(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 105718390 A
(43)申请公布日 2016.06.29

(21)申请号 201510796006.9

(22)申请日 2015.11.18

(30)优先权数据
14/576,125 2014.12.18 US

(71)申请人 英特尔公司
地址 美国加利福尼亚

(72)发明人 M·C·仁 D·达斯夏尔马 M·韦格 V·伊耶

(74)专利代理机构 永新专利商标代理有限公司 72002
代理人 张立达 王英

(51)Int.Cl.
G06F 13/16(2006.01)

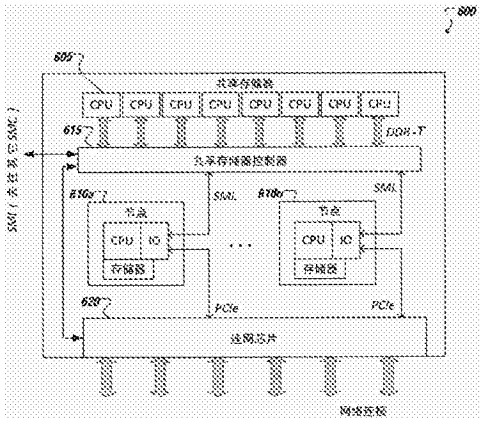
权利要求书2页 说明书26页 附图15页

(54)发明名称

共享存储器链路中的低功率进入

(57)摘要

在根据存储器访问链路协议的链路上发送数据，以对应于与共享存储器相关联的装载/存储类型操作，且该存储器访问链路协议覆盖另一个不同的链路协议上。发送进入低功率状态的请求，其中该请求包括在令牌的字段中编码的数据值，该令牌指示分组数据的起始，并进一步指示在该令牌之后发送的随后数据是否包括根据另一链路协议和存储器访问链路协议其中之一的数据。



CN 105718390 A

1.一种用于传送数据的装置,所述装置包括:

发送器,其用于发送数据以对应于与共享存储器相关联的装载/存储类型操作,其中,所述数据是在根据存储器访问链路协议的链路上进行发送的,并且所述存储器访问链路协议覆盖在另一个不同的链路协议上;

功率管理逻辑单元,其用于发送进入低功率状态的请求,其中所述请求包括在令牌的字段中编码的数据值,所述令牌指示分组数据的起始,并进一步指示在所述令牌之后发送的随后数据是否包括根据另一链路协议和所述存储器访问链路协议其中之一的数据。

2.根据权利要求1所述的装置,其中,所述另一链路协议包括通用输入/输出(I/O)互连协议。

3.根据权利要求2所述的装置,其中,所述令牌是根据所述通用I/O互连协议来规定的。

4.根据权利要求3所述的装置,其中,所述令牌的第一字段指示所述随后数据是包括存储器访问链路协议数据,还是包括通用I/O互连协议数据。

5.根据权利要求4所述的装置,其中,当所述第一字段指示所述随后数据包括存储器访问链路协议数据时,将所述数据值编码在所述令牌的第二字段中。

6.根据权利要求5所述的装置,其中,所述数据值包括根据所述存储器访问链路协议来规定的编码,并且使用所述第二字段来在所述通用I/O互连协议的物理层上,对所述编码进行隧道化。

7.根据权利要求2所述的装置,其中,基于所述令牌的实例,在所述链路上发送的数据在存储器访问链路协议数据和通用I/O互连协议数据之间切换。

8.根据权利要求7所述的装置,其中,在发送进入所述低功率状态的所述请求之前,所述功率管理逻辑单元检查所述通用I/O互连协议的数据是否在等待进行发送。

9.根据权利要求8所述的装置,其中,所述功率管理逻辑单元等待发送进入所述低功率状态的所述请求,直到发送了所述通用I/O互连协议数据。

10.根据权利要求7所述的装置,其中,所述存储器访问链路协议的链路层或者通用I/O互连协议的链路层能够触发物理层进入到低功率状态。

11.根据权利要求10所述的装置,其中,在所述通用I/O互连协议中用于进入低功率状态的信号,与在所述存储器访问链路协议中用于进入低功率状态的信号不同。

12.根据权利要求11所述的装置,其中,所述存储器访问链路协议规定了在其中发送在所述存储器访问链路协议中用于进入所述低功率状态的所述信号的周期性控制窗口。

13.一种用于传送数据的装置,所述装置包括:

接收器,其用于在链路上接收数据流,其中所述数据流包括令牌,所述令牌包括第一字段以指示在所述令牌随后的数据包括根据至少两种可选协议其中之一的数据,所述至少两种可选协议包括存储器访问协议和通用I/O协议,在所述链路上发送的所有数据都在所述通用I/O协议的物理层上进行发送,并且对所述令牌的第二字段进行编码以指示所述存储器访问协议的链路层的进入低功率链路状态的请求;以及

协议逻辑单元,其用于基于所述请求,发起进入所述低功率链路状态。

14.根据权利要求13所述的装置,还包括:

发送器,其用于发送对所述请求的响应,其中所述响应包括确认和否定确认其中之一,并且所述响应是在所述令牌的另一个实例中进行发送的。

15.根据权利要求14所述的装置，其中，所述响应是在所述令牌的另一实例的所述第二字段的实例中进行发送的。

16.一种用于传送数据的方法，所述方法包括：

在链路上接收数据流，其中所述数据流包括令牌的第一实例，所述令牌的实例包括第一字段以指示在所述令牌随后的数据包括根据至少两种可选协议其中之一的数据，所述至少两种可选协议包括存储器访问协议和通用I/O协议，在所述链路上发送的所有数据都在所述通用I/O协议的物理层上进行发送，并且对所述令牌的所述第一实例的第二字段进行编码以指示所述存储器访问协议的链路层的进入低功率链路状态的请求；

发送对所述请求的响应，其中所述响应包括确认和否定确认其中之一，并且所述响应是在所述令牌的另一个实例中进行发送的；以及

基于所述请求，发起进入所述低功率链路状态。

17.一种用于传送数据的装置，所述装置包括：

第一协议的链路层；

不同的第二协议的链路层；

所述第一协议的物理层，其中所述第二协议的数据在所述第一协议的所述物理层上进行发送；

垫层，其用于：

接收数据流，其中所述数据流包括令牌，所述令牌包括第一字段以指示在所述令牌随后的数据包括第一协议数据或者第二协议数据其中之一，并且对所述令牌的第二字段进行编码以指示所述第二协议的链路层的进入低功率链路状态的请求；

向所述物理层发送数据，以使所述物理层进入所述低功率状态。

18.根据权利要求17所述的装置，其中，所述第一协议包括通用I/O互连协议，并且所述第二协议包括存储器访问协议。

19.根据权利要求17所述的装置，还包括：

所述第一协议的一个或多个层以及所述第二协议的一个或多个层。

20.根据权利要求17所述的装置，其中，所述第一协议的所述链路层是禁用的。

共享存储器链路中的低功率进入

技术领域

[0001] 本公开内容涉及计算系统，特别地(但非排他性地)，本公开内容涉及计算系统中的部件之间的存储器访问。

背景技术

[0002] 半导体处理和逻辑设计的进步已准许在集成电路器件上可以存在的逻辑单元的数量增加。作为推论，计算机系统配置已从系统中的单一或多个集成电路演变成在各个集成电路上存在多核、多硬件线程和多逻辑处理器，以及在这些处理器中集成的其它接口。处理器或集成电路通常包括单一物理处理器管芯，其中该处理器管芯可以包括任意数量的内核、硬件线程、逻辑处理器、接口、存储器、控制器集线器等等。

[0003] 随着在更小的封装中容纳更多的处理能力的更大能力，更小的计算设备的普及度已增加。智能电话、平板设备、超薄笔记本和其它用户设备已成指数地增长。但是，这些更小的设备依赖于服务器以进行数据存储和超出形状因子的复杂处理。因此，对于高性能计算市场的需求(即，服务器空间)也发生了增长。例如，在现代服务器中，通常不仅存在具有多个内核的单一处理器，而且还存在多个物理处理器(其还称为多个插槽)来增加计算能力。但是，随着处理能力连同计算系统中的设备数量发生增加，在插槽和其它设备之间的通信也变得更加关键。

[0004] 事实上，互连已从更加传统的主要处理电通信的多点式总线，演变为有助于快速通信的完全成熟的互连架构。不幸的是，随着未来处理器按照甚至更高速率进行处理的需求，对于现有互连架构的能力也施加了相应的需求。

附图说明

[0005] 图1示出了包括互连架构的计算系统的实施例。

[0006] 图2示出了包括分层栈的互连架构的实施例。

[0007] 图3示出了在互连架构中生成或者接收的请求或分组的实施例。

[0008] 图4示出了用于互连架构的发送器和接收器对的实施例。

[0009] 图5示出了互连的示例性分层协议栈。

[0010] 图6示出了示例性节点的实施例的简化框图。

[0011] 图7示出了包括多个节点的示例性系统的实施例的简化框图。

[0012] 图8示出了包括垫层(shim)逻辑单元的节点的示例性逻辑单元。

[0013] 图9是根据示例性共享存储器链路发送的数据的图示。

[0014] 图10示出了在数据流中嵌入的示例性控制窗口的图示。

[0015] 图11A-11B示出了使用嵌入式控制窗口的示例性握手的流程图。

[0016] 图12是另一种示例性握手的图示。

[0017] 图13是数据成帧令牌的示例性起始的图示。

[0018] 图14是示出用于进入低功率链路状态的示例性技术的流程图。

[0019] 图15示出了用于包括多核处理器的计算系统的框图的实施例。

[0020] 图16示出了用于包括多核处理器的计算系统的框图的另一个实施例。

[0021] 各个附图中的相同附图标记和标注指示相同的元素。

具体实施方式

[0022] 在下文的描述中，为了对本发明提供一个透彻理解，对众多特定细节进行了阐述，诸如以下示例：特定类型的处理器和系统配置、特定的硬件结构、特定的架构和微架构细节、特定的寄存器配置、特定的指令类型、特定的系统部件、特定的测量/高度、特定的处理器流水线阶段和操作等等。但是，对于本领域普通技术人员来说将显而易见的是，不必采用这些特定细节来实践本发明。在其它实例中，为了避免对本发明造成不必要的模糊，没有详细描述公知的部件或方法，例如，特定的或替代的处理器架构、用于所描述的算法的特定逻辑电路/代码、特定的固件代码、特定的互连操作、特定的逻辑配置、特定的制造技术和材料、特定的编译器实现方式、利用代码的算法的特定表示、特定的掉电和门控技术/逻辑、以及计算机系统的其它特定操作细节。

[0023] 虽然可以参照特定的集成电路中的(例如，计算平台或微处理器中的)节能和能量效率来描述下面的实施例，但其它实施例可适用于其它类型的集成电路和逻辑器件。本文所描述的实施例的类似技术和教导，可以应用于也能从更佳的能量效率和节能中获益的其它类型的电路或半导体器件。例如，所公开的实施例并不限于桌面型计算机系统或超级本™。其还可以用于其它设备，例如，手持设备、平板设备、其它薄笔记本、片上系统(SOC)设备和嵌入式应用。手持设备的一些示例包括蜂窝电话、互联网协议设备、数码相机、个人数字助理(PDA)和手持式PC。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、网络集线器、广域网(WAN)交换机、或者能够执行下面所教示的功能和操作的任何其它系统。此外，本文所描述的装置、方法和系统并不限于物理计算设备，还可以涉及用于节能和能量效率的软件优化。如下面的描述中将变得显而易见的，本文所描述的方法、装置和系统的实施例(无论是参照硬件、固件、软件，还是它们的组合)对于平衡性能考量的“绿色技术”未来都是至关重要的。

[0024] 随着计算系统不断提高，其中的部件也变得越来越复杂。结果，用于在这些部件之间进行耦合和通信的互连架构在复杂度上也不断增加，以确保满足带宽需求来实现最优的部件操作。此外，不同的市场分部要求互连架构的不同方面来适合市场的需求。例如，服务器需要更高的性能，而移动生态系统有时能够为了省电而牺牲整体性能。另外，大多数结构的唯一目的是在具有最大省电的情况下，提供最大可能的性能。下面将讨论多种互连，它们将潜在地从本文所描述的本发明的方面受益。

[0025] 一种互连结构架构包括外围部件互连(PCI)高速(PCIe)架构。PCIe的主要目标是使不同供应商的部件和设备能够在开放的架构下进行互操作，其跨度多个市场分部、客户端(桌面型和移动型)、服务器(标准和企业)、以及嵌入式设备和通信设备。PCI Express是针对各种各样的未来计算和通信平台所规定(define)的一种高性能、通用I/O互连。诸如使用模型、装载/存储架构和软件接口之类的一些PCI属性历经其修订版仍在维持，而先前的并行总线实现方式已被高度可扩展的、完全串行接口来替代。PCI Express的最新版本充分利用点对点互连中的改进、基于交换的技术和分组化协议，来传送新的水平的性能和特征。

功率管理、服务质量(QoS)、热插拔/热交换支持、数据完整性和错误处理都是PCI Express所支持的高级特征中的一些。

[0026] 参见图1,示出了由点对点链路组成的结构的实施例,其中该点对点链路对部件集合进行互连。系统100包括耦合到控制器集线器115的系统存储器110和处理器105。处理器105包括诸如微处理器、主机处理器、嵌入式处理器、协同处理器或其它处理器之类的任何处理单元。处理器105通过前端总线(FSB)106耦合到控制器集线器115。在一个实施例中,FSB 106是如下所述的串行点对点互连。在另一个实施例中,链路106包括遵循不同的互连标准的串行的、差分互连架构。

[0027] 系统存储器110包括任何存储器设备,例如,随机存取存储器(RAM)、非易失性(NV)存储器或者系统100中的设备可访问的其它存储器。系统存储器110通过存储器接口116来耦合到控制器集线器115。存储器接口的示例包括双倍数据速率(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。

[0028] 在一个实施例中,控制器集线器115是外围部件互连高速(PCIe或PCIE)互连等级中的根集线器、根复合体或者根控制器。控制器集线器115的示例包括芯片集、存储控制器集线器(MCH)、北桥、互连控制器集线器(ICH)、南桥和根控制器/集线器。通常,术语芯片集指代两个物理独立的控制器集线器,即,耦合到互连控制器集线器(ICH)的存储控制器集线器(MCH)。应当注意,当前系统通常包括与处理器105集成的MCH,而控制器115以与如下所述的类似方式与I/O设备进行通信。在一些实施例中,通过根复合体115来可选地支持对等路由。

[0029] 这里,控制器集线器115通过串行链路119耦合到交换器/桥120。输入/输出模块117和121(它们还可以称为接口/端口117和121)包括/实现分层协议栈,以提供控制器集线器115和交换器120之间的通信。在一个实施例中,多个设备能够耦合到交换器120。

[0030] 交换器/桥120将来自设备125的分组/消息向上游路由到控制器集线器115(即,朝着根复合体沿层级往上),将来自处理器105或系统存储器110的分组/消息向下游路由到设备125(即,离开根控制器沿层级往下)。在一个实施例中,交换器120被称为多个虚拟PCI到PCI桥设备的逻辑组件。设备125包括要耦合到电子系统的任何内部或外部设备或部件,诸如I/O设备、网络接口控制器(NIC)、插入式卡、音频处理器、网络处理器、硬盘驱动器、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行话中,例如将设备称为端点。虽然没有详细地示出,但设备125可以包括PCIe到PCI/PCI-X桥,以支持传统或其它版本PCI设备。通常,将PCIe中的端点设备分类成传统、PCIe或集成根复合体的端点。

[0031] 此外,图形加速器130也通过串行链路132耦合到控制器集线器115。在一个实施例中,图形加速器130耦合到MCH,MCH耦合到ICH。随后,交换器120以及相应地I/O设备125耦合到ICH。I/O模块131和118也用于实现分层协议栈,以便在图形加速器130和控制器集线器115之间进行通信。类似于上面讨论的MCH,图形控制器或图形加速器130自身可以集成在处理器105中。

[0032] 转到图2,示出了一种分层协议栈的实施例。分层协议栈200包括任何形式的分层通信栈,例如,快速通道互连(QPI)栈、PCIe栈、下一代高性能计算互连栈或者其它分层栈。虽然下面参照图1-4所紧跟着的讨论与PCIe栈有关,但相同的概念也可应用于其它互连栈。

在一个实施例中，协议栈200是包括事务层205、链路层210和物理层220的PCIe协议栈。诸如图1中的接口117、118、121、122、126和131之类的接口可以表示成协议栈200。表示成通信协议栈还可以称为实现/包括协议栈的模块或接口。

[0033] PCI Express使用分组来在部件之间传送信息。在事务层205和数据链路层210中形成分组，以携带从发送部件到接收部件的信息。随着所发送的分组流经其它层，利用在这些层对分组进行处理所必需的另外信息来扩展分组。在接收侧，发生反向过程，且分组将从它们的物理层220表示变换到数据链路层210表示，并最后(对于事务层分组)变换成接收设备的事务层205可以处理的形式，来获得分组。

[0034] 事务层

[0035] 在一个实施例中，事务层205用于提供设备的处理内核和互连架构之间的接口，例如，数据链路层210和物理层220。在该方面，事务层205的主要责任是分组(即，事务层分组或TLP)的组装和拆卸。通常，事务层管理针对TLP的基于信用的流控制。PCIe实现拆分事务，即，具有通过时间来分隔的请求和响应的事务，其允许链路携带其它业务，同时目标设备收集用于进行响应的数据。

[0036] 此外，PCIe使用基于信用的流控制。在该方案中，设备将用于事务层205中的每一个接收缓冲区的信用度的初始量告之于众。位于该链路的另一端的外部设备(例如，图1中的控制器集线器115)对每一个TLP所消耗的信用量进行计数。如果该事务没有超过信用额度，则可以发送该事务。在接收到响应之后，恢复信用量。信用方案的一种优点在于信用返回的时延并不影响性能(倘若没有达到信用额度的话)。

[0037] 在一个实施例中，四种事务地址空间包括配置地址空间、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请求和写请求其中的一个或多个，以将数据传送往/返存储器映射的位置。在一个实施例中，存储器空间事务能够使用两种不同的地址格式，例如，短地址格式(如，32比特地址)或长地址格式(如，64比特地址)。配置空间事务用于访问PCIe设备的配置空间。针对配置空间的事务包括读请求和写请求。对消息空间事务(或者，简单的消息)进行规定，以支持PCIe代理之间的带内通信。

[0038] 因此，在一个实施例中，事务层205对分组报头/有效载荷206进行组装。可以在PCIe规范网站的PCIe规范中，找到用于当前的分组报头/有效载荷的格式。

[0039] 快速地参见图3，示出了PCIe事务描述符的实施例。在一个实施例中，事务描述符300是用于携带事务信息的机制。在该方面，事务描述符300支持系统中的事务的识别。其它潜在用途包括：跟踪缺省事务排序的修改以及事务与信道的关联性。

[0040] 事务描述符300包括全局标识符字段302、属性字段304和信道标识符字段306。在所示出的例子中，将全局标识符字段302描述成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对于所有显著请求来说是唯一的。

[0041] 根据一种实现方式，本地事务标识符字段308是请求代理所生成的字段，且其对于对该请求代理来说需要完成的所有显著请求来说是唯一的。此外，在该示例中，源标识符310在PCIe等级中唯一地标识请求方代理。因此，与源ID 310一起，本地事务标识符308字段在等级域中提供事务的全局标识。

[0042] 属性字段304指定该事务的特性和关系。在该方面，潜在地使用属性字段304来提供允许对事务的缺省处理进行修改的另外信息。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和不监听字段318。这里，发起方可以对优先级子字段312进行修改，以便为事务分配优先级。将保留属性字段314保留以便未来使用，或者供应商定义的用途。可以使用保留属性字段来实现利用优先级或安全属性的可能使用模型。

[0043] 在该示例中，使用排序属性字段316来提供用于传送排序类型的可选信息，其中该信息可以修改缺省的排序规则。根据一种示例性实现方式，排序属性“0”指示将应用缺省的排序规则，其中排序属性“1”指示放松的排序，其中写入可以在相同的方向传递写入，而读完成可以在相同的方向传递写入。使用监听属性字段318来判断是否对事务进行监听。如图所示，信道ID字段306标识与事务相关联的信道。

[0044] 链路层

[0045] 链路层210(其还称为数据链路层210)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的责任是提供用于在链路的两个部件之间交换事务层分组(TLP)的可靠机制。数据链路层210的一侧接受事务层205所组合的TLP，应用分组序列标识符211(即，标识号或分组编号)，计算和应用错误检测码(即，CRC 212)，并向物理层220提交修改的TLP以便通过物理层向外部设备传输。

[0046] 物理层

[0047] 在一个实施例中，物理层220包括逻辑子块221和电子子块222，以物理地向外部设备发送分组。这里，逻辑子块221负责物理层221的“数字”功能。在该方面，该逻辑子块包括：发送段，以准备用于由物理子块222进行传输的输出信息；以及接收器段，用于在将接收的信息传送给链路层210之前，识别和准备该接收的信息。

[0048] 物理块222包括发送器和接收器。逻辑子块221向发送器提供符号，其中发送器对这些符号进行串行化，并将其发送到外部设备上。向接收器提供来自外部设备的串行化的符号，并将所接收的信号转换成比特流。将该比特流进行去串行化，并将其提供给逻辑子块221。在一个实施例中，使用8b/10b传输编码，其中发送/接收十比特符号。这里，使用特殊符号来利用帧223对分组进行成帧。此外，在一个示例中，接收机还提供从输入的串行流中恢复的符号时钟。

[0049] 如上所述，虽然参照PCIe协议栈的特定实施例来讨论了事务层205、链路层210和物理层220，但分层协议栈并不受此限制。事实上，可以包括/实现任何分层协议。举例而言，表示成分层协议的端口/接口，包括：(1)用于对分组进行组装的第一层(即，事务层)；用于对分组进行序列化的第二层(即，链路层)，以及用于发送这些分组的第三层(即，物理层)。举一个特定的例子，使用公共标准接口(CSI)分层协议。

[0050] 接着参见图4，示出了PCIe串行点对点结构的实施例。虽然示出了PCIe串行点对点链路的实施例，但串行点对点链路并不受此限制，这是由于其包括用于发送串行数据的任何传输路径。在所示出的实施例中，基础PCIe链路包括两个低电压、差分驱动信号对：发送对406/411和接收对412/407。因此，设备405包括用于向设备410发送数据的传输逻辑单元406和用于从设备410接收数据的接收逻辑单元407。换言之，在PCIe链路中包括两个发送路径(即，路径416和417)和两个接收路径(即，路径418和419)。

[0051] 传输路径指代用于发送数据的任何路径，例如，传输线、铜线、光纤线缆、无线通信信道、红外通信链路或者其它通信路径。两个设备(例如，设备405和设备410)之间的连接称为链路(例如，链路415)。链路可以支持一个通道，每一个通道表示差分信号对集合(一对用

于传输，一对用于接收)。为了调整带宽，链路可以聚合通过xN来表示的多个通道，其中N是任何支持的链路宽度(例如，1、2、4、8、12、16、32、64或更宽)。

[0052] 差分对指代用于发送差分信号的两个传输路径(例如，线路416和417)。举例而言，当线路416从低电压电平切换到高电压电平时(即，上升沿)，线路417从高逻辑电平驱动到低逻辑电平(即，下降沿)。差分信号潜在地说明更佳的电特性，例如，更佳的信号完整性(即，交叉耦合、电压过冲/下冲、瞬时振荡(ringing)等等)。这允许实现更佳的时间窗口，其能获得更快速的传输频率。

[0053] 可以利用包括PCIe的现有互连和通信架构的物理层，来在系统中提供共享存储器和I/O服务。传统上，不能使用传统的装载/存储(LD/ST)存储器语义，在独立的系统之间共享可高速缓存的存储器。独立系统或“节点”在下面的意义上可以是独立的：其用作单一逻辑实体，由单一操作系统(和/或单一BIOS或虚拟机监视程序(VMM))来控制，和/或具有独立的故障域。单一节点可以包括一个或多个处理器设备，实现在单一电路板或多个电路板上，并包括本地存储器(其包括同一节点上的设备能够使用LD/ST语义来访问的可高速缓存存储器)。在节点之内，共享存储器可以包括能由节点中的几个不同的处理器(例如，中央处理单元(CPU))访问的一个或多个存储器块(例如，随机存取存储器(RAM))。共享存储器还可以包括处理器或者节点中的其它设备的本地存储器。节点中具有共享存储器的多个设备可以共享该共享存储器中的单一数据视图。涉及共享存储器的I/O通信可以具有非常低的时延，并允许所述多个处理器快速地访问该存储器。

[0054] 传统上，在不同的节点之间的存储器共享不允许根据装载/存储范式进行存储器共享。例如，在一些系统中，通过分布式存储器架构，有助于不同的节点之间的存储器共享。在传统的解决方案中，计算任务对本地数据进行操作，如果期望另一个节点的数据，则该计算任务(例如，由另一个CPU节点来执行)例如使用通信协议栈(如，以太网、无线带宽或者另一种分层协议)，通过通信信道与另一节点进行通信。在传统的多节点系统中，不同的节点的处理器并不需要了解数据位于的位置。与使用装载/存储范式在节点中进行存储器共享相比，使用传统方法(例如，通过协议栈)来共享数据具有明显更高的时延。具体而言，不是直接对共享存储器中的数据进行寻址和操作，而是，一个节点可以使用诸如以太网(或无线带宽)之类的现有协议握手来从另一个节点请求数据，而源节点可以提供该数据，使得请求节点可以存储和操作该数据，等等其他示例。

[0055] 在一些实现方式中，可以提供允许使用装载/存储(LD/ST)存储器语义，在独立节点之间对存储器进行共享以实现排他或共享访问的共享存储器架构。举一个例子，可以在引脚的公用集合或者引脚的单独集合上，输出存储器语义(和目录信息，如果有的话)连同I/O语义(用于诸如PCIe之类的协议)。在这种系统中，该改进的共享存储器架构可以使系统中的多个节点里的每一个节点维持其自己独立的故障域(和本地存储器)，同时实现由节点进行访问的共享存储器池，以及使用根据LD/ST语义的存储器，在节点之间实现低时延消息传送。在一些实现方式中，可以在不同的节点之间动态地(或静态地)分配这种共享存储器池。因此，当需要出现时，例如，人们可以将系统的各个节点配置成动态变化的节点组，以利用共享存储器基础结构，对各种任务协作地和灵活地工作。

[0056] 在一个实施例中，可以在高性能计算平台(例如，工作站或服务器)中提供互连，以连接处理器、加速器、计算块、I/O设备等等。互连架构可以是分层的，以包括诸如协议层(一

致的、非一致的，以及可选地其它基于存储器的协议)、路由层、链路层和物理层之类的规定层。图5示出了互连的示例性分层协议栈的实施例。在一些实现方式中，图5中所示出的分层中的至少一些是可选的。每一个层处理其自己层级粒度或者数量的信息(具有分组530的协议层505a、b，具有微片(flit)535的链路层510a、b，以及具有元(phit)540的物理层505a、b)。应当注意，在一些实施例中，基于实现方式，分组可以包括部分微片、单一微片或者多个微片。

[0057] 作为第一示例，元540的宽度包括链路宽度与比特的1对1映射(例如，20比特链路宽度包括20比特的元等等)。微片可以具有更大的大小，例如，184、192或200比特。应当注意，如果元540是20比特宽，而微片535的大小是184比特，则其占用分数数量的元540来发送一个微片535(例如，20比特的9.2个元发送184比特的微片535，或者20比特的9.6个元发送192比特的微片等等其它示例)。应当注意，物理层处基础链路的宽度可以改变。例如，每一方向的通道的数量可以包括2、4、6、8、10、12、14、16、18、20、22、24等等。在一个实施例中，链路层510a、b能够在单一微片中嵌入不同事务的多个片段，并且可以将一个或多个报头(例如，1、2、3、4)嵌入在该微片中。在一个示例中，可以将这些报头分割到相应的时隙，以使该微片中的多个消息的目的地去往不同的节点。

[0058] 在一个实施例中，物理层505a、b可以负责在物理介质(电介质或光介质等等)上快速地传送信息。物理链路可以是两个链路层实体(例如，层505a和505b)之间的点对点连接。链路层510a、b可以从上层抽象物理层505a、b，并提供可靠地传输数据(以及请求)和在两个直接连接的实体之间管理流控制的能力。链路层还可以负责将物理信道虚拟化成多个虚拟信道和消息类别。协议层520a、b在将协议消息移交给物理层505a、b以通过物理链路进行传输之前，依赖于链路层510a、b来将协议消息映射到适当的消息类别和虚拟信道。链路层510a、b可以支持诸如请求、监听、响应、回写、非一致数据等等其它示例之类的多种消息。

[0059] 可以规定物理层505a、b(或者PHY)以实现在电子层(即，连接两个部件的电导体)之上和链路层510a、b之下，如图5中所示。物理层和相应的逻辑单元可以位于每个代理上，并连接彼此之间分离的两个代理(A和B)(例如，链路的两端的设备上)上的链路层。本地和远程电子层通过物理介质(例如，导线、导体、光纤等等)来连接。在一个实施例中，物理层505a、b具有两个主要阶段(初始化和操作)。在初始化期间，连接对于链路层而言是不透明的，而信令可能涉及定时状态和握手事件的组合。在操作期间，连接对于链路层而言是透明的，而信令处于某个速度，其中所有通道一起操作成单一链路。在操作阶段期间，物理层从代理A向代理B，以及从代理B向代理A传输微片。该连接还称为链路，并且当与链路层交换微片和当前配置(例如，宽度)的控制/状态时，从链路层中抽象包括介质、宽度和速度的一些物理方面。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)。

[0060] 在一个实施例中，可以实现链路层510a、b，以便在两个协议或路由实体之间提供可靠的数据传输。链路层可以从协议层520a、b抽象物理层505a、b，并可以负责两个协议代理(A、B)之间的流控制，并向协议层(消息类别)和路由层(虚拟网络)提供虚拟信道服务。通常，协议层520a、b和链路层510a、b之间的接口可以处于分组层级。在一个实施例中，链路层处最小传输单元称为微片，其具有指定数量的比特(例如，192比特)或者一些其它命名。链路层510a、b依赖于物理层505a、b，以将物理层505a、b的传输单元(元)成帧成链路层510a、b

的传输单元(微片)。此外,链路层510a、b可以被逻辑地拆分成两个部分(发送器和接收器)。一个实体上的发送器/接收器对可以连接到另一个实体上的接收器/发送器对。通常在微片和分组的基础上,执行流控制。还潜在地在微片基础上,执行错误检测和纠错。

[0061] 在一个实施例中,路由层515a、b可以提供用于将事务从源路由到目的地的灵活和分布式方法。由于可以通过每个路由器处的可编程路由表(在一个实施例中,通过固件、软件或者其组合来执行该编程),来指定针对多个拓扑的路由算法,因此该方案是灵活的。该路由功能可以是分布式的,可以通过一系列的路由步骤来实现该路由,其中通过对源、中间或目的地路由器处的表进行查询,来规定每一个路由步骤。可以利用源路由器处的查表,将分组投放到结构中。可以利用中间路由器处的查表,将分组从输入端口路由到输出端口。可以利用目的端口处的查表,来传送到目的协议代理处。应当注意,在一些实现方式中,路由层可能较薄,这是由于规范没有详细地规定路由表和因此的路由算法。这种情形允许具有灵活性和多种多样的使用模型,其包括由系统实现方式所规定的灵活平台架构拓扑。路由层515a、b依赖于链路层510a、b来提供多达三个(或更多)虚拟网络(VN)的使用,举一个例子,两个无死锁VN(VN0和VN1),它们具有每个虚拟网络中规定的几种消息类别。可以在链路层中规定共享的自适应虚拟网络(VNA),但这种自适应网络不能直接暴露在路由概念中,这是由于每一种消息类别和虚拟网络可能具有专用的资源和保证的转发进度,以及其它特征和示例。

[0062] 在一个实施例中,互连架构可以包括一致性协议层520a、b,以支持对来自存储器的数据线进行高速缓存的代理。希望对存储器数据进行高速缓存的代理,可以使用该一致性协议来读取数据线以装载到其高速缓存中。希望修改其高速缓存中的数据线的代理,可以在修改数据之前,使用该一致性协议来获取该数据线的所有权。在修改数据线之后,代理可以遵循在其高速缓存中保持该数据的协议要求,直到其将该数据线写回到存储器中,或者将该数据线包括在针对外部请求的响应中为止。最后,代理可以履行外部请求,以使其高速缓存中的数据线无效。这种协议通过规定所有的高速缓存代理都可遵循的规则,来确保数据的一致性。其还提供了不具有高速缓存的代理一致性地读取和写入存储器数据的方式。

[0063] 可以实施两种条件来支持使用示例性一致性协议的事务。首先,举例而言,该协议可以在每一地址基础上,在代理的高速缓存中的数据之间,以及这些数据和存储器中的数据之间,维持数据一致性。非正式地,数据一致性可以指代:代理的高速缓存中的每一个有效数据线表示该数据的最新更新的值,并且在一致性协议分组中发送的数据可以表示在发送该数据的时间,该数据的最新更新的值。当在高速缓存或者在传输中不存在数据的有效复制时,该协议可以确保该数据的最新更新值位于存储器之中。第二,该协议可以提供用于请求的明确定义的提交点。用于读取的提交点可以指示该数据何时可用,而对于写入而言,它们可以指示所写入的数据何时是全局可看到的,并且将被随后的读取装载。该协议可以针对一致性存储器空间中的可高速缓存和不可高速缓存(UC)请求,支持这些提交点。

[0064] 在一些实现方式中,互连可以使用嵌入式时钟。可以将时钟信号嵌入在使用该互连发送的数据中。在时钟信号嵌入在数据中的情况下,可以省略不同的和专用的时钟通道。这可以是有用的,例如,其可以允许设备的更多引脚专用于数据传输,特别是在引脚空间非常宝贵的系统中。

[0065] 可以在互连的任一端上的两个代理之间建立链路。发送数据的代理可以是本地代理，而接收数据的代理可以是远程代理。这两个代理可以利用状态机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从链路层向电子前端发送微片。在一种实现方式中，控制路径包括状态机(其还称为链路训练状态机或者类似的术语)。状态机的动作和退出状态，可以取决于内部信号、定时器、外部信号或其它信息。事实上，这些状态中的一些(例如，一些初始化状态)可以具有用于提供超时值以退出某种状态的定时器。应当注意的是，在一些实施例中，检测指代：检测通道的两个径上的事件，但不是必须同时地检测。

[0066] 在状态机中规定的状态可以包括重置状态、初始化状态和操作状态等等其它分类和子分类。在一个实施例中，动作和退出可以是基于训练序列的交换。在一个实施例中，链路状态机运行在本地代理时钟域，并从一个状态转换到下一个状态与发送器训练序列边界相重合。可以利用状态寄存器来反映当前状态。例如，示例性状态可以包括：

[0067] 发送链路状态：链路状态。将微片发送到远程代理。该状态可以从阻塞链路状态进入，并可以基于某一事件(例如，超时)而返回到阻塞链路状态。在该发送链路状态(TLS)期间，发送器发送微片，而接收器接收微片。还可以从TLS退出到低功率链路状态。在一些实现方式中，TLS可以称为“L0”状态。

[0068] 阻塞链路状态：链路状态。发送器和接收器以统一的方式进行操作。可以存在对链路层微片进行推迟，同时将物理层信息传输到远程代理的定时状态。可以退出到低功率链路状态(或者基于设计方案而退出到其它链路状态)。在一个实施例中，阻塞链路状态(BLS)可以周期性地发生。该周期可以称为BLS间隔，并且其可以是定时的，并可以在慢速和操作速度之间不同。应当注意，可以周期性地阻塞链路层发送微片，使得可以发送某个长度的物理层控制序列(例如，在发送链路状态或者部分宽度发送链路状态期间)。在一些实现方式中，阻塞链路状态(BLS)可以称为L0控制或者“L0c”状态。

[0069] 部分宽度发送链路状态：省电链路状态。在一个实施例中，非对称部分宽度指代两个方向链路的每一个方向具有不同的宽度，其中在一些设计方案中，可以支持这些不同的宽度。虽然在部分宽度状态下不改变速度，但可以改变链路的通道的宽度。因此，潜在地按照不同的宽度来发送微片。部分宽度状态可以退出到其它链路状态，例如，基于接收的和发送的某些消息而退出到低功率链路状态，或者基于其它事件，部分宽度发送链路状态或者链路阻塞状态的退出。在一个实施例中，发送器端口可以以交错方式关闭空闲通道，以提供更佳的信号完整性(即，噪声减轻)。在链路宽度发生改变的时段期间，可以使用不可重试微片(例如，空微片)。相应的接收器可以丢弃这些空微片，并以交错方式来关闭空闲通道，以及在一个或多个结构中记录当前和先前通道映射。应当注意，状态和相关联的状态寄存器可以保持不变。在一些实现方式中，部分宽度发送链路状态可以称为部分L0或L0p状态。

[0070] 退出部分宽度发送链路状态：退出部分宽度状态。在一些实现方式中，可以使用阻塞链路状态，也可以不使用。在一个实施例中，发送器通过在空闲通道上发送部分宽度退出模式来对它们进行训练和去偏移，来发起退出。举一个例子，退出模式起始于EIEOS，其中EIEOS被检测和去抖动以发出该通道已准备开始进入完全发送链路状态，并可以以空闲通道上的SDS或快速训练序列(FTS)来结束的信号。在退出序列期间的任何失败(接收器动作，例如在超时之前去偏移没有完成)都停止微片传输到链路层，并发出重置，其通过在下一个阻塞链路状态发生时将链路重置来进行处理。SDS还可以将通道上的加扰器/解扰器初始化

成适当的值。

[0071] 低功率链路状态：其是更低功率状态。在一个实施例中，与部分宽度链路状态相比，该状态具有更低功率，这是由于在该实施例中，停止所有通道以及两个方向上的信令。发送器可以使用阻塞链路状态来请求低功率链路状态。这里，接收器可以对该请求进行解码，并使用ACK或NAK进行响应；否则可以触发重置。在一些实现方式中，低功率链路状态可以称为L1状态。

[0072] 转到图6，示出了用于描述一种示例系统的简化框图600，该示例系统包括能够被多个独立节点610a-610n中的每一个使用装载/存储技术进行访问的共享存储器605。例如，可以提供共享存储器控制器615，其可以接受系统上的各个节点610a-610n的装载/存储访问请求。可以使用同步动态随机存取存储器(SDRAM)、双列直插存储器模块(DIMM)和其它非易失性存储器(或易失性存储器)来实现共享存储器605。

[0073] 每个节点可以自身具有一个或多个CPU插槽，且还可以包括本地存储器，该本地存储器与系统中的其它节点的LD/ST访问保持隔离。节点可以使用一个或多个协议(其包括PCIe、QPI、以太网等等其它示例)，与系统上的其它设备(例如，共享存储器控制器615、连网控制器620、其它节点等等)进行通信。在一些实现方式中，可以提供共享存储器链路(SML)协议，通过该SML协议来支持低时延LD/ST存储器语义。例如，SML可以用于传送系统的各个节点610a-610n(通过共享存储器控制器615)对于共享存储器605的读和写。在一些实现方式中，SML可以采用结合图5的例子所描述的互连架构和协议的多个方面。

[0074] 举一个例子，SML可以是基于存储器访问协议，例如，可扩展存储器互连(SMI)第三代(SMI3)。或者，也可以使用其它存储器访问协议，例如，诸如完全缓存的DIMM(FB-DIMM)之类的事务型存储器访问协议、DDR事务型(DDR-T)等等其它示例。在其它实例中，SML可以是基于具有额外目录扩展的本机PCIe存储器读/写语义。由于进行了裁剪以高速缓存线路存储器访问，因此SML的基于存储器协议的实现方式可以提供带宽效率优势。虽然存在高性能设备间通信协议(例如，PCIe)，但这些协议的上层(例如，事务和链路层)可能引入时延，这使得在用于LD/ST存储器事务时(其包括涉及共享存储器605的事务)，整个协议的应用性能降低。诸如SMI3之类的存储器链路协议，可以具有提供更低时延访问的潜在另外优点，这是由于其旁路了另一协议栈(例如，PCIe)的大部分。因此，SML的实现方式可以使用SMI3或者在另一个协议的逻辑和物理PHY上运行的另一种存储器协议，例如，PCIe上的SMI3。

[0075] 如上所述，在一些实现方式中，可以提供共享存储器控制器(SMC)615，其包括用于对系统中的节点610a-610n的装载/存储请求进行处理的逻辑单元。SMC 615可以通过使用SML并将节点610a-610n连接到SMC 615的链路，来接收装载/存储请求。在一些实现方式中，可以将SMC 615实现为诸如专用集成电路(ASIC)之类的设备，其包括用于对节点610a-610n针对共享存储器资源的访问请求进行服务的逻辑单元。在其它实例中，SMC 615(以及共享存储器605)可以位于与节点610a-610n其中之一或多个(或者甚至全部)相分离的设备、芯片或电路板上。SMC 615还可以包括：用于协调各个节点的涉及共享存储器605的事务的逻辑单元。另外，SMC可以维持目录，以跟踪对于共享存储器605中包括的各个数据资源(例如，每个高速缓存线路)的访问。例如，数据资源可以处于共享访问状态(比如，其能够同时地被节点中的多个处理设备和/或I/O设备进行访问(如，装载或读取))、独占访问状态(比如，如果不是临时的话，被节点中的单一处理设备和/或I/O设备进行独占地保留(如，用于存储或

写入操作))、未高速缓存状态等等其它潜在示例。此外，虽然每一个节点可以具有对于共享存储器605的一个或多个部分的直接访问，但各个节点(例如，610a-610n)可以使用不同的寻址方案和值，其导致：第一节点根据第一地址值(例如，在指令中)来引用相同的共享存储器数据，而第二节点根据第二地址值来引用相同的数据。SMC 615可以包括用于允许SMC 615解释各个节点的各种访问请求的逻辑单元(其包括将节点的地址映射到共享存储器资源的数据结构)。

[0076] 另外，在一些情况下，共享存储器的一些部分(例如，某些划分、存储器块、记录、文件等等)可以服从某些许可、规则和分配，使得(例如，SMC 615)只允许节点610a-610n中的一部分访问相应的数据。事实上，可以向系统的节点610a-610n的相应子集(在一些情况下，不同的子集)分配每个共享存储器资源。这些分配可以是动态的，且SMC 615可以修改这些规则和许可(例如，按需地、动态地等等)，以适应可应用于共享存储器605的给定部分的新的或者改变的规则、许可、节点分配和所有权。

[0077] SMC 615的示例还可以跟踪各个事务，这些事务涉及系统中的节点(例如，610a-610n)访问一个或多个共享存储器资源。例如，SMC 615可以跟踪每一个共享存储器605事务的信息，其包括：该事务中涉及的节点的标识、该事务的进度(例如，无论其是否已完成)等等其它事务信息。这可以准许传统的分布式存储器架构的面向事务方面中的一些方面应用于本文所描述的改进的多节点共享存储器架构。另外，(例如，SMC)可以使用事务跟踪来辅助维持或者实施每一个相应节点的不同的和独立的故障域。例如，SMC可以在其内部数据结构中(其包括在存储器中)，维持每一个进行中事务的相应节点ID，以及使用该信息来实施访问权利，并维持每一个节点的各个故障域。因此，当这些节点其中之一停止工作时(例如，由于关键错误、触发的恢复序列或者其它故障或事件)，仅仅该节点和其涉及共享存储器605的事务被中断(例如，被SMC转出)，剩余节点的涉及共享存储器605的事务则独立于所述另一个节点的故障而继续。

[0078] 系统可以包括多个节点。另外，一些示例性系统可以包括多个SMC。在一些情况下，节点可以能够访问其没有直接连接到的远程SMC的共享存储器(即，该节点的本地SMC通过一个或多个SML链路跳来连接到该远程SMC)。远程SMC可以处于相同的电路板中，也可以处于不同的电路板中。在一些情况下，这些节点中的一些可以在系统外(例如，板外或片外)，但仍然访问共享存储器605。例如，一个或多个系统外节点可以使用遵循SML的链路来直接连接到SMC，等等其它示例。另外，包括有它们自己的SMC和共享存储器的其它系统也可以与SMC 610相连接，以便将存储器605的共享延伸到例如另一个电路板上包括的节点，其中所述另一个电路板通过SML链路与连接到该SMC的另一SMC对接。另外，可以对网络连接隧道化，以进一步将访问延伸到其它板外或片外节点。例如，SML可以在通信地耦合图6的示例系统与另一个系统的以太网连接(例如，其通过网络控制器620来提供)上进行隧道化，其中，该另一个系统还可以包括一个或多个其它节点，并允许这些节点也获得对于SMC 615和因此的共享存储器605的访问，等等其它示例。

[0079] 再举一个例子，如图7的简化框图700中所示，准许多个独立的节点根据LD/ST存储器语义进行共享访问的改进的共享存储器架构，可以灵活地允许提供多种不同的多节点系统设计。可以分配多个节点的各种组合，来共享该示例系统中提供的一个或多个共享存储器块的多个部分。例如，图7的例子中所示出的另一个示例性系统可以包括被实现成例如单

独的管芯、板、芯片等等的多个设备705、710、715、720，每一个设备包括一个或多个独立的CPU节点(例如，610a-610h)。每一个节点可以包括其自己的本地存储器。所述多个设备705、710、715、720其中之一或多个，还可以包括可以由系统的节点610a-610h中的两个或更多进行访问的共享存储器。

[0080] 图7中所示出的系统是用于描绘可以通过改进的共享存储器架构来实现的变化性中的一些的例子，例如，本文所示出和描述的。例如，设备A 705和设备C 715中的每一个可以包括各自的共享存储器元件(例如，605a、605b)。因此，在一些实现方式中，不同的设备上的每一个共享存储器元件还可以包括各自的共享存储器控制器(SMC)615a、615b。节点610a-610h的各种组合可以通信地耦合到每一个SMC(例如，615a、615b)，其允许这些节点访问相应的共享存储器(例如，605a、605b)。举例而言，设备A 705的SMC 615a可以使用支持SML的直接数据链路，来连接到设备A上的节点610a、610b。另外，另一个设备(例如，设备C 715)上的另一个节点610c也可以通过从节点610c(和/或其设备715)到SMC 615a的直接硬连线连接(其支持SML)，来访问共享存储器605a。还可以使用间接的、基于网络的或者其它这种连接来允许远程设备或板外设备(例如，设备D 720)的节点(例如，610f-610h)利用传统协议栈与SMC 615a对接，从而也能访问共享存储器605a。例如，可以通过耦合设备A和设备D的以太网、无线带宽或者其它连接，来建立SML隧道725。虽然建立和维持该隧道可能引入一些另外的开销和时延，但与在其它欠软件管理的物理连接上运行的SML相比，该SML隧道725(当建立时)可以操作为其它SML信道，并允许节点610f-610h通过SML与SMC 615a对接和访问共享存储器605a，如同通过SML链路与SMC进行通信的任何其它节点可以一样。例如，SML信道中的分组的可靠性的排序可以由系统中的连网部件来实施，也可以在SMC之间进行端到端实施。

[0081] 在其它示例中，与拥有共享存储器(例如，605a)的特定部分不同的设备上的节点(例如，615d、615e)可以通过直接连接到另一个SMC(例如，615b)，来间接地连接到相应的SMC(例如，SMC 615a)，其中所述另一个SMC自身(例如，使用SML链路)耦合到该相应的SMC(例如，615a)。链接两个或更多SMC(例如，615a、615b)可以有效地扩展可用于系统上的节点610a-610h的共享存储的量。例如，在一些实现方式中，依靠图7的例子中的SMC 615a、615b之间的链路，这些节点(例如，610a-610c、610f-610h)中的能够通过SMC 615a来访问共享存储器605a的任何节点，也可以通过SMC 615a和SMC 615b之间的连接来潜在地访问共享存储器605b。同样，在一些实现方式中，间接地访问SMC 615b的每一个节点也可以通过SMC 615a和SMC 615b之间的连接来访问可共享存储器605a，等等其它潜在示例。

[0082] 如上所述，一种改进的共享存储器架构可以包括基于存储器访问协议(例如，SMI3)的低时延链路协议(即，SML)，提供该协议以有助于涉及共享存储器的装载/存储请求。但是可以配置传统的SMI3和其它存储器访问协议以用于单一节点中的存储器共享，SML可以将存储器访问语义扩展到多个节点，以允许多个节点之间的存储器共享。此外，SML还可以潜在地用在任何物理通信链路上。SML可以使用支持LD/ST存储器语义的存储器访问协议，其中该存储器访问协议覆盖在适于互连不同的设备(和节点)的物理层(和相应的物理层逻辑单元)上。另外，SML的物理层逻辑单元可以提供无分组丢弃和差错重试功能等等其它特征。

[0083] 在一些实现方式中，可以通过将SMI3覆盖在PCIe PHY上来实现SML。可以在流控制

和其它特征之前提供SML链路层(例如,替代传统的PCIe链路层),且SML链路层有助于更低时延的存储器访问,如同传统的CPU存储器访问架构中的特性。举一个例子,SML链路层逻辑单元可以在共享存储器事务和其它事务之间进行复用。例如,SML链路层逻辑单元可以在SMI3和PCIe事务之间复用。例如,SMI3(或另一种存储器协议)可以覆盖在PCIe(或者另一种互连协议)之上,使得链路可以在SMI3和PCIe事务之间动态地切换。在一些实例中,这允许传统的PCIe业务有效地共存于与SML业务相同的链路上。

[0084] 转到图8的简化框图800,示出了一种示例性计算节点(例如,610)的特征。计算节点610可以使用通过将SMI3(或另一种存储器协议)覆盖在遵循PCIe(例如,PCIe 3.0)的电子物理链路上所实现的示例性SML,与共享存储器控制器芯片615进行通信。节点610可以包括多列分层的栈,其中该多列分层的栈包括一列用于存储器访问协议(例如,SIM3),而另一列用于另一种互连协议(例如,诸如PCIe的串行通用互连协议)。每一列可以具有其自己的链路层(例如,805、810),以及位于每个相应的链路层805、810之上的其它层,包括事务、路由、协议和/或一致性层(没有示出)。在该示例中,存储器访问协议可以使用采用192比特微片的链路层。可以在另一种协议的PHY上(在该情况下,使用本机128比特编码方案(如,128b/130b)的遵循PCIe的PHY),来发送这些微片。因此,节点610的PCIe PHY逻辑单元可以包括PCIe逻辑PHY 815、PCIe单板820(例如,坐落在数字域和模拟域之间的额外层)和PCIe模拟前端(AFE)825,等等其它示例。

[0085] 转换逻辑单元830可以将根据存储器访问协议所发送的192比特微片数据转换成128比特PCIe有效载荷,以便在PCIe物理链路上携带。同样,可以(例如,使用逻辑单元835)将128比特有效载荷转换回192比特微片数据。在该示例中,PCIe列可以包括模拟转换逻辑单元840、845,以便将256b PCIe 3.0有效载荷转换成128比特数据。可以将垫层电路850提供为链路层和PCIe物理层之间的接口。垫层850可以控制向物理层发送哪个链路层数据。在一些实现方式中,可以选择性地配置垫层850(例如,通过设置节点610的硬件中的引信),使得其只允许使用多个列中的单一列(以及链路层805、810中的单一链路层)。例如,可以对垫层850进行设置,使得节点实现本机PCIe端口(即,只使用PCIe链路层805)或者SML端口(即,覆盖在PCIe物理层之上的存储器访问协议,并且只使用存储器访问协议链路层810)。在其它实例中,垫层850可以实现对来自任一列的数据进行动态复用的功能,其造成数据流在PCIe数据模式和SML数据模式之间进行切换。

[0086] 图9示出了将存储器访问协议覆盖在另一种不同的串行互连协议(例如,PCIe)上的例子。在图7A的例子中,可以使用物理层成帧令牌来用信号发送存储器访问和互连协议业务之间的转换。成帧令牌(或者“令牌”)可以是指定或者隐含在与该令牌相关联的数据流中将包括的符号的数量的物理层数据封装。因此,成帧令牌可以标识流开始,以及隐含着其将结束的位置,因此也可以用于标识下一个成帧令牌的位置。数据流的成帧令牌可以位于数据流的第一数据块的第一通道(例如,通道0)的第一符号(符号0)中。可以使用现有符号(如用于传输该业务的物理层的本机协议中所规定的)来实现该成帧令牌。例如,在PCIe的例子中,可以规定五种成帧令牌,其包括:TLP业务的起始(STP)令牌、数据流的结束(EDS)令牌、结束坏(EDB)令牌、DLLP的起始(SDP)令牌和逻辑空闲(IDL)令牌。

[0087] 在图9的例子中,可以通过将SMI3或另一种数据访问协议覆盖在PCI上来实现SML,并且可以对标准PCIe STP令牌进行重用和编码以规定新的STP令牌,其中该新的STP令牌标

识SMI3(替代TLP业务)将在该链路的通道上开始。在端口在SML中静态工作的情况下,可以对STP令牌进行编码以指示该业务是SML。在可以将PCIe和SML业务复用在该链路上的情况下,STP令牌可以指示PCIe业务的突发和SML业务的突发之间的转换,等等其它示例。

[0088] 举一个例子,可以对标准PCIe STP令牌的保留位进行修改,以规定SML业务STP令牌(例如,"SML令牌")的起始。例如,规定的所允许分组或有效载荷长度(例如,所允许的PCIe有效载荷的长度)集合可以被规定用于主机协议。STP令牌可以包括长度字段,且该长度字段可以允许落在针对PCIe(或者物理层的另一种主机协议)所规定的允许长度集合之外的编码方式。在一种实现方式中,利用不允许的值或者另一种规定的值(其落在所允许的长度之外)对长度字段进行编码,可以用于将STP令牌标识为SML令牌。

[0089] 返回到图9的例子,示出了动态SML/PCIe复用,其中SML使用PCIe PHY协议上的SMI3。例如,可以规定同步报头数据,以遵循针对传统PCIe 128b/130b编码所指定的编码方式。例如,在915a-c处,接收到具有值10b的同步报头,其指示数据块即将到来。当接收到PCIe STP(例如,920)时,预期PCIe TLP有效载荷,并相应地处理数据流。与PCIe STP 920中标识的有效载荷长度相一致,PCIe TLP有效载荷可以使用分配的全部有效载荷长度。本质上,可以在TLP有效载荷结束之后的数据块中的任何时间,接收另一个STP令牌。例如,在925处,可以接收SMI3 STP,其用信号表明从PCIe TLP数据转换到SMI3微片数据。随后,例如只要一识别PCIe分组数据结束,就可以发送SMI3 STP。

[0090] 关于PCIe TLP数据,SMI3 STP 925可以规定之后的SMI3微片有效载荷的长度。例如,SMI3数据的有效载荷长度可以对应于将发送的SMI3微片的数量(或者相应的DW的数量)。与有效载荷长度相对应的窗口(例如,其结束于通道3的符号15处)可以因此在这些通道上进行规定,其中在这些通道中只发送SMI3数据。当该窗口结束时,可以发送其它数据,例如,另一个PCIe STP重新开始发送TLP数据或其它数据(例如,有序集数据)。例如,如图9的例子中所示,在SMI3 STP令牌925所规定的SMI3数据窗口的结束之后发送EDS令牌。EDS令牌可以用信号发送数据流的结束,并隐含着有序集块将跟在其后,如同图9的例子中的情况。发送同步报头940,其被编码成01b以指示将发送有序集块。在该情况下,发送PCIe SKP有序集。可以周期性地或者根据设定的间隔或窗口来发送这些有序集,使得可以执行各种PHY层级任务和协调,其包括:初始化比特对齐,初始化符号对齐,交换PHY参数,补偿两个通信端口的不同比特速率,等等其它示例。在一些情况下,可以发送托管的有序集,以通过相应的SMI3 STP令牌来中断针对SMI3微片数据所指定的规定窗口或数据块。

[0091] 虽然在图9的例子中没有显式地示出,但也可以使用STP令牌来将该链路上的SMI3微片数据转换成PCIe TLP数据。例如,在规定的SMI3窗口结束之后,可以发送PCIe STP令牌(例如,类似于令牌920)以指示下一个窗口用于发送指定数量的PCIe TLP数据。

[0092] 在一些实施例中,存储器访问微片(例如,SMI3微片)可以在大小上变化,其使得很难针对存储器访问有效载荷,先验地预测有多少数据保存在相应的STP令牌(例如,SMI3 STP令牌)中。举例而言,如图9中所示,SMI3 STP 925可以具有指示在SMI3 STP 925之后预期将有244字节的SMI3数据的长度字段。但是,在该例子中,仅仅十个微片(例如,SMI3微片0-9)准备在该窗口期间进行发送,且这十个SMI3微片只使用244个字节中的240个。因此,剩下了四个(4)字节的空带宽,且利用IDL令牌来填充这些字节。当PCIe TLP数据进行排队并等待SMI3窗口关闭时,这可能是特别不理想的。在其它情况下,为了发送SMI3微片所提供的

窗口可能不足够用于发送已为该通道准备好的SMI3数据的量。可以使用仲裁技术来确定如何在链路上共存的SMI3和PCIe TLP数据之间进行仲裁。此外，在一些实现方式中，可以动态地修改SMI3窗口的长度，以有助于该链路的更高效使用。例如，仲裁或其它逻辑单元可以监测如何充分利用规定的SMI3窗口，来判断是否可以针对该通道所预期的SMI3的量(以及与PCIe TLP业务进行竞争)，对规定的窗口长度进行更佳的优化。因此，在这些实现方式中，可以根据应当向SMI3微片数据分配的链路带宽的量(例如，相对于包括TLP、DLLP和有序集数据的其它PCIe数据)，动态地调整SMI3 STP令牌的长度字段值(例如，在不同的值之间)，等等其它示例。

[0093] 串行输入/输出(I/O)操作可能在操作时消耗大量的功率。因此，当在链路的两个方向中没有未决的业务时，期望使链路进入低功率或者空闲状态(例如，L1状态)。这对于PCIe或SML模式操作中的任意一个或二者来说都是成立的。此外，在SML模式中使用的存储器访问协议可以规定进入和退出低功率模式，其与主机PHY层协议中(例如，在PCIe中)规定的不同。此外，该存储器访问协议所使用的状态机也与管理物理层的其它协议中使用的状态机不同。

[0094] 举例说明，在存储器访问协议的一种实现方式中，存储器访问协议的链路层可以提供用于物理层执行控制任务的机会，这些控制任务可以包括从发送链路状态(TLS)转换到低功率或者L1状态。例如，当在涉及使用外部协议的PHY的SML上下文之外使用时，存储器访问协议可以提供控制状态(L0c)、或者阻塞链路状态、可以在其中发送物理层请求和响应的窗口。举例而言，可以提供控制状态L0c，以有助于与这些控制任务有关的消息传送。可以将L0c状态提供成TLS中的周期性窗口，以便允许在使用链路层发送的微片流之间发送物理层控制消息。例如，如图10中所示出的例子里所表示的，可以将L0状态细分成L0c间隔。每一个L0c间隔以L0c状态或窗口(例如，1005)开始，在此期间可以发送物理层控制代码和其它数据。L0c间隔的剩余部分(例如，1010)可以专用于发送微片。可以通过例如一个或多个设备的BIOS或者另一个基于软件的控制器等等其它示例，对L0c间隔的长度和每一个间隔中的L0c状态进行编程性地规定。与L0c间隔的剩余部分相比，L0c状态可以指数级别地更短。例如，举一个例子，L0c可以是8UI，而L0c间隔的剩余部分是4KUI的量级等等其它示例。这允许在基本无需扰乱或者浪费链路数据带宽的情况下，在其中发送相对较短的、预先规定的消息的窗口。

[0095] L0c状态消息可以在物理层层级传送多种状况。举一个例子，一个设备可以发起链路或通道的重置(例如，基于超过特定的阈值量的比特错误或者其它错误)。还可以在L0c窗口(例如，在先的L0c窗口)中传送这些错误。还可以利用L0c状态来实现其它带内信令，例如，用于辅助或者触发其它链路状态之间的转换的信令。举一个例子，可以利用L0c消息来将链路从活动L0状态转换成待机或低功率状态(例如，L1状态)。如图11A的简化流程图1100a中所示，可以使用特定的L0c状态来传送L1进入请求(例如，1110)。当设备(或者设备上的代理)等待该请求1110的确认时，可以发送另外的微片(例如，1120、1130)。该链路上的另一设备可以发送该确认(例如，1140)。在一些示例中，还可以在L0c窗口中发送该确认。在一些实例中，可以在L1请求1110的接收/发送之后的下一个L0c窗口中发送该确认。可以利用定时器来同步每一个设备处的L0c间隔，请求设备可以基于确认1140在下一个L0c窗口时发送的标识，将确认1140识别为请求1110的确认(例如，而不是独立的L1进入请求)等等其

它示例。在一些实例中，可以通过与L1进入请求1110中所使用的代码不同的L0c代码，来传送确认。在其它实例中，确认1140可以包括在请求1110中使用的L1进入请求代码的回应等等其它示例。此外，在替代的示例中，例如图11B中所示，可以在L0c窗口中传送非确认信号或者NAK 1145，其造成尽管L1进入请求1110，该链路仍然处于链路发送状态。

[0096] 图10、11A和图11B的例子只是根据SML的实现方式所使用的存储器访问协议或其它协议的低功率转换握手的一个例子。在其它实施例中，可以利用其它机制来实现从活动状态向低功率或空闲状态的转换。但是，在存储器访问协议覆盖在另一种协议的物理层之上的实例中(例如，在SML的实现中，此时，存储器访问协议(如，SMI3)覆盖在诸如PCIe的另一种串行互连协议之上)，可能并不支持这些机制。事实上，PHY所使用的主机协议可以不同地实现从活动状态向低功率状态的转换，其使存储器访问协议的发起转换到低功率状态的能力复杂化。在静态地实现SML的实施方式中，这可能是特别麻烦(即，其是其它协议的物理层上的唯一业务)。

[0097] 为了说明这些潜在的差别，将图10、11A和11B中所示出的用于进入低功率状态的机制与PCIe中的进入到低功率L1状态的表示(图12中)进行比较。例如，第一(上游)部件1205可以向下游部件1210发送配置写请求1215。下游部件1210可以基于接收到该配置写请求1215，开始从发送链路状态转换到(例如，在1220处)低功率状态(L1)的处理。下游部件1210可以发送针对该配置写请求的完成1225，并累积新的事务层分组(TLP)的块调度和最小信用度(在1230处)。下游部件1210可以进行等待(在1235处)，以便接收针对最后TLP的确认(ACK)。一旦下游部件的所有TLP都被确认，则下游部件可以开始发送1240 Enter_L1数据链路层分组(DLLP)(以指示进入到L1状态)，直到其从上游部件1205接收到响应。

[0098] 在接收到Enter_L1 DLLP时，上游部件1205可以(在1245处)阻塞新的TLP传输的调度。随后，上游部件1205可以进行等待，直到其接收到其先前已发送的最后TLP的链路层确认。下游部件1210可以(在1250处)等待针对所发送的Enter_L1 DLLP的确认DLLP(例如，Request_ACK)。当上游部件1205(在1255处)接收到针对其最后TLP的ACK时，上游部件1205可以响应于所接收的Enter_L1 DLLP，重复地发送Request_ACK DLLP 1260。一旦下游部件1210已在其接收通道上捕捉到该Request_ACK DLLP(其用信号表明上游部件1205对转换到L1请求进行了确认)，则其随后(在1265处)禁用DLLP传输，并使上游针对的物理链路进入电子空闲状态。此外，随着上游部件1205上的接收通道进入电子空闲状态，上游部件1205停止发送Request_ACK DLLP，(在1270处)禁用DLLP传输，并将其发送通道带入电子空闲，从而完成该链路向L1的转换。在一些实例中，链路的另一端的部件可以退出L1。

[0099] 如上所述，可期望提供主机协议或者SML协议，以便用于发起转换到低功率链路状态。但是，在一些实现方式中，SML所使用的存储器访问协议可能不能向PHY层委托进入低功率状态的协商，例如，在图10、11A和图11B的例子中所示出和描述的。因此，在SML的一些实现方式中，可以提供L1进入方案，该方案允许在用于促进SML的令牌中，对用于进入低功率状态的存储器访问协议的本机编码进行隧道化。例如，举一个例子，SML所使用的存储器访问协议可以在阻塞链路状态中或者控制窗口中，发送链路状态转换请求(以及确认)。可以将这些相同的编码包括在该令牌的保留或未使用字段中，使得在不影响主机(例如，PCIe)协议或者分层栈的情况下，将它们传送给下游部件。在一种实现方式中，可以利用指示进入低功率状态的请求(或ACK/NAK)的数据值，对STP令牌(例如，SML令牌)的字段进行编码。

[0100] 转到图13，示出了STP令牌1305的表示。STP令牌1305可以包括几个字段，其包括TLP序列号字段1310、以及标识在该STP之后的存储器访问协议(例如，SMI3)有效载荷的长度(根据微片的数量)的长度字段1315。如上所述，在一些实现方式中，可以使用该长度字段来指示后面的数据是根据主机互连协议(例如，PCIe)还是根据存储器访问协议(例如，SMI3)。例如，可以针对TLP数据，规定一个或多个标准有效载荷长度。在一些实现方式中，可以对SMI3数据进行规定以包括固定数量的微片，或者在其它情况下，SMI3数据可以具有可变数量的微片，在后一情况下，针对SMI3微片的数量的长度字段变成可以忽视的长度。此外，可以将针对SMI3 STP的长度字段，规定成不同于所规定的TLP有效载荷长度其中之一的长度。因此，举一个例子，可以基于在STP长度字段中存在的非TLP长度值，来识别SMI3 STP。

[0101] 继续图13的例子，当STP是SML STP令牌时，可以保留TLP序列号字段1310(或者不使用)。可以对TLP序列号字段1310进行编码，以指示请求转换到L1，指示针对进入到L1的请求的ACK或NAK，或者指示在SML STP令牌中不包括操作，以及等等其它可能的编码方式。例如，在一种示例性实现方式中，可以规定一组值，以便用于存储器访问协议的本机版本中的控制窗口中，可以在SML STP令牌的保留字段中，对这些相同的值进行编码。例如，在一种实现方式中，使用SMI3作为存储器访问协议。

[0102] 表1

[0103]

编码	含义
x333	SMI3 QNOP: 无操作
x999	SMI3 QL1: 请求进入低功率 L1 状态
X666	SMI3 QL1n: 针对进入 L1 的请求的否定确认(NAK)
其它	保留

[0104] 继续表1的例子，可以规定一种握手，使得在SML STP令牌中编码的进入L1的请求，可以利用编码以进行回应的SML STP令牌来进行应答，并对进入L1的请求(例如，QL1)进行确认，或者利用否定确认消息(例如，QL1n)进行应答。在后一情况下，可以周期性地重复该过程，直到实现确认握手。SML STP令牌中的没有被编码成用于指示进入L1的请求(或响应)或者响应的指定字段，可以利用替代的编码方式或者无操作编码方式来进行编码，等等其它示例。

[0105] 图14是根据至少一个例子，示出进入L1的握手的例子的流程图。在该例子中，垫层850可以用于促进对PCIe上的存储器访问协议的低功率状态进入请求进行隧道化。例如，功率管理代理(PMA)1401、1402可以异步地启用(例如，在1405、1415处)相应的链路层以进入低功率。随后，(在1410、1420处)相应的链路层(LL)810a、810b可以与垫层(分别805a、805b)进行通信，以发起进入到低功率状态(例如，L1)。例如，垫层805a、805b可以根据存储器访问协议来接收L1进入信号1410、1420。垫层805a、805b可以包括用于根据存储器访问协议，识别EL 1c数据是进入低功率状态L1的请求的逻辑单元。垫层805b可以利用在存储器访问协议中规定的低功率进入请求编码方式，对PCIe令牌的字段(例如，基于PCIe的STP令牌的TLP

序列号字段)进行编码。例如,存储器访问协议可以使用阻塞链路状态(L0c)来发送这些编码,并且垫层805a、b可以用于利用基于PCIe的令牌字段,对L0c低功率进入握手进行隧道化(在1425处)。因此,当垫层805a接收到编码的令牌时,其可以向链路层810a发送信号(在1430处)以进入L1。垫层805a还可以发送L1进入请求的确认(在1425处),并且在垫层805b接收到ACK时,其可以(在1435处)向其链路层810b发送信号以进入L1。此外,在接收到该确认时,垫层805b还可以(在1440处)向物理层815b发送信号,以发起PCIe物理层L1进入握手。因此,PHY 815b可以发送一个或多个电子空闲有序集(EIEOS),且PHY 815a进行回应以造成PCIe物理层进入电子空闲1450。可以通过信号1455、1460,针对相应的垫层805a、805b进行相同的确认。

[0106] 继续图14的例子,在链路进入到L1之后的一些实例,链路层810b(如通过软件所提示的)可以发起从低功率L1状态中退出,以重新进入发送链路状态(例如,L0或L0p)。例如,可以将L1退出信号1465发送给垫层805b并由其进行解码,且垫层805b可以(在1470处)向PHY 815b发送信号以退出低功率状态,并开始将该链路配置成发送链路状态。例如,为了对链路进行去抖动、同步和配置,可以(在1475处)交换有序集(OS)和训练序列(TS)。在链路训练结束之后(例如,通过1475),垫层805a、805b可以识别该物理层正处于发送状态L0。随后,垫层805a、805b可以向链路层810a、810b通知已进入L0。在一些情况下,当垫层805a、805b在PCIe物理层上提供存储器访问协议数据和PCIe数据的动态复用时,垫层805a、805b可以向存储器访问协议和PCIe链路层两者传送进入到L0(或L1),等等其它示例。

[0107] 在GPIO协议的物理层上,支持通用I/O(GPIO)协议(如,PCIe)和存储器访问协议(如,SMI3)之间的动态切换的实现方式中,垫层可以在向公共物理层发送信号以进入电子空闲之前,确定两个协议栈准备好进入低功率链路状态(例如,L1)。例如,GPIO协议链路层(在带宽的其一部分期间)或者存储器访问协议可以(根据其相应的协议)独立地请求进入低功率链路状态。但是,当一个列尝试进入低功率状态时,另一列可能具有等待在活动链路上发送的数据。在一些实现方式中,动态垫层可以具有在栈之间仲裁带宽需求的能力。因此,在一些实现方式中,当对来自两个不同的协议栈的数据进行复用时,垫层可以对这两个列进行监测,以便在同意其它列的进入低功率状态的请求之前,确保这些列其中之一在即将到来的带宽划分中没有数据要发送(例如,在STP令牌之后)。在一些实现方式中,垫层可以将PHY已进入L1的确认合成到请求进入L1的列,同时抑制向该PHY进行将触发实际进入到L1的请求,直到垫层确定这样做将不会影响等待从另一列发送的数据。例如,在另一第二列已请求进入L1,并认为链路实际处于L1之后,可以准许第一列使用物理链路。在这些实例中,只要第一列完成发送数据,垫层就可以向物理层发送信号以使得该链路进入空闲。在其它实例中,当该链路保持活动时,第二列的链路层可以重新唤醒,并尝试重新激活已经活动的链路(即,其用于发送另一列的数据)。在这些实例中,垫层可以从第二列的链路层接收数据,以发起该链路的激活。在这些实例中,垫层可以将链路的重新唤醒的确认,合成到链路层第二列中。在实践中,由于该链路已经进行了训练并是可操作的,因此不执行如与物理层相关的动作,且垫层可以响应于来自第二列的链路层的用于使该链路“退出”低功率状态的信号,简单地向第二列发送该链路是活动的确认信号。在这些实例中,不仅在两个不同的协议的数据之间可以共享信号物理链路的带宽,而且还可以支持这些不同协议的低功率状态进入范式。

[0108] 应当注意的是，虽然有时在诸如PCIe和SMI3之类的某些协议的背景下，描述上面的原理和示例中的大部分。但是，应当理解的是，这些协议仅仅只是作为例子，在SML的实现时可以使用任何串行GPIO协议和存储器访问协议。事实上，本文所描述的原理、解决方案和特征可以等同地适用于其它协议和系统。此外，可以在系统中应用上面的解决方案的组合，其包括针对链路和其相应逻辑单元(如本文所描述的)的逻辑和物理增强的组合，等等其它示例。

[0109] 应当注意，上面所描述的装置、“方法”和系统可以在如上所述的任何电子设备或系统中实现。举一些特定的说明，下面的附图提供了用于利用如本文所描述的发明内容的示例性系统。随着更详细地描述下面的系统，公开、描述了多种不同的互连，并根据上面的讨论重新回访多种不同的互连。显而易见的是，上面所描述的改进可以应用于这些互连、结构或架构中的任何一种。

[0110] 参见图15，描述了用于包括多核处理器的计算系统的框图的实施例。处理器1500包括任何处理器或处理设备，例如，微处理器、嵌入式处理器、数字信号处理器(DSP)、网络处理器、手持处理器、应用处理器、协同处理器、片上系统(SOC)或者用于执行代码的其它设备。在一个实施例中，处理器1500包括至少两个内核(内核1501和1502)，它们可以包含非对称内核或者对称内核(所示出的实施例)。但是，处理器1500可以包括任意数量的处理元件，它们可以是对称的，也可以是非对称的。

[0111] 在一个实施例中，处理元件指代用于支持软件线程的硬件或逻辑单元。硬件处理元件的例子包括：线程单元、线程槽、线程、处理单元、上下文、上下文单元、逻辑处理器、硬件线程、内核和/或能够保持处理器的状态(例如，执行状态或架构状态)的任何其它元件。换言之，在一个实施例中，处理元件指代能够独立地与代码(例如，软件线程、操作系统、应用或其它代码)相关联的任何硬件。物理处理器(或处理器插槽)通常指代集成电路，其潜在地包括任意数量的其它处理元件(例如，内核或硬件线程)。

[0112] 通常，内核指代能够维持独立的架构状态的位于集成电路上的逻辑单元，其中每一个独立维持的架构状态与至少一些专用执行资源相关联。与内核相比，硬件线程通常指代能够维持独立的架构状态的位于集成电路上的任何逻辑单元，其中独立维持的架构状态共享对于执行资源的访问。如图中所观察的，当某些资源是共享的，而其它资源专用于架构状态时，内核和硬件线程的术语之间的线路重叠。通常，操作系统将内核和硬件线程视作为个体的逻辑处理器，其中操作系统能够单独地调度每个逻辑处理器上的操作。

[0113] 如图15中所示，物理处理器1500包括两个内核(内核1501和1502)。这里，将内核1501和1502视作为对称内核，即，具有相同配置、功能单元和/或逻辑单元的内核。在另一个实施例中，内核1501包括无序处理器内核，而内核1502包括有序处理器内核。但是，内核1501和1502可以从例如下面的任何类型的内核中单独地选择：本机内核、软件管理内核、适于执行本机指令集架构(ISA)的内核、适于执行转换的指令集架构(ISA)的内核、协同设计的内核或者其它已知的内核。在异构内核环境(即，非对称内核)下，可以利用一些形式的转换(例如，二进制转换)来调度或执行一个或多个内核上的代码。为了进一步讨论起见，下面将进一步详细描述内核1501中所示出的功能单元，内核1502中的单元在所描述的实施例中以类似的方式进行操作。

[0114] 如上所述，内核1501包括两个硬件线程1501a和1501b，它们还可以称为硬件线程

槽1501a和1501b。因此，在一个实施例中，诸如操作系统之类的软件实体将处理器1500潜在地视作为四个单独的处理器，即，能够并发地执行四个软件线程的四个逻辑处理器或处理元件。如上文所提到的，第一线程与架构状态寄存器1501a相关联，第二线程与架构状态寄存器1501b相关联，第三线程可以与架构状态寄存器1502a相关联，以及第四线程可以与架构状态寄存器1502b相关联。这里，架构状态寄存器(1501a、1501b、1502a和1502b)中的每一个可以称为处理元件、线程槽或线程单元，如上所述。如图所示，架构状态寄存器1501a复制在架构状态寄存器1501b中，所以能够存储逻辑处理器1501a和逻辑处理器1501b的各自架构状态/上下文。在内核1501中，也可以针对线程1501a和1501b，复制其它更小的资源(例如，分配器和重命名器块1530中的重命名逻辑单元和指令指针)。诸如重新排序/引退单元1535中的重新排序缓冲区、ILTB 1520、装载/存储缓冲区和队列之类的一些资源，可以通过划分来进行共享。诸如通用内部寄存器、基于页表的寄存器、低层级数据高速缓存和数据TLB 1515、执行单元1540、以及乱序单元1535的一部分之类的其它资源，则潜在地完全共享。

[0115] 通常，处理器1500包括其它资源，这些资源可以是完全共享的，通过划分来共享，或者被处理元件专用/专用于处理元件。在图15中，示出了具有处理器的示例性逻辑单元/资源的纯粹示例性处理器的实施例。应当注意，处理器可以包括或者省略这些功能单元中的任何单元，以及包括没有示出的任何其它已知的功能单元、逻辑单元或固件。如上所述，内核1501包括简化的、代表性无序(000)处理器内核。但在不同的实施例中，也可以使用有序处理器。000内核包括：用于预测要执行/采用的分支的分支目标缓冲区1520和用于存储针对指令的地址转换条目的指令转换缓冲区(I-TLB)1520。

[0116] 内核1501还包括耦合到获取单元1520的解码模块1525，以便对获取的元素进行解码。在一个实施例中，获取逻辑单元包括分别与线程槽1501a和1501b相关联的各个定序器。通常，内核1501与规定/指定在处理器1500上可执行的指令的第一ISA相关联。通常，作为第一ISA的一部分的机器代码指令，包括该指令的一部分(其称为操作码)，其引用/指定要进行执行的指令或操作。解码逻辑单元1525包括根据其操作码来识别这些指令，并以流水线方式传递这些解码的指令来进行如第一ISA所规定的处理的电路。例如，如下面所进一步详细讨论的，在一个实施例中，解码器1525包括设计用于或者适于识别特定的指令(例如，事务指令)的逻辑单元。作为解码器1525的识别的结果，架构或内核1501采取特定的、预先规定的动作来执行与适当指令相关联的任务。重要的是要注意，本文所描述的任务、块、操作和方法中的任何一个可以是响应于单条指令或多条指令来执行，这些指令中的一些可以是新指令或旧指令。应当注意，在一个实施例中，解码器1526识别相同的ISA(或者其子集)。或者，在异构内核环境下，解码器1526识别第二ISA(不同的ISA或第一ISA的子集)。

[0117] 举一个例子，分配器和重命名器块1530包括用于保留资源的分配器，例如，用于存储指令处理结果的寄存器文件。但是，线程1501a和1501b潜在地能够乱序执行，其中分配器和重命名器块1530还保留其它资源(例如，重新排序缓冲区)以跟踪指令结果。单元1530还可以包括寄存器重命名器，以将程序/指令参考寄存器重命名成处理器1500内部的其它寄存器。重新排序/引退单元1535包括诸如上面所提及的重新排序缓冲区、装载缓冲区和存储缓冲区之类的部件，以支持乱序执行以及乱序执行后的指令的随后有序引退。

[0118] 在一个实施例中，调度器和执行单元块1540包括调度器单元，以调度执行单元上

的指令/操作。例如，将浮点指令调度在执行单元的一个端口上，其中该执行单元具有可用的浮点执行单元。还包括与这些执行单元相关联的寄存器文件，以存储信息指令处理结果。示例性执行单元包括浮点执行单元、整数执行单元、跳执行单元、装载执行单元、存储执行单元和其它已知的执行单元。

[0119] 低层级数据高速缓存和数据转换缓冲区(D-TLB)1550耦合到执行单元1540。该数据高速缓存用于存储最近使用/操作的元素(例如，数据操作数)，它们潜在地以存储器一致性状态进行保持。D-TLB用于存储最近的虚拟/线性与物理地址转换。举一个特定的例子，处理器可以包括用于将物理存储器分解成多个虚拟页的页表结构。

[0120] 这里，内核1501和1502共享对于高层级或者进一步外面的高速缓存(例如，与片上接口1510相关联的第二层级高速缓存)的访问。应当注意，高层级或者进一步外面指代高速缓存水平增加或者进一步远离执行单元。在一个实施例中，更高层级高速缓存是上一层级数据高速缓存(处理器1500上的存储器等级里的上一高速缓存)，例如，第二层级或第三层级数据高速缓存。但是，高层级高速缓存并不受此限制，因为其可以与指令高速缓存相关联或者包括指令高速缓存。在解码器1525之后，可以替代地耦合追踪(trace)高速缓存(其是一种类型的指令高速缓存)以存储最近解码的迹线。这里，指令潜在地指代宏指令(即，解码器识别的通用指令)，宏指令可以解码成多条微指令(微操作)。

[0121] 在所描述的配置中，处理器1500还包括片上接口模块1510。历史上，下面将进一步详细描述的存储器控制器，包括在处理器1500外部的计算系统中。在该场景中，片上接口1510用于与处理器1500外部的设备进行通信，例如，系统存储器1575、芯片集(其通常包括用于连接到存储器1575的存储器控制器集线器，以及用于连接外围设备的I/O控制器集线器)、存储器控制器集线器、北桥或者其它集成电路。在该场景中，总线1505可以包括任何已知的互连，例如，多点式总线、点对点互连、串行互连、并行总线、一致性(例如，高速缓存一致性)总线、分层协议架构、差分总线和GTL总线。

[0122] 存储器1575可以专用于处理器1500，或者与系统中的其它设备进行共享。存储器1575的类型的通常示例包括DRAM、SRAM、非易失性存储器(NV存储器)和其它已知的存储设备。应当注意，设备1580可以包括图形加速器、耦合到存储器控制器集线器的处理器或卡、耦合到I/O控制器集线器的数据存储单元、无线收发机、闪存器件、音频控制器、网络控制器或者其它已知设备。

[0123] 但是最近，随着在单一管芯上集成更多的逻辑单元和器件(例如，SOC)，可以将这些器件中的每一个合并在处理器1500上。例如，在一个实施例中，存储器控制器集线器与处理器1500位于相同的封装和/或管芯上。这里，内核的一部分(核上部分)1510包括用于与诸如存储器1575或图形设备1580之类的其它设备对接的一个或多个控制器。包括有用于与这些设备对接的互连和控制器的配置通常称为核上(或者非内核配置)。举例而言，片上接口1510包括用于片上通信的环形互连和用于片外通信的高速串行点对点链路1505。在SOC环境下，可以将甚至更多的设备(例如，网络接口、协同处理器、存储器1575、图形处理器1580和任何其它已知的计算机设备/接口)集成在单一芯片或者集成电路上，以便提供具有高功能性和低功耗的较小形状因子。

[0124] 在一个实施例中，处理器1500能够对应用代码1576执行编译、优化，和/或转换器代码1577用于对应用代码1576进行编译、转换和/或优化，从而支持本文所描述的装置和方

法或与其对接。通常，编译器包括用于将源文本/代码转换成目标文本/代码的程序或者程序集。通常，利用编译器对程序/应用代码的编译是在多个阶段和多遍完成的，以将高级编程语言代码转换成低级机器或汇编语言代码。单遍编译器仍然可以用于简单的编译。编译器可以使用任何已知的编译技术，并执行任何已知的编译器操作，例如，词法分析、预处理、解析、语义分析、代码生成、代码转换和代码优化。

[0125] 更大的编译器通常包括多个阶段，但通常情况下，这些阶段包括在两个通用阶段中：(1)前端，即，通常可以发生句法处理、语义处理和一些转换/优化的位置；和(2)后端，即，通常发生分析、转换、优化和代码生成的位置。一些编译器指代中间物，其描绘了编译器的前端和后端之间界线的模糊。结果，对于编译器的插入、关联、生成或者其它操作的引用，可以在任何前述的阶段或者遍、以及编译器的任何其它已知阶段或遍中发生。举一个示例性例子，编译器潜在地在编译的一个或多个阶段中插入操作、调用、函数等等，例如，在编译的前端阶段中插入调用/操作，随后在转换阶段期间，将这些调用/操作转换成低级代码。应当注意，在动态编译期间，编译器代码或者动态优化代码可以插入这些操作/调用，以及对代码进行优化以便在运行时期间执行。举一个特定的示例性例子，可以在运行时期间，对二进制代码(已经编译的代码)进行动态优化。这里，程序代码可以包括动态优化代码、二进制代码或者其组合。

[0126] 类似于编译器，诸如二进制转换器之类的转换器对代码进行静态地或动态地转换，以优化和/或转换代码。因此，对于代码、应用代码、程序代码或者其它软件环境的执行的引用，可以指代：(1)编译器程序、优化代码优化器、或者转换器的动态或者静态执行，以便编译程序代码、维持软件结构、执行其它操作、优化代码或者转换代码；(2)包括操作/调用的主程序代码(例如，已优化/编译的应用代码)的执行；(3)与主程序代码相关联的其它程序代码(例如，库)的执行，以维持软件结构，执行其它与软件有关的操作，或者优化代码；或者(4)其组合。

[0127] 现参见图16，示出了多核处理器的实施例的框图。如图16的实施例中所示，处理器1600包括多个域。具体而言，内核域1630包括多个内核1630A-1630N，图形域1660包括具有媒体引擎1665的一个或多个图形引擎，以及系统代理域1610。

[0128] 在各个实施例中，系统代理域1610处理功率控制事件和功率管理，使得域1630和1660的各个单元(例如，内核和/或图形引擎)可被独立地控制，以便鉴于在给定的单元中发生的活动(或者不活动)，按照适当的功率模式/水平(例如，活动、加速、休眠、冬眠、深度休眠或者其它高级配置功率接口类似状态)进行动态地操作。域1630和1660中的每一个可以按照不同的电压和/或功率进行操作，此外，这些域中的各个单元均潜在地按照独立的频率和电压进行操作。应当注意，虽然只示出了三个域，但应当理解，本发明的范围在该方面并不受限，在其它实施例中可以存在额外的域。

[0129] 如图所示，除了各个执行单元和额外处理元件之外，每一个内核1630还包括低层级高速缓存。这里，各个内核彼此之间耦合，并耦合到由上一级高速缓存(LLC)1640A-1640N的多个单元或切片(slice)构成的共享高速缓存；这些LLC通常包括存储和高速缓存控制器功能，并在内核之间、以及潜在地在图形引擎之间进行共享。

[0130] 如图所示，环形互连1650将内核耦合在一起，并经由多个环形点1652A-1652N来提供内核域1630、图形域1660和系统代理电路1610之间的互连，其中每一个环形点处于内核

和LLC切片之间的耦合处。如图16中所观察到的，互连1650用于携带各种信息，其包括地址信息、数据信息、确认信息和监听/无效信息。虽然示出了环形互连，但可以使用任何已知的管芯上互连或架构。举一个示例性的例子，可以以类似的方式，使用上面所讨论的架构中的一些(例如，另一种管芯上互连、片上系统结构(OSF)、高级微控制器总线结构(AMBA)互连、多维度网格架构或者其它已知的互连架构)。

[0131] 作为进一步的描述，系统代理域1610包括显示引擎1612，显示引擎1612提供针对到相关联的显示的接口的控制。系统代理域1610可以包括其它单元，例如：提供到系统存储器(例如，利用多个DIMM实现的DRAM)的接口的集成存储器控制器1620；用于执行存储器一致性操作的一致性逻辑单元1622。可以给出多个接口以实现处理器和其它电路之间的互连。例如，在一个实施例中，提供至少一个直接媒体接口(DMI)1616接口以及一个或多个PCIe™接口1614。显示引擎和这些接口通常经由PCIe™桥1618来耦合到存储器。另外，为了提供其它代理(例如，额外的处理器或其它电路)之间的通信，可以提供一个或多个其它接口。

[0132] 虽然相对于有限数量的实施例来描述了本发明，但本领域普通技术人员应当理解它们的众多修改和变形。所附权利要求书旨在覆盖落入本发明的实质精神和范围之内的所有这些修改和变形。

[0133] 设计方案可能经历从产生到模拟到构造的各个阶段。表示设计的数据可以利用多种方式来表示该设计。首先，如同在模拟中所使用的，可以采用硬件描述语言(HDL)或者另一种功能描述语言来表示硬件。另外，在设计过程的一些阶段，可以生成具有逻辑单元和/或晶体管门的电路层级模型。此外，大部分设计在某个阶段，达到表示各个器件在硬件模型中的物理放置的数据水平。在使用传统的半导体构造技术的情况下，表示硬件模型的数据可以是针对用于制造集成电路的掩模，指示不同的掩模层上的各个特征的存在或缺失的数据。

[0134] 在一些实现方式中，基于软件的硬件模型和HDL以及其它功能描述语言对象，可以包括寄存器转换语言(RTL)文件等等其它示例。这些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来对HDL对象进行解析，以及根据对象来确定物理电路和/或片上布局。可以使用设计工具的输出来制造物理设备。例如，设计工具可以根据HDL对象，来确定各种硬件和/或固件元件的配置(例如，总线宽度、寄存器(其包括大小和类型)、存储器块、物理链路路径、结构拓扑等等其它属性，其会被执行以便实现在HDL对象中建模的系统)。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对象可以用作用于开发模型和设计文件的基础，其中在制造设备时可以使用该模型和设计文件来制造所描述的硬件。事实上，可以将HDL对象自身提供为制造系统软件的输入，以导致所描述的硬件。

[0135] 在设计的任何表示中，都可以将数据存储在任何形式的机器可读介质中。存储器或诸如磁盘之类的磁或光存储单元，可以是用于存储经由光波或电波发送的信息的机器可读介质，其中为了发送该信息而对光波或电波进行了调制或者生成。当发送指示或者携带代码或设计的电子载波时，达到执行电信号的复制、缓冲或重新传输的程度时，完成新的复制。因此，通信提供商或者网络提供商可以将体现本公开内容的实施例的技术的制品(例如，编码到载波中的信息)，至少临时地存储在有形的机器可读介质上。

[0136] 如本文所使用的模块指代硬件、软件和/或固件的任意组合。举例而言，模块包括与非临时性介质相关联的硬件(例如，微控制器)，以存储适于由该微控制器执行的代码。因此，在一个实施例中，对于模块的引用，指代硬件，该硬件具体被配置为识别和/或执行在非临时性介质上保持的代码。此外，在另一个实施例中，使用模块来指代包括有代码的非临时性介质，其中该代码具体适于由微控制器执行以执行预定的操作。在另一个实施例中，如同所可以推断的，(该示例中的)术语模块可以指代微控制器和非临时性介质的组合。通常，被示出为独立的模块边界通常会有所变化并潜在地重叠。例如，第一模块和第二模块可以共享硬件、软件、固件或者其组合，同时潜在地保持一些独立的硬件、软件或固件。在一个实施例中，术语逻辑单元的使用包括硬件，例如，晶体管、寄存器或者其它硬件(例如，可编程逻辑器件)。

[0137] 在一个实施例中，使用短语“被配置为”指代：对装置、硬件、逻辑单元或者用于执行指定或确定的任务的元素进行布置、放置在一起、制造、供应销售、引入和/或设计。在该示例中，没有操作的装置或者其元素，仍然“被配置为”执行指定的任务(如果其被设计、耦合和/或互连以执行所述指定的任务的话)。举一个纯粹的示例性例子，逻辑门可以在操作期间提供0或1。但“被配置为”向时钟提供启用信号的逻辑门，并不包括可以提供1或0的每一个潜在逻辑门。事实上，逻辑门是以某种方式进行耦合的，在操作期间，1或0的输出将启用时钟的一个逻辑门。再一次注意，术语“被配置为”的使用并不需要操作，而是聚焦于装置、硬件和/或元素的潜在状态，其中在该潜在状态下，当该装置、硬件和/或元素在操作时，设计该装置、硬件和/或元素以执行特定的任务。

[0138] 此外，在一个实施例中，使用短语“用于”、“能够”和/或“操作为”指代：以这种方式设计某个装置、逻辑单元、硬件和/或元素，将以指定的方式来启用该装置、逻辑单元、硬件和/或元素的使用。如上所述，应当注意，在一个实施例中，使用“用于”、“能够”或“操作为”指代装置、逻辑单元、硬件和/或元素的潜在状态，其中该装置、逻辑单元、硬件和/或元素不进行操作，但以这种方式被设计成以指定的方式来启用装置的使用。

[0139] 如本文所使用的，值包括数字、状态、逻辑状态或者二进制逻辑状态的任何已知表示。通常，还将逻辑电平、逻辑值或者逻辑性值的使用称为1的和0的，其简单地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(例如，晶体管或闪存单元)可以能够保持单一逻辑值或者多个逻辑值。但是，已使用计算机系统中的值的其它表示方式。例如，十进制数字十还可以表示成1010的二进制值和十六进制字母A。因此，值包括能够在计算机系统中保持的信息的任何表示。

[0140] 此外，可以通过值或者值的多个部分来表示状态。举例而言，诸如逻辑“一”之类的第一值可以表示缺省或者初始状态，而诸如逻辑“零”之类的第二值可以表示非缺省状态。此外，在一个实施例中，术语重置和设置分别指代：缺省和更新的值或状态。例如，缺省值潜在地包括高逻辑值(即，重置)，而更新的值潜在地包括低逻辑值(即，设置)。应当注意，可以使用值的任意组合来表示任意数量的状态。

[0141] 上文阐述的方法、硬件、软件、固件或代码的实施例，可以经由在机器可访问、机器可读、计算机可访问或者计算机可读介质上存储的可被处理元件执行的指令或代码来实现。非临时性机器可访问/可读介质包括以机器(例如，计算机或电子系统)可读的形式来提供(即，存储和/或发送)信息的任何机制。例如，非临时性机器可访问介质包括诸如静态RAM

(SRAM)或动态RAM(DRAM)之类的随机存取存储器(RAM);ROM;磁或光存储介质;闪存器件;电存储设备;光存储设备;声存储设备;用于保持从临时(传播的)信号(例如,载波、红外信号、数字信号)接收的信息的其它形式的存储设备等等,它们与可以从其接收信息的非临时性介质区分开来。

[0142] 用于执行本发明的实施例的程序逻辑单元所使用的指令,可以存储在系统中的诸如DRAM、高速缓存、闪存或其它存储单元之类的存储器中。此外,这些指令也可以经由网络或者通过其它计算机可读介质的方式来进行分布。因此,机器可读介质可以包括用于以机器(例如,计算机)可读的形式来存储或发送信息的任何机制,其并不限于:软盘、光盘、光碟只读存储器(CD-ROM)、和磁光盘、只读存储器(ROM)、随机存取存储器(RAM)、可擦除可编程只读存储器(EPROM)、电可擦除可编程只读存储器(EEPROM)、磁卡或光卡、闪存、或者经由电、光、声或其它形式的传播信号(例如,载波、红外信号、数字信号等等)通过互联网来传输信息时使用的有形机器可读存储单元。因此,计算机可读介质包括适合于以机器(例如,计算机)可读的形式来存储或发送电指令或信息的任何类型的有形机器可读介质。

[0143] 下面的示例涉及根据本说明书的实施例。一个或多个实施例可以提供装置、系统、机器可读存储单元、机器可读介质、基于硬件和/或软件的逻辑单元以及方法,以在根据存储器访问链路协议的链路上,发送数据以对应于与共享存储器相关联的装载/存储类型操作,且该存储器访问链路协议将覆盖另一种不同的链路协议上,以及发送进入低功率状态的请求,其中该请求包括在令牌的字段中编码的数据值,该令牌用于指示分组数据的起始,并进一步指示在该令牌之后发送的随后数据是否包括根据另一链路协议和存储器访问链路协议其中之一的数据。

[0144] 在至少一个示例中,所述另一链路协议包括通用输入/输出(I/O)互连协议。

[0145] 在至少一个示例中,所述令牌是根据所述通用I/O互连协议来规定的。

[0146] 在至少一个示例中,所述令牌的第一字段指示随后数据是包括存储器访问链路协议数据,还是包括通用I/O互连协议数据。

[0147] 在至少一个示例中,当第一字段指示随后数据包括存储器访问链路协议数据时,将所述数据值编码在令牌的第二字段中。

[0148] 在至少一个示例中,所述数据值包括根据存储器访问链路协议来规定的编码,并且使用第二字段来在通用I/O互连协议的物理层上,对所述编码进行隧道化。

[0149] 在至少一个示例中,基于令牌的实例,在所述链路上发送的数据在存储器访问链路协议数据和通用I/O互连协议数据之间切换。

[0150] 在至少一个示例中,在发送进入低功率状态的请求之前,功率管理逻辑单元检查通用I/O互连协议的数据是否在等待进行发送。

[0151] 在至少一个示例中,功率管理逻辑单元等待发送进入低功率状态的请求,直到发送通用I/O互连协议数据。

[0152] 在至少一个示例中,存储器访问链路协议或者通用I/O互连协议的链路层,可以触发物理层进入到低功率状态。

[0153] 在至少一个示例中,在通用I/O互连协议中用于进入低功率状态的信号,与在存储器访问链路协议中用于进入低功率状态的信号不同。

[0154] 在至少一个示例中,存储器访问链路协议规定了其中在存储器访问链路协议中发

送进入低功率状态的信号的周期性控制窗口。

[0155] 一个或多个实施例可以提供用于在链路上接收数据流的装置、系统、机器可读存储单元、机器可读介质、基于硬件和/或软件的逻辑单元以及方法。该数据流可以包括令牌，该令牌包括第一字段以指示在令牌随后的数据包括根据至少两种可选协议中的一种的数据。所述至少两种可选协议可以包括存储器访问协议和通用I/O协议。在所述链路上发送的所有数据都在通用I/O协议的物理层上发送，所述令牌的第二字段被编码以指示存储器访问协议的链路层的进入低功率链路状态的请求。可以基于所述请求，来发起进入低功率链路状态。

[0156] 在至少一个示例中，发送针对所述请求的响应，其中该响应包括确认和否定确认其中之一，并且该响应在所述令牌的另一个实例中进行发送。

[0157] 在至少一个示例中，所述响应是在所述令牌的另一实例的第二字段的实例中发送的。

[0158] 一个或多个实施例可以提供装置、系统、机器可读存储单元、机器可读介质、基于硬件和/或软件的逻辑单元，其提供第一协议的链路层、不同的第二协议的链路层、用于发送至少第二协议的数据的第一协议的物理层和垫层。垫层可以接收包括令牌的数据流，其中该令牌包括第一字段以指示在该令牌随后的数据将包括第一协议数据或者第二协议数据其中之一，而该令牌的第二字段被编码以指示第二协议的链路层的进入低功率链路状态的请求。垫层还可以向物理层发送数据，以使物理层进入低功率状态。

[0159] 在至少一个示例中，第一协议包括通用I/O互连协议，而第二协议包括存储器访问协议。

[0160] 在至少一个示例中，还包括第一协议的一个或多个层以及第二协议的一个或多个层。

[0161] 在至少一个示例中，第一协议的链路层被禁用。

[0162] 贯穿本说明书对于“一个实施例”或“实施例”的引用，意味着结合该实施例描述的特定特征、结构或特性包括在本发明的至少一个实施例中。因此，在遍布本说明书的各个地方出现的短语“在一个实施例中”或者“在实施例中”，并不必然全部都指代相同的实施例。此外，可以以任何适当的方式，将这些特定的特征、结构或特性组合在一个或多个实施例中。

[0163] 在上面的说明中，参照特定的示例性实施例来给出详细的描述。但是，将显而易见的是，可以在不偏离如权利要求书所阐述的本发明的更广精神和范围的基础上，对本发明做出各种修改和改变。因此，说明书和附图被视作为示例性的，而不是限制性的。此外，上述实施例的使用和其它示例性语言并不必然指代相同的实施例或者相同的例子，而可以指代不同的和截然不同的实施例，以及潜在地指代相同的实施例。

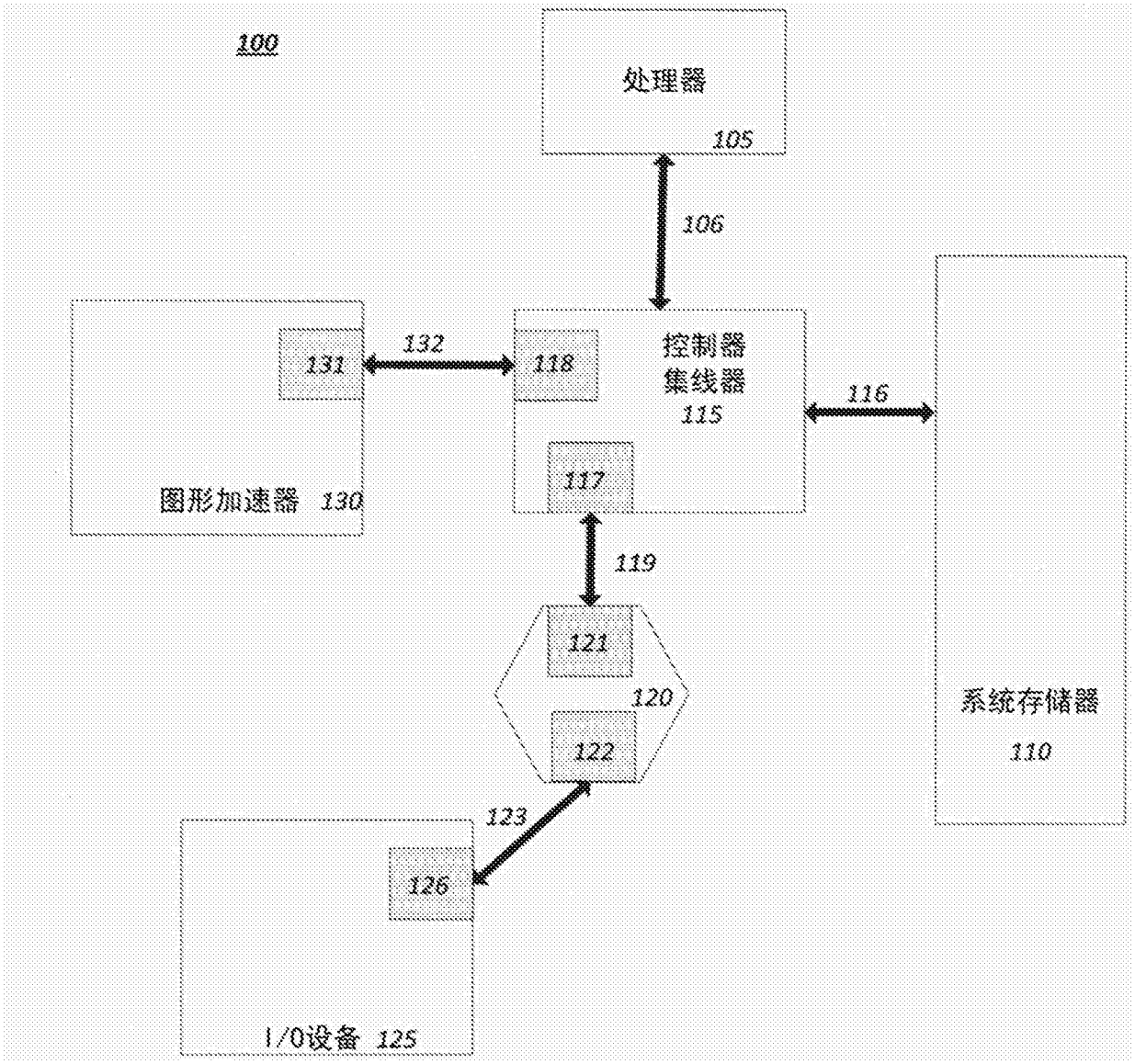

100
处理器
105
106
131
132
118
控制器
集线器
115
116
117
图形加速器 130
119
121
120
122
系统存储器
110
123
126
I/O设备 125

图1

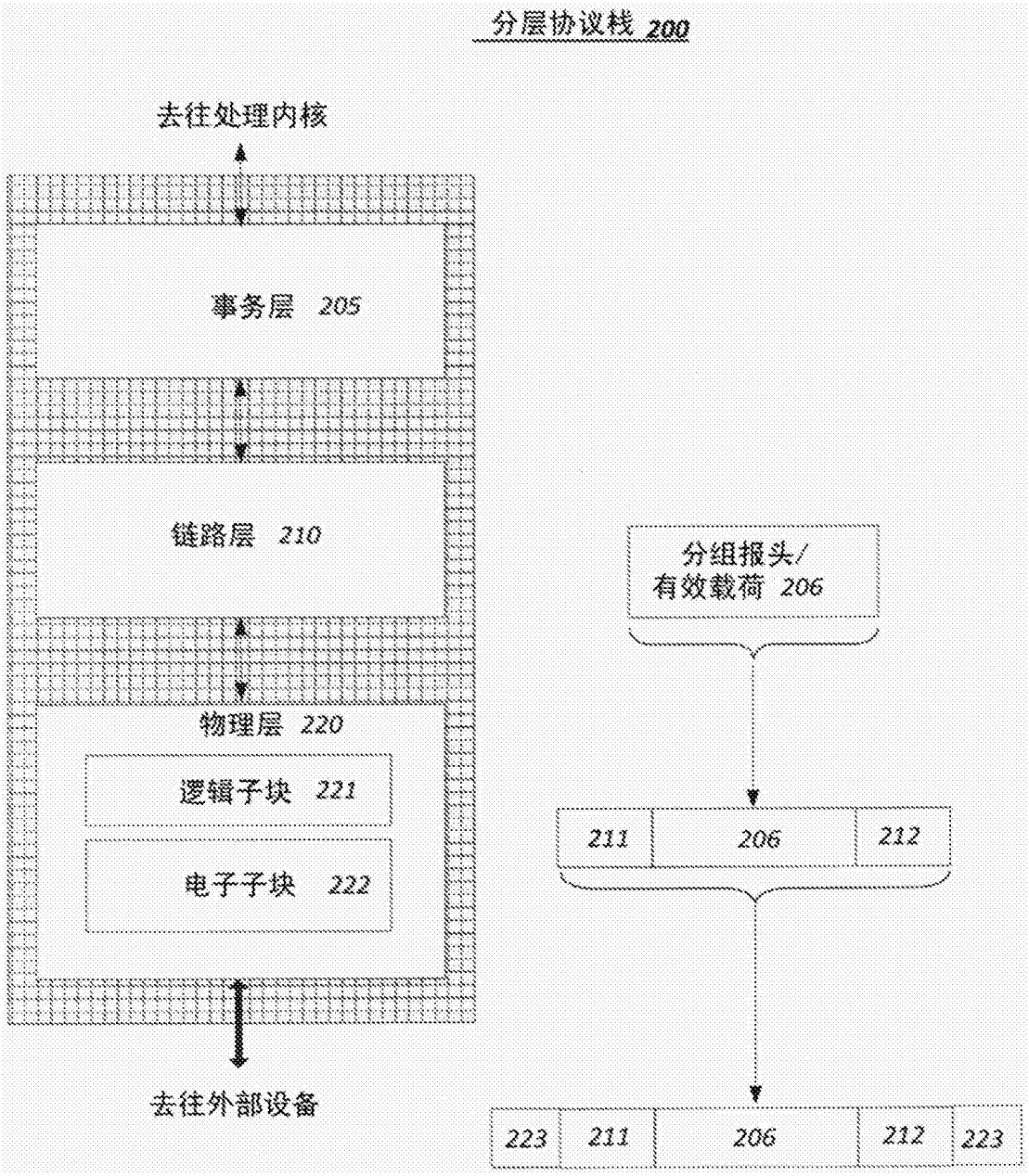

分层协议栈 200
去往处理内核
事务层 205
链路层 210
物理层 220
逻辑子块 221
电子子块 222
去往外部设备
分组报头/
有效载荷 206
211
206
212
223
211
206
212
223

图2

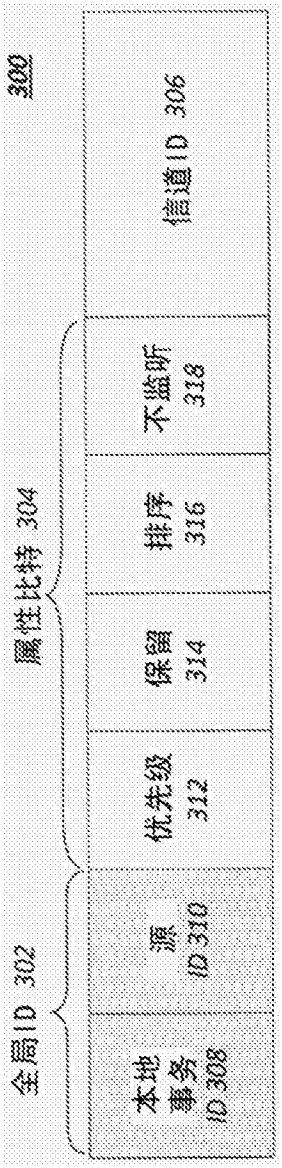

300
信道ID 306
不监听 318
排序 316
保留 314
优先级 312
源 ID 310
本地事务 ID 308
全局ID 302
属性比特 304

图3

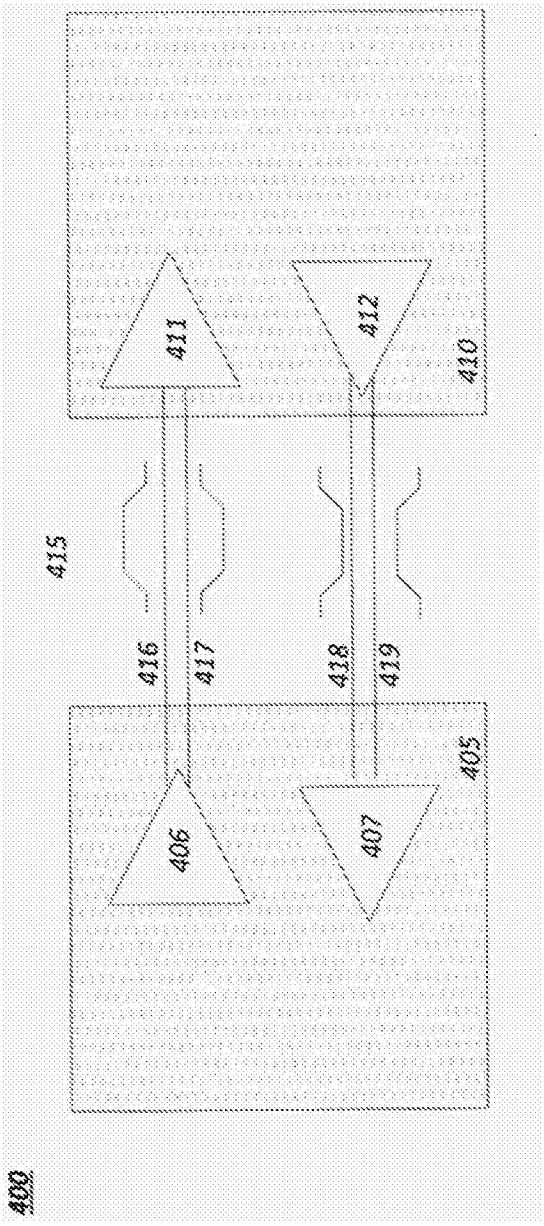

400
411
412
410
415
416
417
418
419
406
407
405

图4

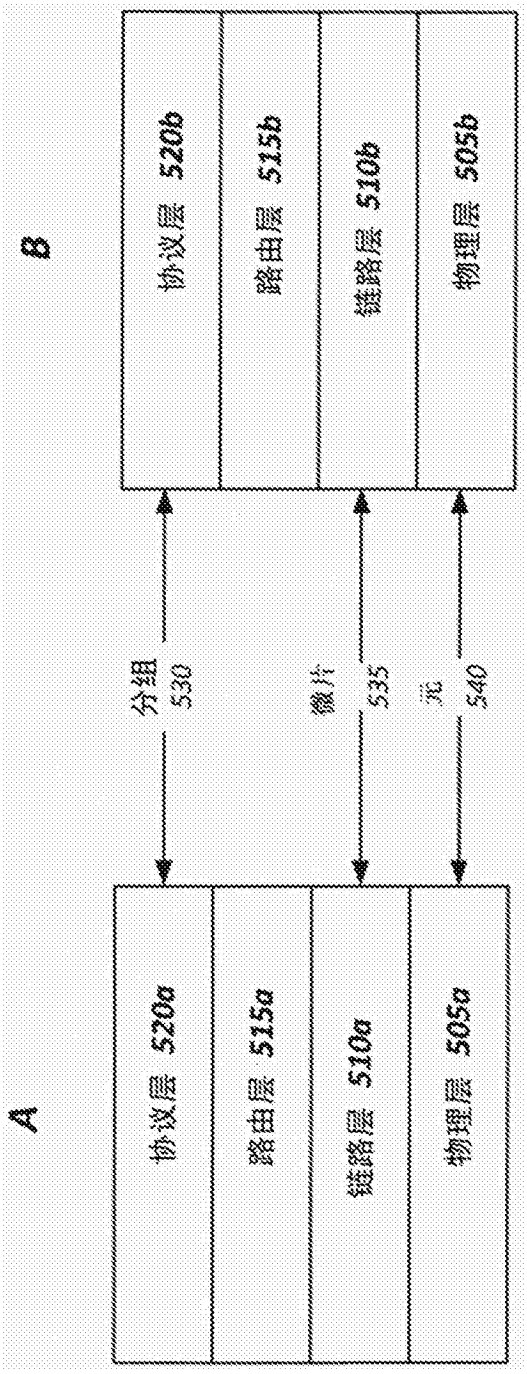

B
协议层 520b
路由层 515b
链路层 510b
物理层 505b
分组 530
微片 535
元 540
A
协议层 520a
路由层 515a
链路层 510a
物理层 505a

图5

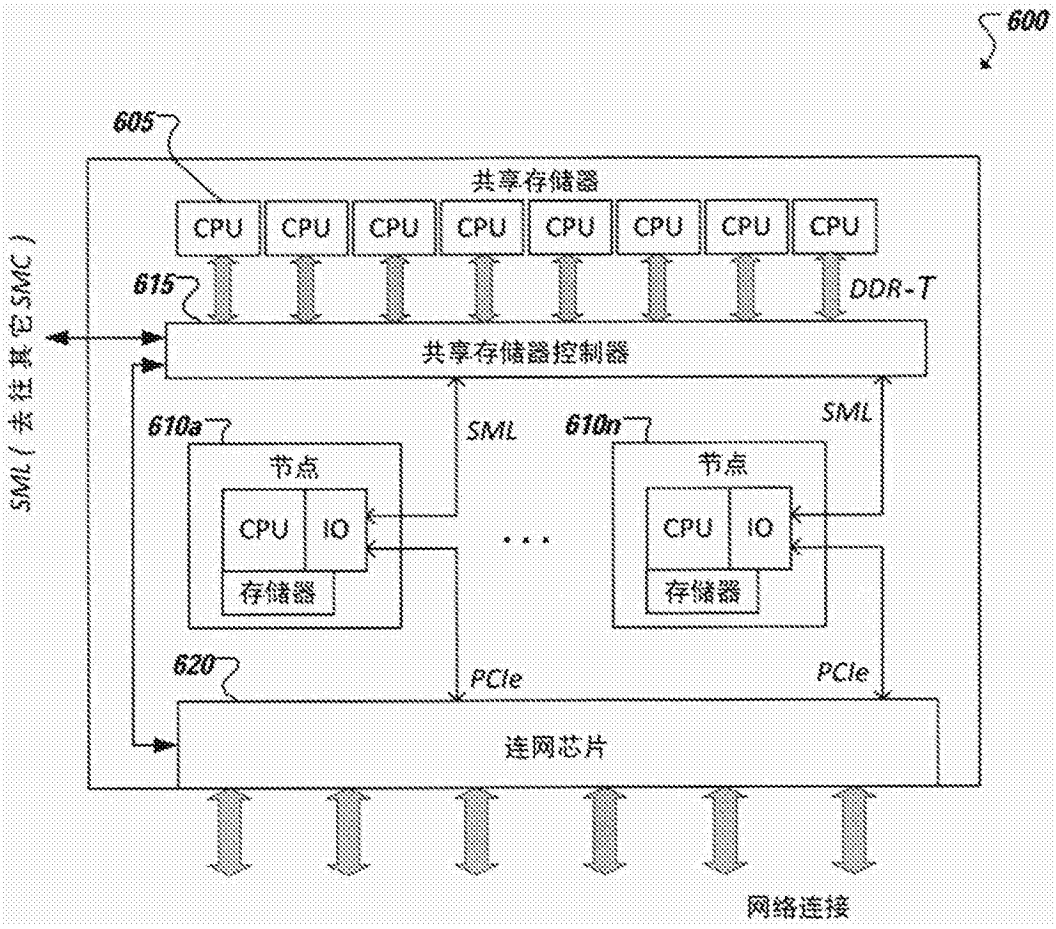

600
605
共享存储器
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
615
DDR-T
共享存储器控制器
SML（去往其它SMC）
610a
SML
610n
SML
节点
CPU
IO
存储器
节点
CPU
IO
存储器
620
PCIe
PCIe
连网芯片
网络连接

图6

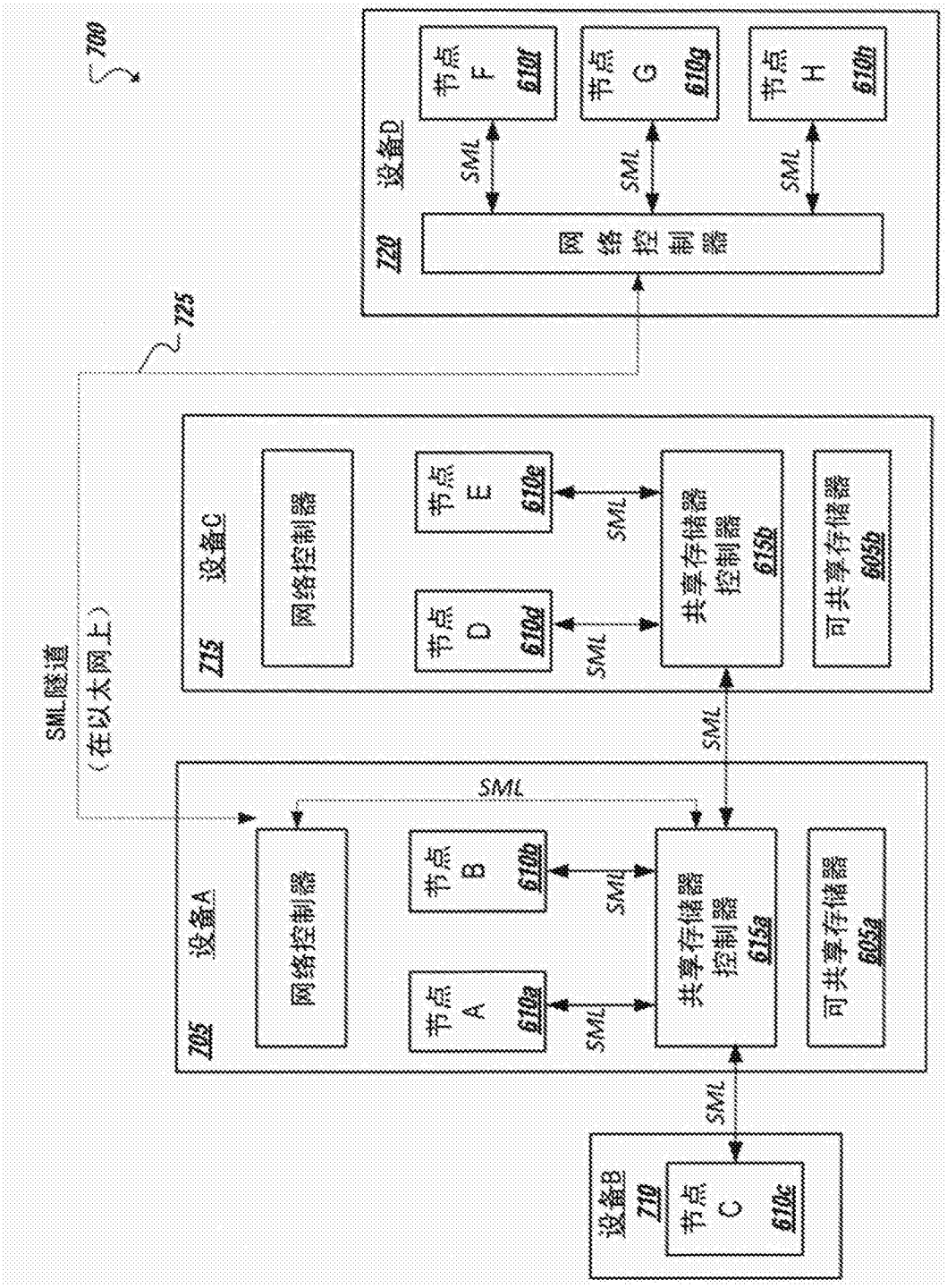

700
725
SML隧道
（在以太网上）
设备D
720
网络控制器
节点 F 610f
节点 G 610g
节点 H 610h
SML
设备C
715
网络控制器
节点 E 610e
节点 D 610d
共享存储器控制器 615b
可共享存储器 605b
设备A
705
网络控制器
节点 B 610b
节点 A 610a
共享存储器控制器 615a
可共享存储器 605a
设备B
710
节点 C 610c

图7

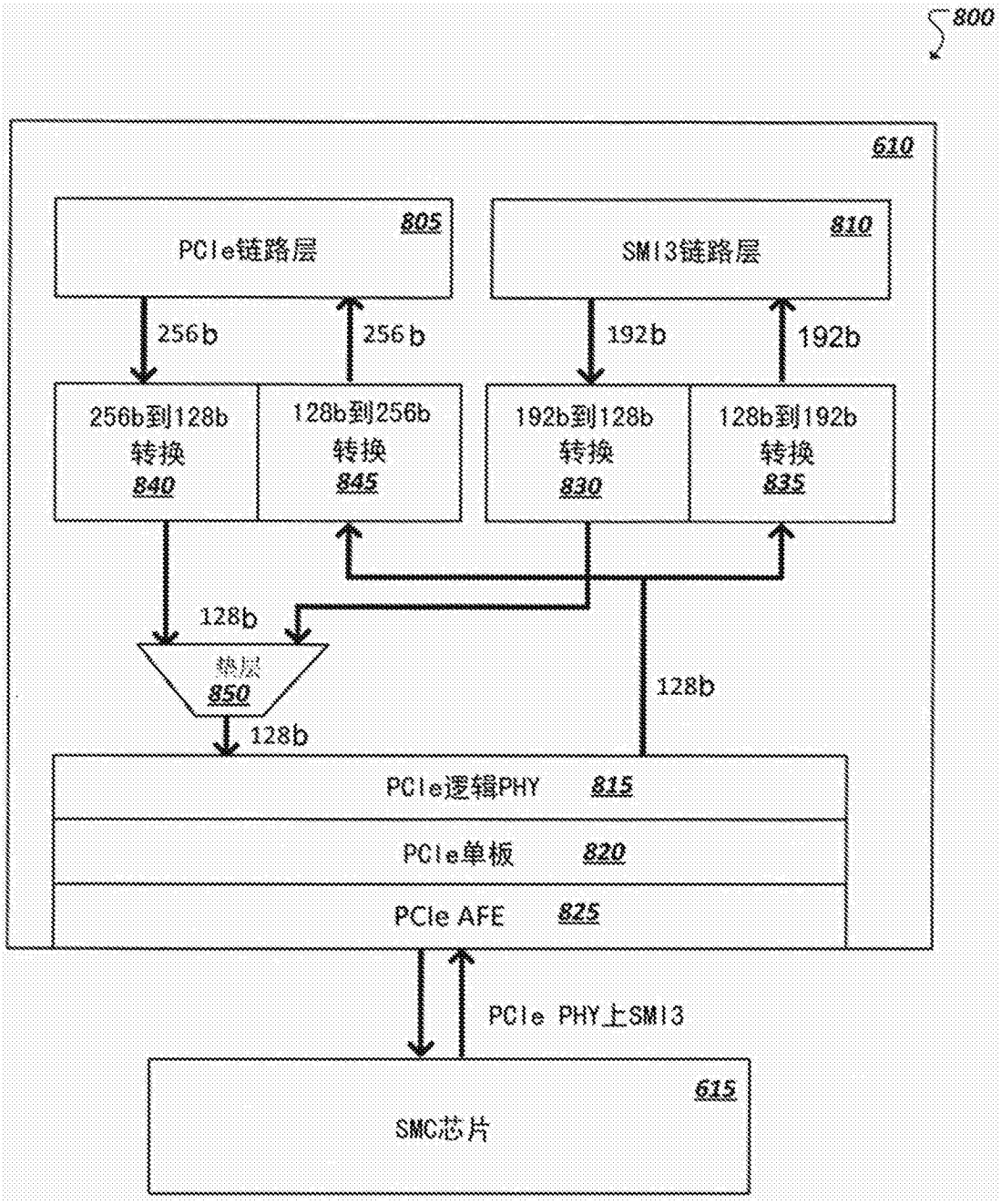

800
610
805
PCIe链路层
810
SMI3链路层
256b
256b
192b
192b
256b到128b
转换
840
128b到256b
转换
845
192b到128b
转换
830
128b到192b
转换
835
128b
单层
850
128b
128b
PCIe逻辑PHY 815
PCIe单板 820
PCIe AFE 825
PCIe PHY上SMI3
615
SMC芯片

图8

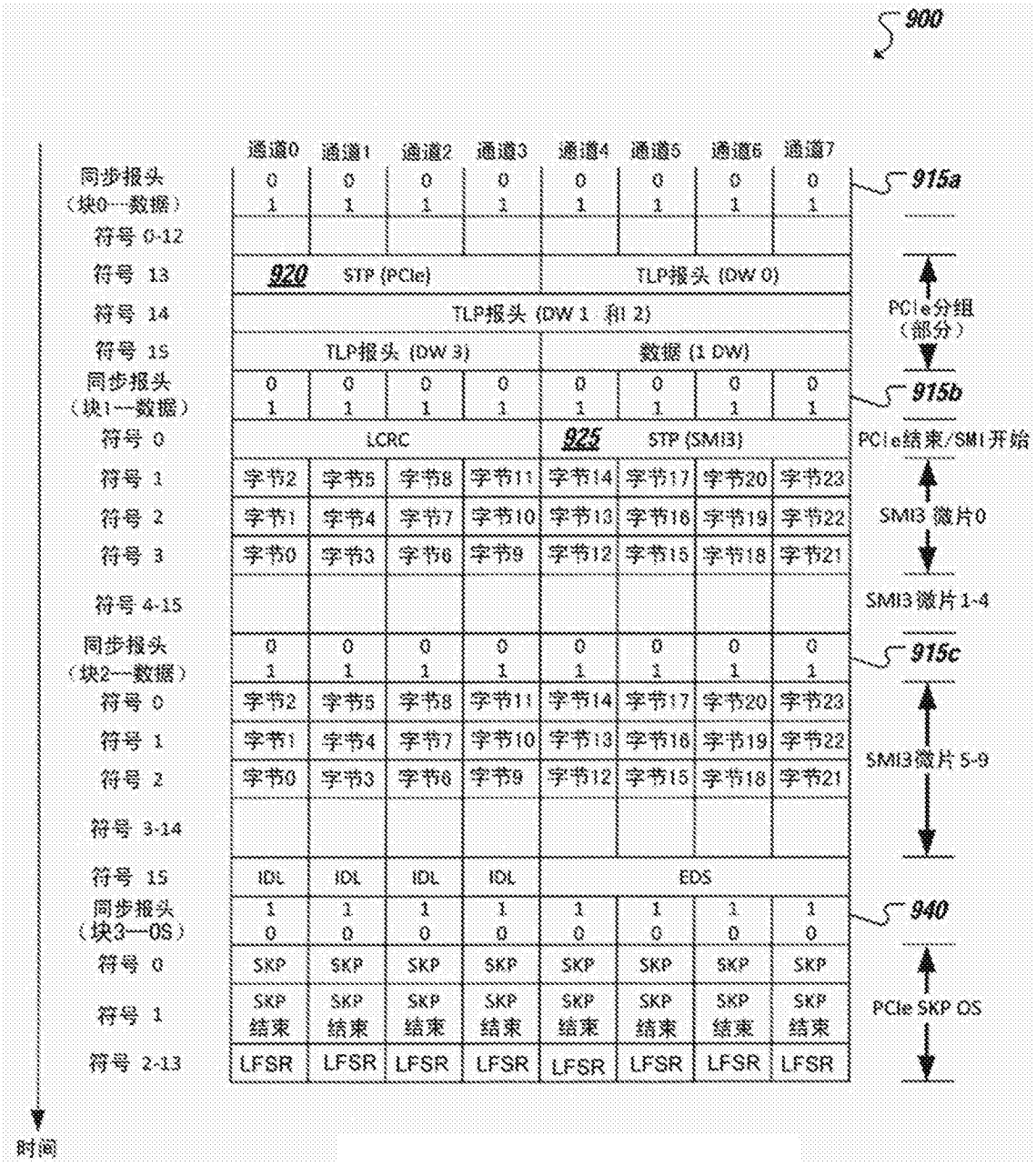

900
通道0 通道1 通道2 通道3 通道4 通道5 通道6 通道7
同步报头（块0—数据）
符号 0-12
符号 13
符号 14
符号 15
同步报头（块1—数据）
符号 0
符号 1
符号 2
符号 3
符号 4-15
同步报头（块2—数据）
符号 0
符号 1
符号 2
符号 3-14
符号 15
同步报头（块3—OS）
符号 0
符号 1
符号 2-13
920
STP (PCIe)
TLP报头 (DW 0)
TLP报头 (DW 1 和 2)
TLP报头 (DW 3)
数据 (1 DW)
LCRC
925
STP (SMI3)
字节2 字节5 字节8 字节11 字节14 字节17 字节20 字节23
字节1 字节4 字节7 字节10 字节13 字节16 字节19 字节22
字节0 字节3 字节6 字节9 字节12 字节15 字节18 字节21
IDL
EDS
SKP
SKP 结束
LFSR
915a
915b
915c
940
PCIe分组（部分）
PCIe结束/SMI开始
SMI3 微片0
SMI3 微片1-4
SMI3微片5-9
PCIe SKP OS
时间

图9

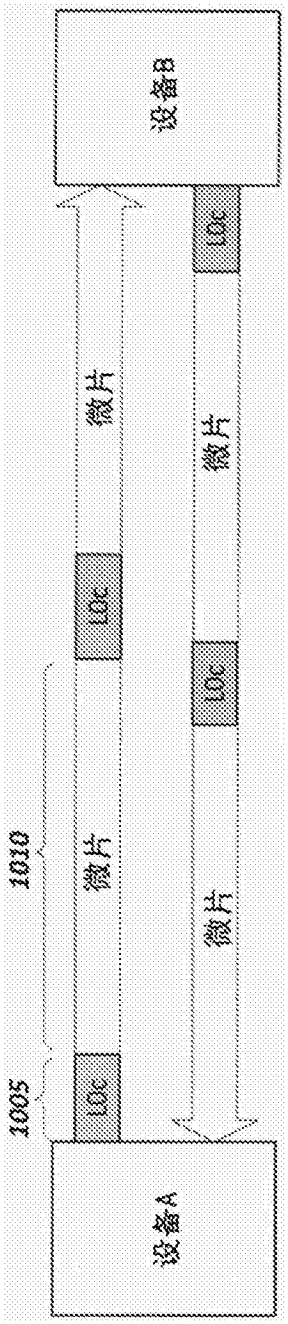

设备B
LOc
微片
微片
LOc
LOc
1010
微片
微片
1005
LOc
设备A

图10

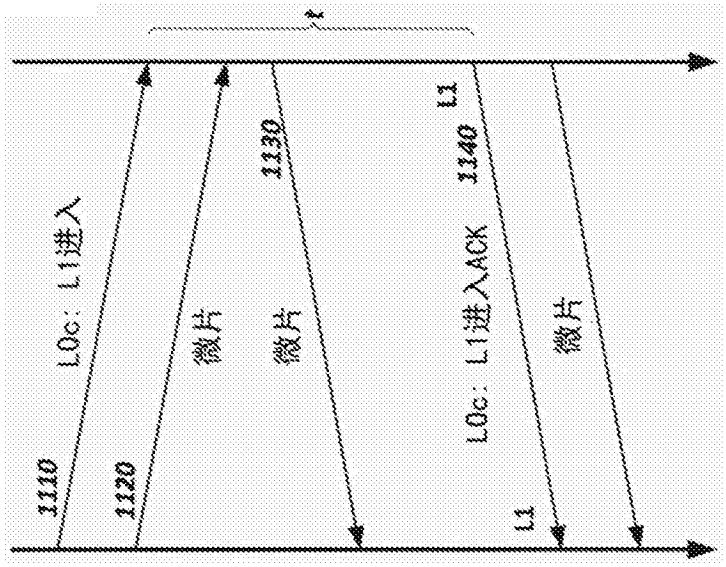

t
L1
1130
1140
1110
LOc: L1进入
1120
微片
微片
LOc: L1进入ACK
L1
微片

图11A

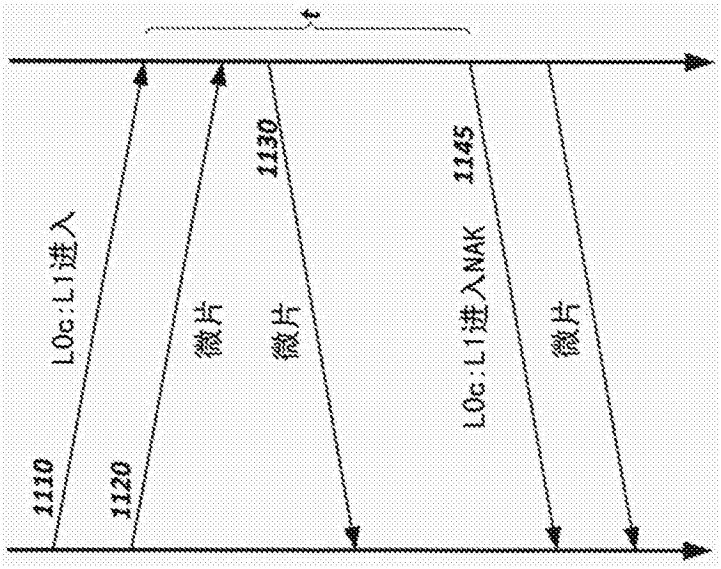


图11B

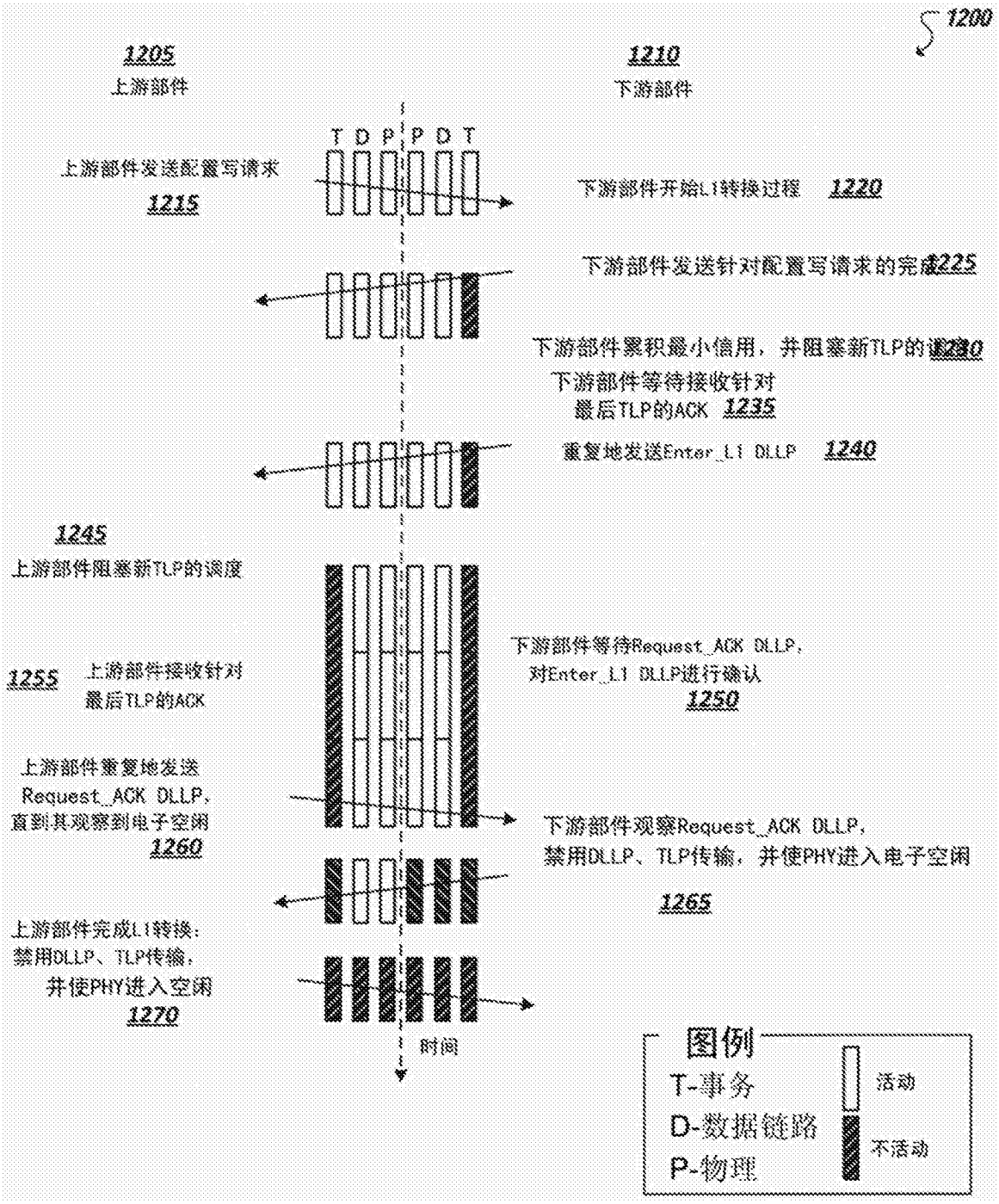

1200
1205
上游部件
1210
下游部件
T D P P D T
上游部件发送配置写请求
1215
下游部件开始L1转换过程 1220
下游部件发送针对配置写请求的完成 1225
下游部件累积最小信用，并阻塞新TLP的调度 1230
下游部件等待接收针对
最后TLP的ACK 1235
重复地发送Enter_L1 DLLP 1240
1245
上游部件阻塞新TLP的调度
下游部件等待Request_ACK DLLP，
对Enter_L1 DLLP进行确认
1250
1255
上游部件接收针对
最后TLP的ACK
上游部件重复地发送
Request_ACK DLLP，
直到其观察到电子空闲
1260
下游部件观察Request_ACK DLLP，
禁用DLLP、TLP传输，并使PHY进入电子空闲
1265
上游部件完成L1转换：
禁用DLLP、TLP传输，
并使PHY进入空闲
1270
时间
图例
T-事务
D-数据链路
P-物理
活动
不活动

图12

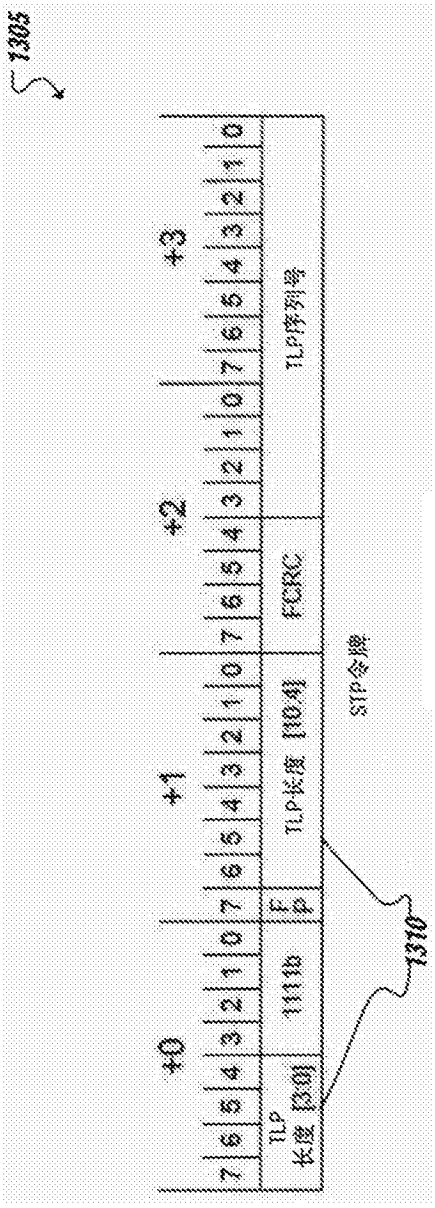

1305
+0
+1
+2
+3
7 6 5 4 3 2 1 0
TLP 长度 [9:0]
1111b
F p
TLP长度 [10:4]
FCRC
TLP序列号
STP令牌
1310

图13

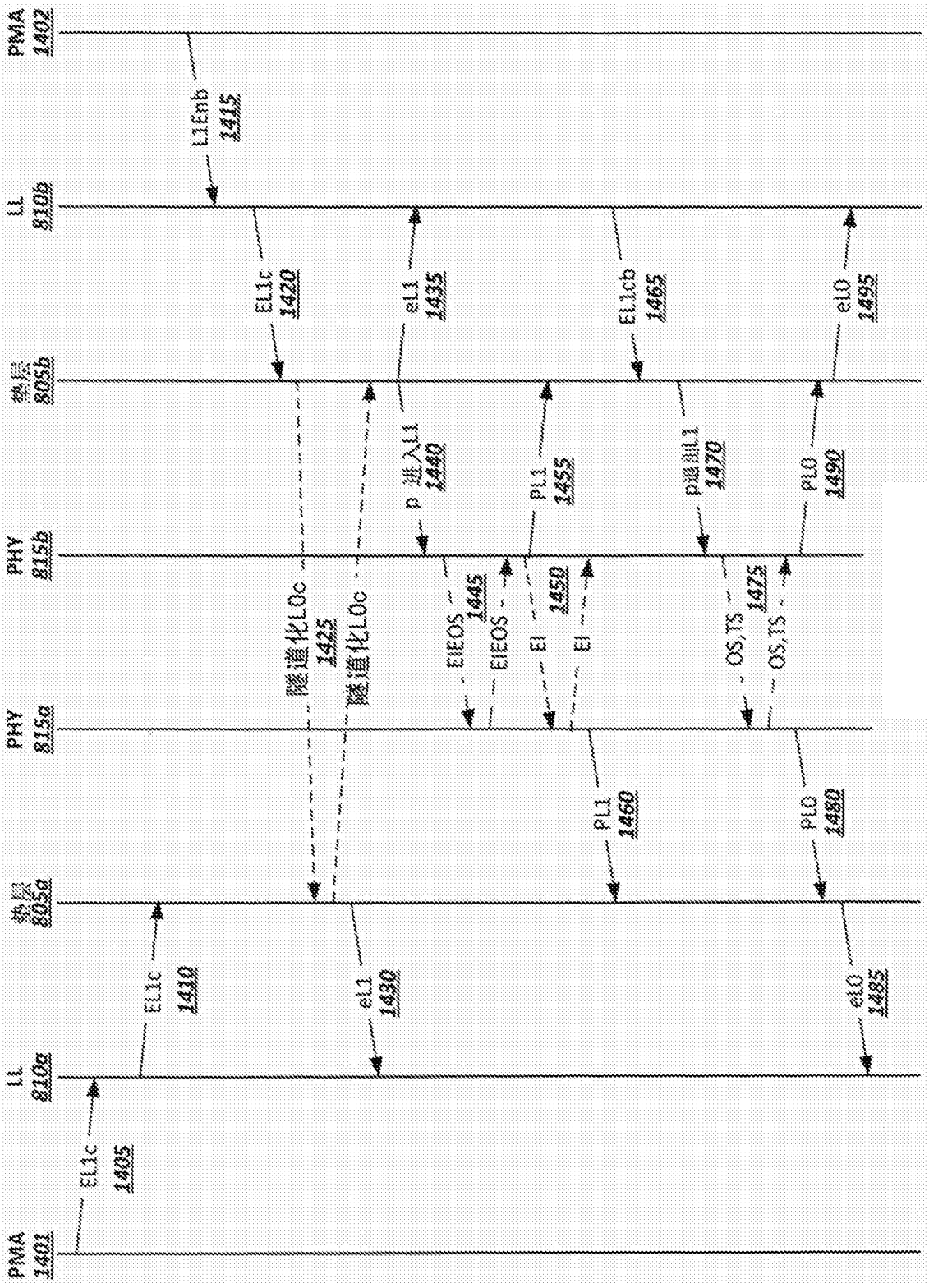

PMA 1402
LL 810b
L1Enb 1415
EL1c 1420
el1 1435
EL1cb 1465
el0 1495
PHY 815b
隧道化L0c 1425
p 进入L1 1440
EIEOS 1445
PL1 1455
EI 1450
p退出L1 1470
OS,TS 1475
PL0 1490
PHY 815a
隧道化L0c
EIEOS
EI
OS,TS
PL1 1460
PL0 1480
PMA 1401
EL1c 1405
LL 810a
EL1c 1410
eL1 1430
el0 1485

图14

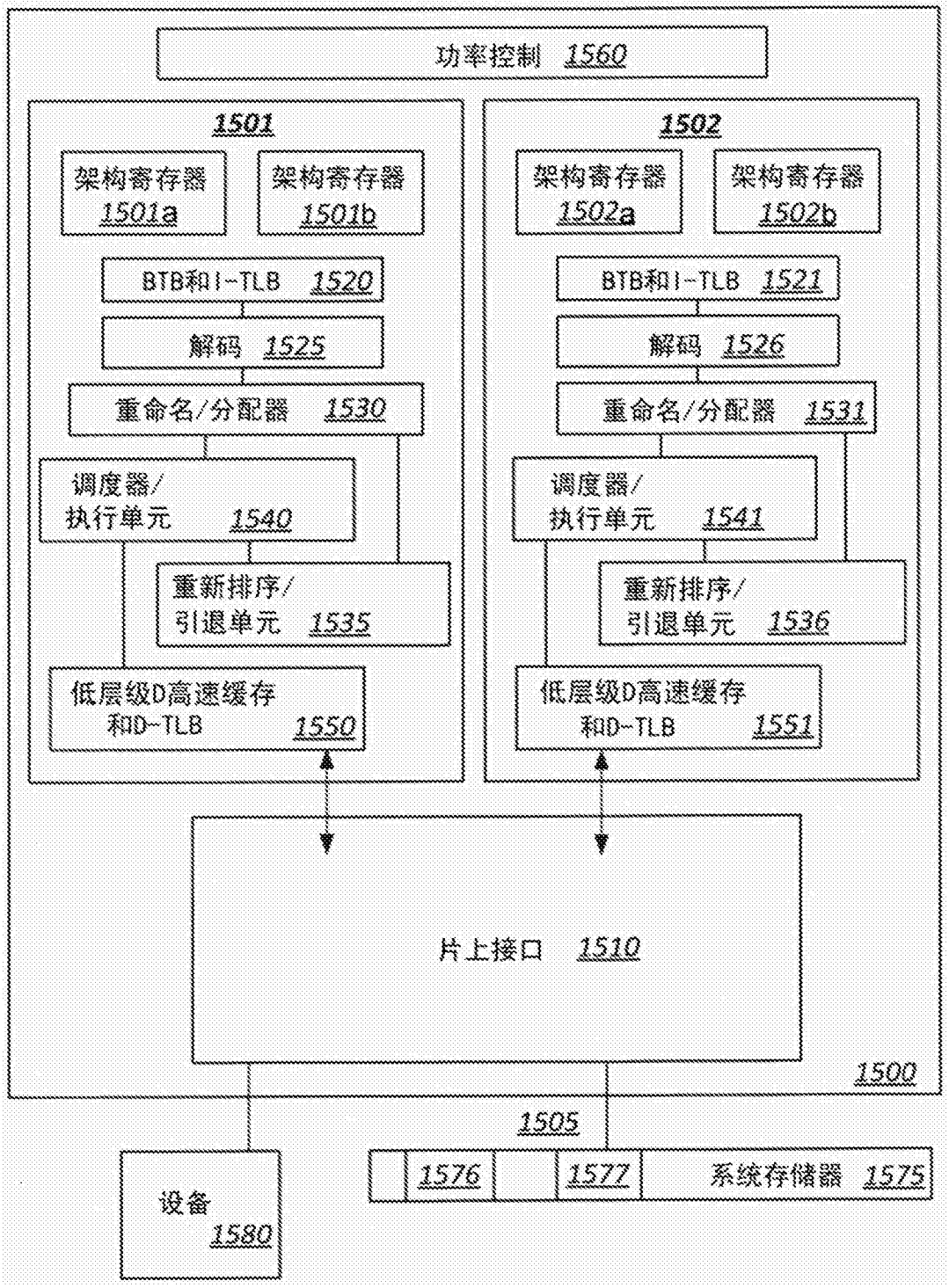

功率控制 1560
1501
架构寄存器 1501a
架构寄存器 1501b
BTB和I-TLB 1520
解码 1525
重命名/分配器 1530
调度器/执行单元 1540
重新排序/引退单元 1535
低层级D高速缓存和D-TLB 1550
1502
架构寄存器 1502a
架构寄存器 1502b
BTB和I-TLB 1521
解码 1526
重命名/分配器 1531
调度器/执行单元 1541
重新排序/引退单元 1536
低层级D高速缓存和D-TLB 1551
片上接口 1510
1500
1505
1576
1577
系统存储器 1575
设备 1580

图15

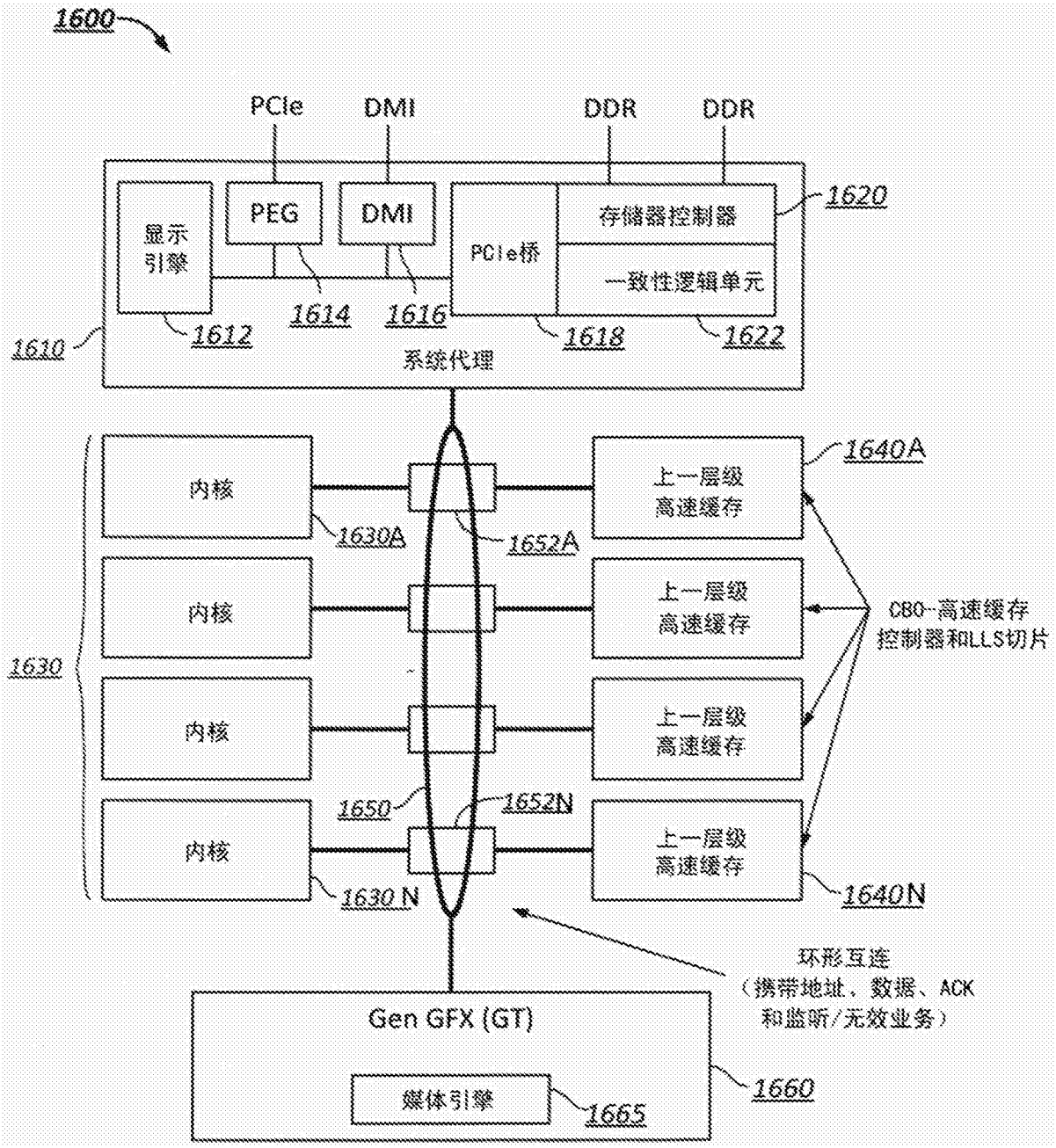

1600
PCIe
DMI
DDR
DDR
显示引擎
PEG
DMI
PCIe桥
存储器控制器
一致性逻辑单元
1620
1610
1612
1614
1616
1618
1622
系统代理
内核
1630A
1652A
上一层级高速缓存
1640A
内核
上一层级高速缓存
CBO-高速缓存控制器和LLS切片
1630
内核
上一层级高速缓存
1650
1652N
内核
1630 N
上一层级高速缓存
1640N
环形互连
（携带地址、数据、ACK和监听/无效业务）
Gen GFX (GT)
媒体引擎
1665
1660

图16