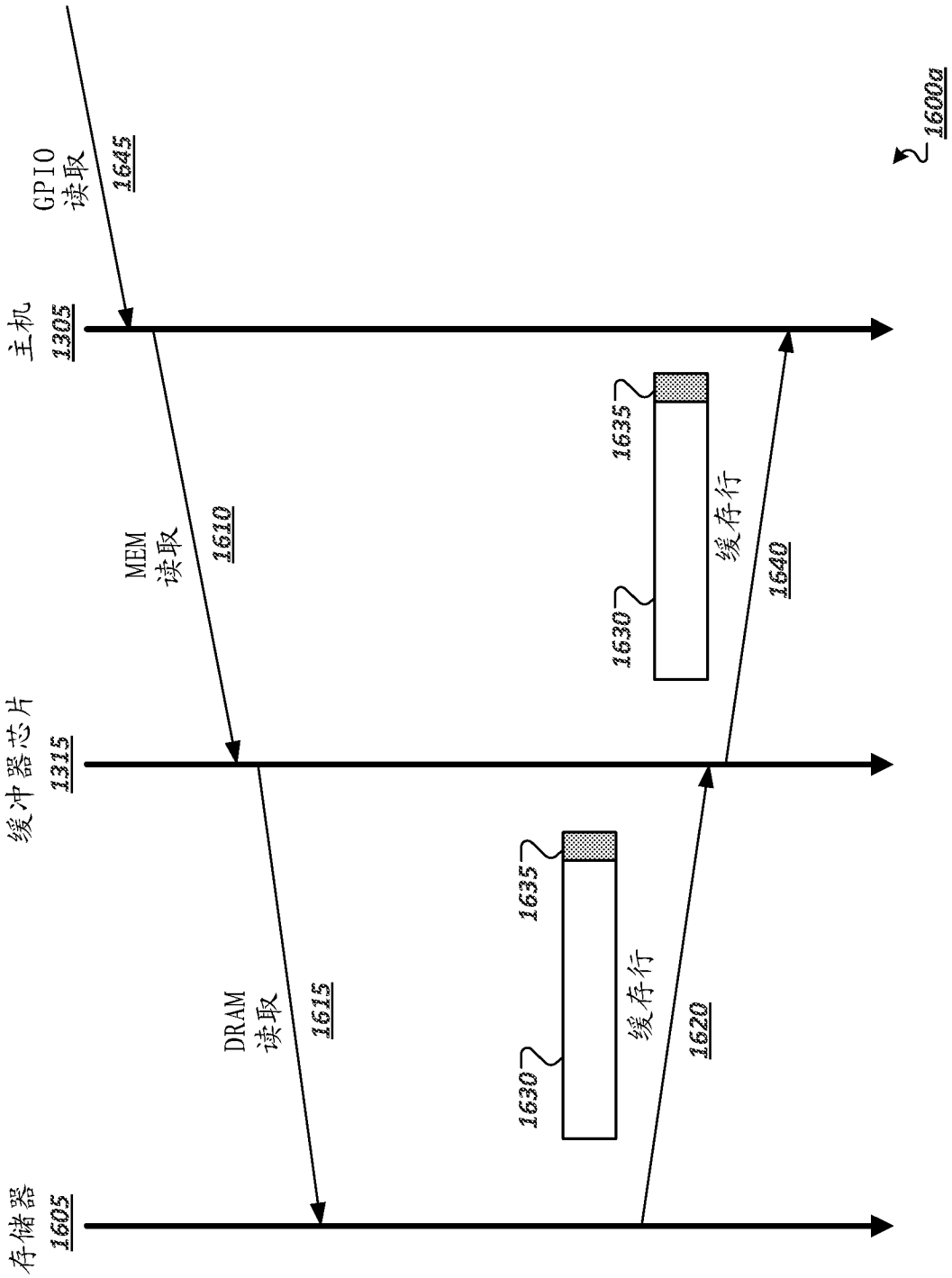

1600a
GPIO
读取
1645
主机
1305
MEM
读取
1610
缓冲器芯片
1315
DRAM
读取
1615
存储器
1605
1635
缓存行
1630
1640
1635
缓存行
1630
1620

图 16A

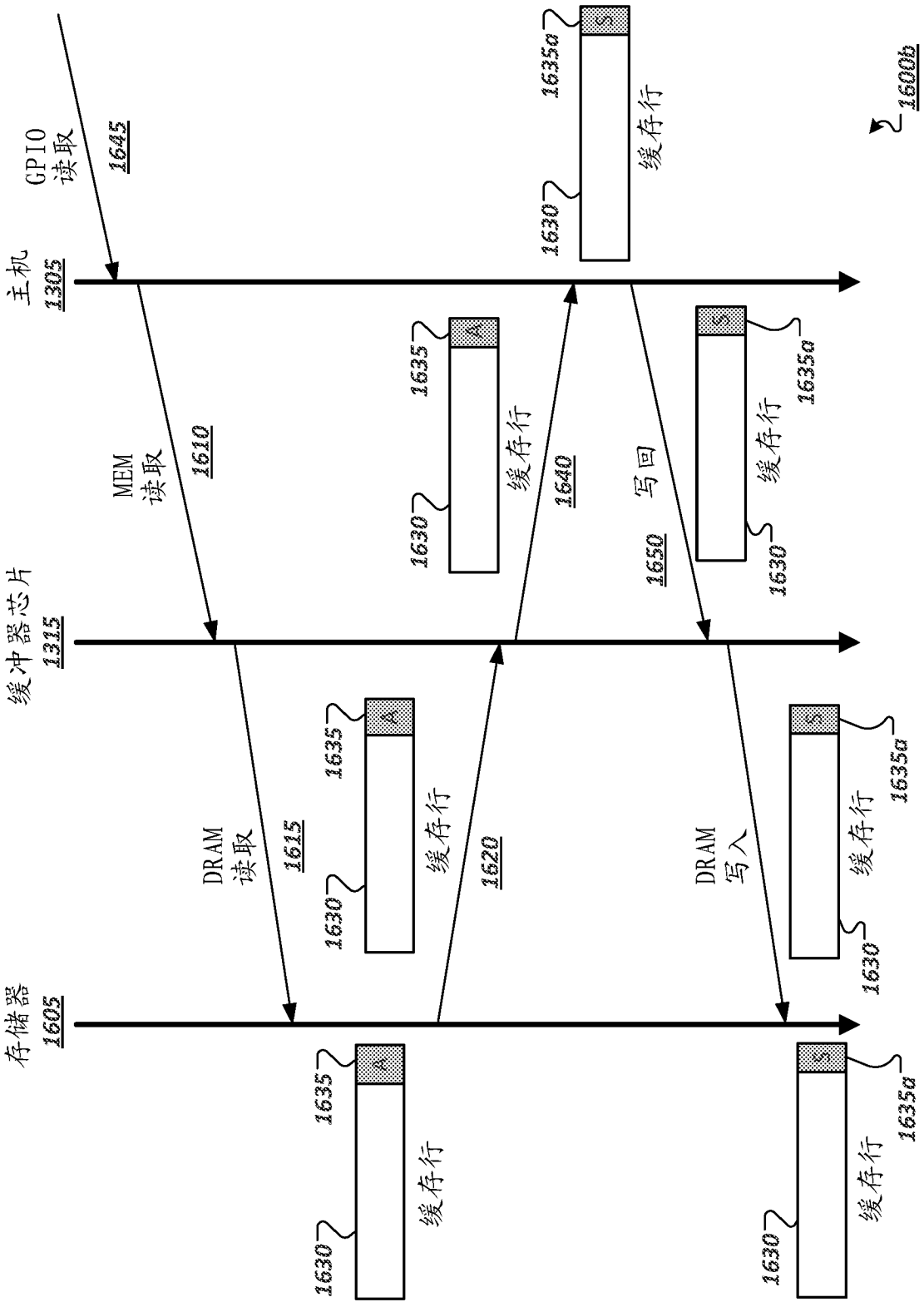

主机
1305
缓冲器芯片
1315
存储器
1605
GPIO
读取
1645
MEM
读取
1610
DRAM
读取
1615
1620
1640
写回
1650
DRAM
写入
1630
1635
1635a
缓存行
1600b

图 16B

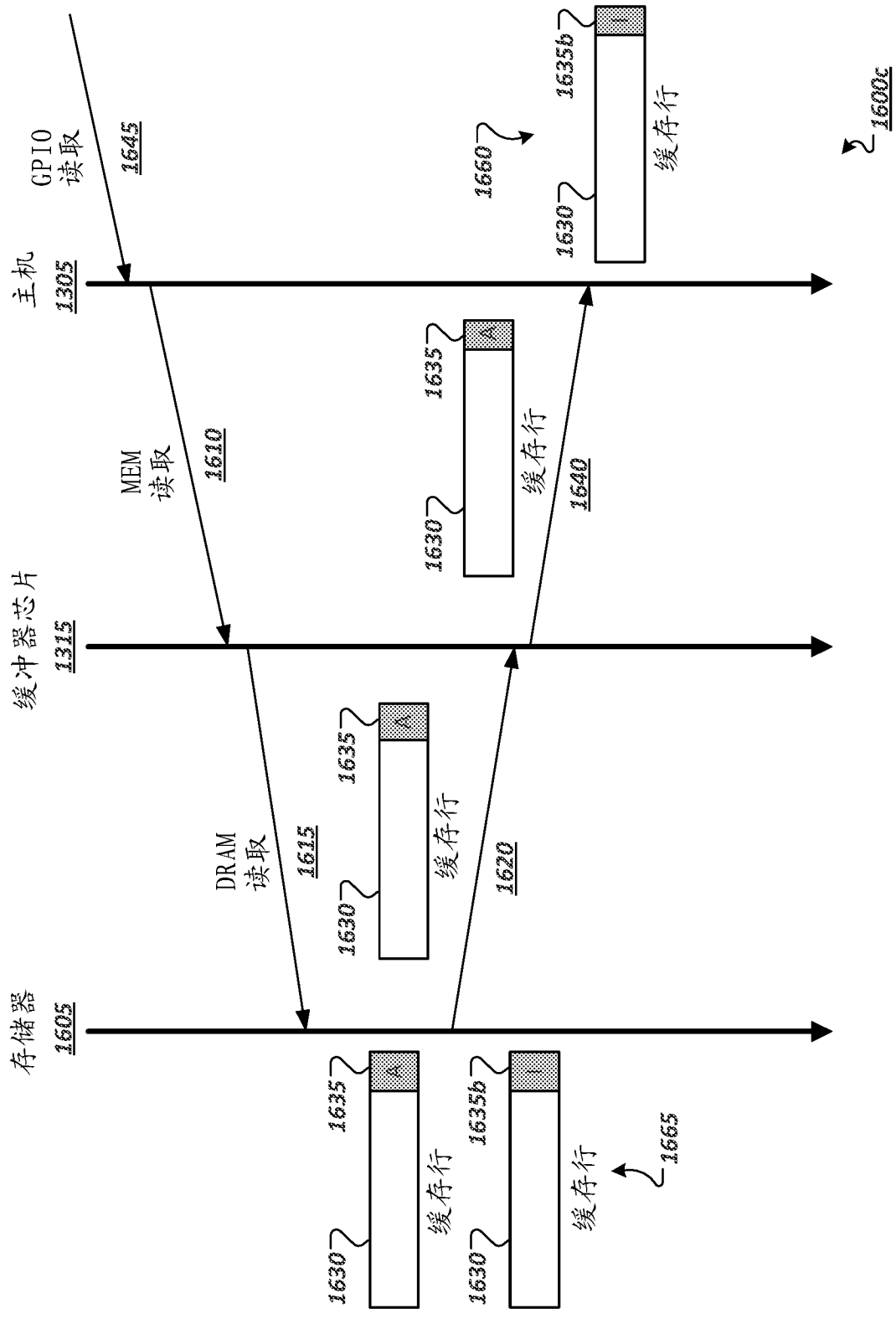

主机
1305
缓冲器芯片
1315
存储器
1605
GPIO
读取
1645
MEM
读取
1610
DRAM
读取
1615
1620
1640
1660
1665
1600c
1630
1635
1635b
缓存行
A
I

图 16C

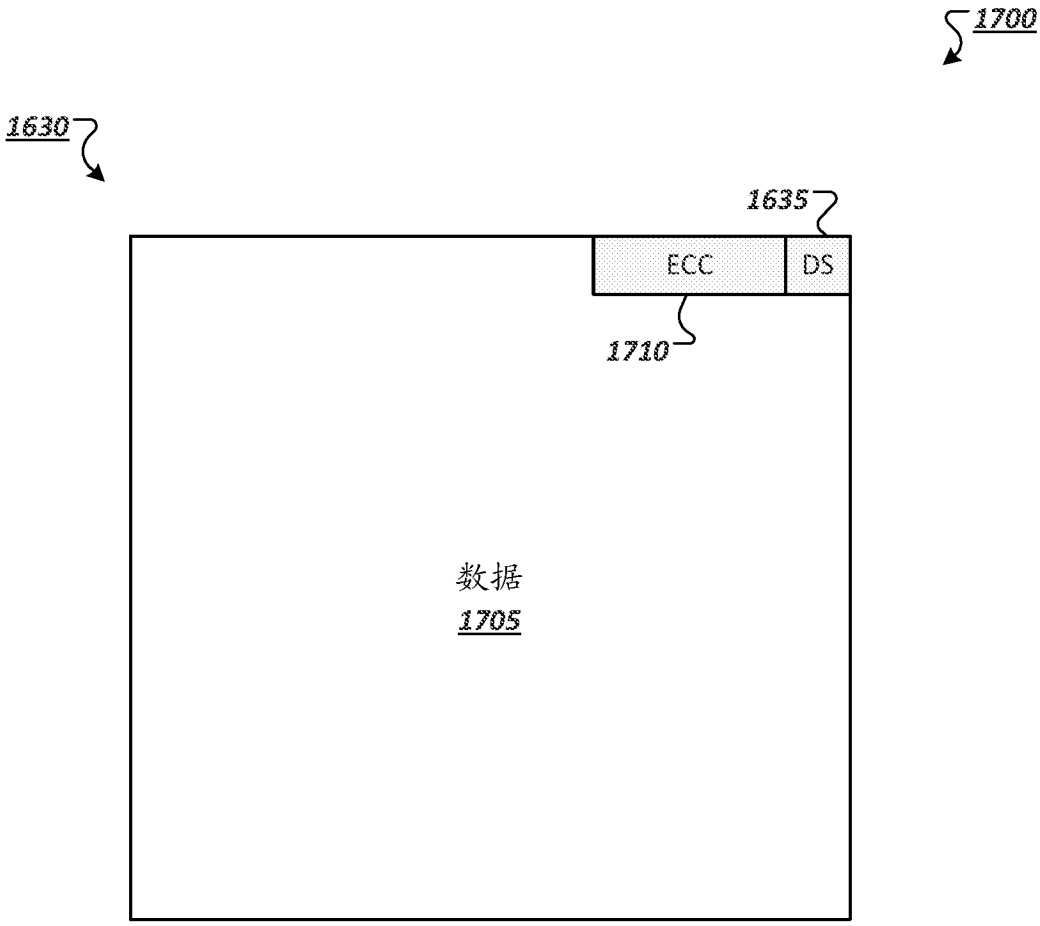


图 17

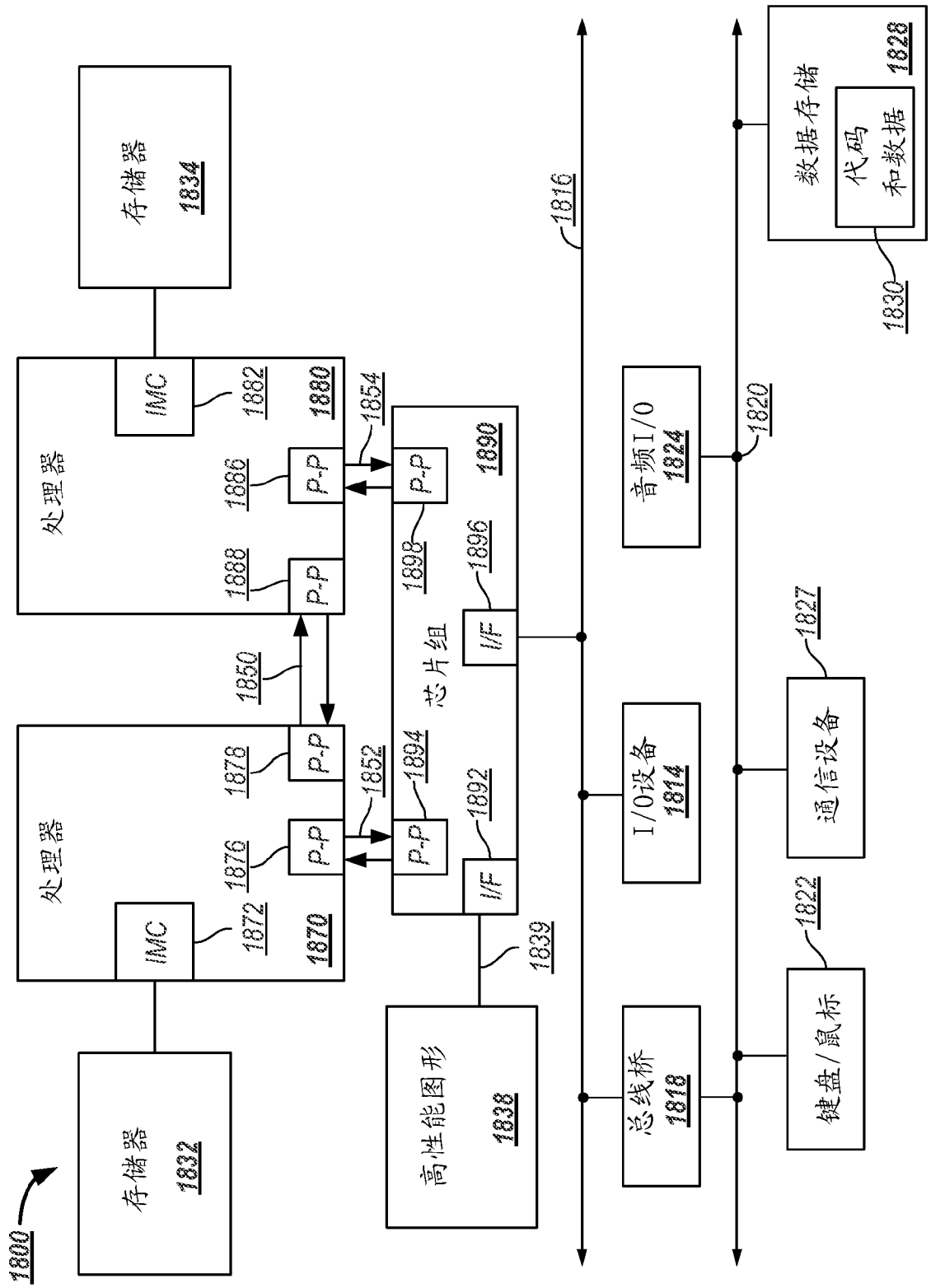

1800
存储器
1832
处理器
IMC
1872
1870
P-P
1876
P-P
1878
1852
处理器
IMC
1882
1880
P-P
1886
P-P
1888
1850
1854
芯片组
1890
P-P
1894
P-P
1898
I/F
1892
I/F
1896
1839
高性能图形
1838
存储器
1834
1816
总线桥
1818
I/O设备
1814
音频I/O
1824
1820
通信设备
1827
键盘/鼠标
1822
数据存储
代码
和数据
1828
1830

图 18

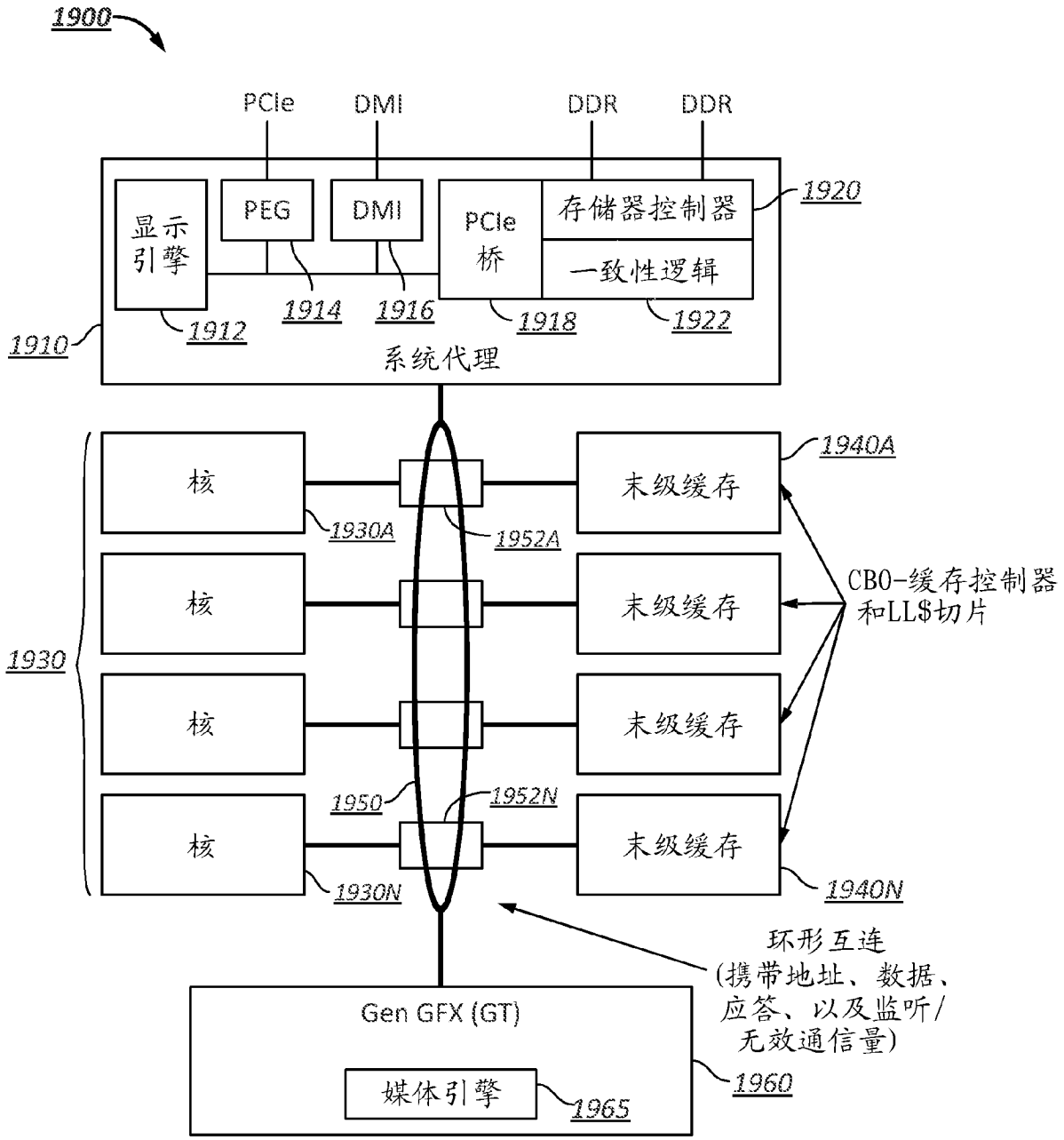

1900
PCIe
DMI
DDR
DDR
显示引擎
PEG
DMI
PCIe桥
存储器控制器
一致性逻辑
1920
1910
1912
1914
1916
1918
1922
系统代理
核
1930A
1952A
末级缓存
1940A
核
末级缓存
CBO-缓存控制器和LL$切片
1930
核
末级缓存
1950
1952N
核
末级缓存
1930N
1940N
环形互连
(携带地址、数据、应答、以及监听/无效通信量)
Gen GFX (GT)
媒体引擎
1965
1960

图 19