(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 110442534 A
(43)申请公布日 2019.11.12

(21)申请号 201910271381.X

(22)申请日 2019.04.04

(30)优先权数据
62/667,230 2018.05.04 US
16/141,729 2018.09.25 US

(71)申请人 英特尔公司
地址 美国加利福尼亚

(72)发明人 I•阿加瓦尔 P•普罗西特
N•帕利沃尔 A•斯里尼瓦桑

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

(51)Int.Cl.
G06F 12/0877(2016.01)
G06F 12/0815(2016.01)

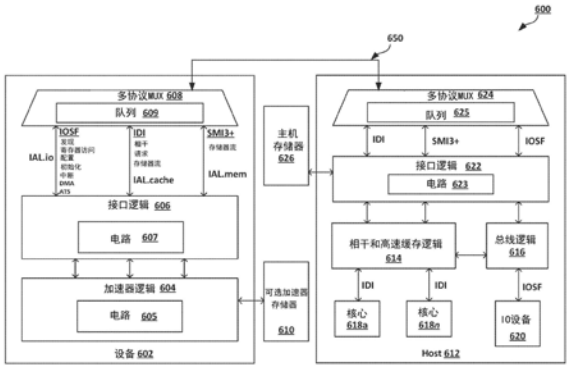
权利要求书3页 说明书22页 附图18页

(54)发明名称

用于相干消息的高带宽链路层

(57)摘要

系统、方法和设备可以包括链路层逻辑，该链路层逻辑用于：由链路层设备识别以第一协议格式从存储器接收的第一数据，由链路层设备识别以第二协议格式从高速缓存接收的第二数据，由链路层设备复用第一数据的部分和第二数据的部分以产生复用数据；以及由链路层设备生成包括复用数据的流控制单元(flit)。



CN 110442534 A

1.一种装置,包括:

存储器设备;

高速缓存设备;以及

链路层组件,其耦合到所述存储器设备和所述高速缓存设备,所述链路层组件用于:

识别从所述存储器设备接收的第一数据,所述第一数据采用第一协议格式;

识别从所述高速缓存设备接收的第二数据,所述第二数据采用第二协议格式;以及

基于所述第一数据和所述第二数据来生成用于发送的流控制单元(flit),其中,所述flit的大小为528比特。

2.如权利要求1所述的装置,其中,生成所述flit包括:复用所述第一数据的部分和所述第二数据的部分以生成所述flit。

3.如权利要求1所述的装置,其中,所述flit包括用于有效载荷的512比特和用于循环冗余校验的16比特。

4.如权利要求3所述的装置,其中,所述512比特由所述第一数据的部分和所述第二数据的部分组成。

5.如权利要求1所述的装置,其中,所述flit被划分为相等大小的时隙,并且其中,所述flit的第一时隙包含报头信息。

6.如权利要求1所述的装置,其中,所述链路层组件包括发送器链路层和接收器链路层,并且其中,所述发送器链路层维护翻转,并且所述接收器链路层维护与所述翻转相对应的影子翻转。

7.如权利要求6所述的装置,其中,所述flit包括全数据flit,其中,所述发送器链路层发送所述flit,并且其中,所述接收器链路层基于所述发送器链路层对所述flit的发送和所述影子翻转来确定何时预期全数据响应flit。

8.如权利要求1所述的装置,其中,所述flit包括全数据flit。

9.如权利要求1所述的装置,其中,所述flit是第一flit,其中,所述第一flit包括所述第一数据的第一部分和所述第二数据的第一部分,其中,所述链路层组件还用于生成第二flit,所述第二flit包括所述第一数据的第二部分和所述第二数据的第二部分,其中,所述第一flit包括与所述第一flit和所述第二flit相关联的报头,其中,所述第二flit包括全数据flit,并且其中,所述第二flit在所述第一flit之后被发送。

10.如权利要求1所述的装置,其中,所述链路层组件包括flit打包器电路。

11.如权利要求1所述的装置,其中,所述链路层组件包括桥接器或交换机。

12.一种方法,包括:

由链路层设备识别以第一协议格式接收的第一数据;

由所述链路层设备识别以第二协议格式接收的第二数据;

由所述链路层设备复用所述第一数据的部分和所述第二数据的部分以产生复用数据;以及

由所述链路层设备生成包括所述复用数据的流控制单元(flit)。

13.如权利要求12所述的方法,其中,所述flit的大小为528比特。

14.如权利要求12所述的方法,其中,所述flit包括用于有效载荷的512比特和用于循环冗余校验的16比特,并且其中,所述有效载荷包括所述复用数据。

15.如权利要求14所述的方法,其中,所述有效载荷由所述复用数据组成。

16.如权利要求12所述的方法,其中,所述flit被划分为相等大小的时隙,并且其中,所述flit的第一时隙包含报头信息。

17.如权利要求12所述的方法,还包括:

由所述链路层的发送器链路层维护翻转;以及

由所述链路层的接收器链路层维护与所述翻转相对应的影子翻转。

18.如权利要求17所述的方法,还包括:

由所述发送器链路层发送所述flit,其中,所述flit包括全数据flit;以及

由所述接收器链路层基于所述发送器链路层对所述flit的发送和所述影子翻转来确定何时预期全数据响应flit。

19.如权利要求12所述的方法,其中,所述flit包括全数据flit。

20.如权利要求12所述的方法,其中,所述第一数据的部分是所述第一数据的第一部分,其中,所述第二数据的部分是所述第二数据的第一部分,其中,所述复用数据是第一复用数据,其中,所述flit是第一flit,并且其中,所述方法还包括:

由所述链路层复用所述第一数据的第二部分和所述第二数据的第二部分以产生第二复用数据;

由所述链路层生成包括所述第二复用数据的第二flit,其中,所述第二flit包括全数据flit,并且其中,所述第一flit包括与所述第一flit和所述第二flit相关联的报头信息;

由所述链路层发送所述第一flit;以及

由所述链路层在所述第一flit之后发送所述第二flit。

21.一种系统,包括:

发送器设备,其包括:

用于存储数据的存储器,

用于缓存数据的高速缓存,

至少部分地以硬件实现的链路层逻辑,以及

至少部分地以硬件实现的复用器逻辑;

接收器设备,其包括:

用于对从所述发送器接收的数据包进行解码的解码器设备;

其中,所述链路层逻辑用于:

由链路层设备识别以第一协议格式从所述存储器接收的第一数据;

由所述链路层设备识别以第二协议格式从所述高速缓存接收的第二数据;

由所述链路层设备复用所述第一数据的部分和所述第二数据的部分以产生复用数据;以及

由所述链路层设备生成包括所述复用数据的流控制单元(flit)。

22.如权利要求21所述的系统,其中,所述flit的大小为528比特。

23.如权利要求21所述的系统,其中,所述flit包括用于有效载荷的512比特和用于循环冗余校验的16比特,并且其中,所述有效载荷包括所述复用数据。

24.如权利要求23所述的系统,其中,所述有效载荷由所述复用数据组成。

25.如权利要求21所述的系统,其中,所述flit被划分为相等大小的时隙,并且其中,所

述flit的第一时隙包含报头信息。

用于相干消息的高带宽链路层

[0001] 相关申请的交叉引用

[0002] 本申请要求享有2018年5月4日提交的美国临时申请第62/667,230号的权益,其全部内容通过引用并入本文。

背景技术

[0003] 在计算中,高速缓存是存储数据因此可以更快地提供针对该数据的未来请求的组件。例如,存储在高速缓存中的数据可能是较早计算的结果,或者是存储在别处的数据的副本。通常,当在高速缓存中找到所请求的数据时,可以发生高速缓存命中,而当在高速缓存中未找到所请求的数据时,可以发生高速缓存未命中。通过从高速缓存读取数据来提供高速缓存命中,这典型地比重新计算结果或从较慢的数据存储库读取更快。因此,经常可以通过从高速缓存提供更多请求来实现效率的提高。

附图说明

[0004] 图1是根据一个实施例的包括用于连接计算机系统中的I/O设备的串行点对点互连的系统的简化框图的示意图。

[0005] 图2是根据一个实施例的分层协议栈的简化框图的示意图。

[0006] 图3是事务描述符的实施例的示意图。

[0007] 图4是串行点对点链路的实施例的示意图。

[0008] 图5是根据本公开的实施例的包括连接的加速器的处理系统的示意图。

[0009] 图6是根据本公开的实施例的示例计算系统的示意图。

[0010] 图7A是根据本公开的实施例的具有固定宽度和16比特循环冗余校验(CRC)的示例flit格式的示意图。

[0011] 图7B是根据本公开的实施例的全数据flit的示意图。

[0012] 图8A是根据本公开的实施例的示例flit打包环境800的示意图。

[0013] 图8B-E示出了根据本公开的实施例的示例flit封装场景。

[0014] 图9A-B是根据本公开的实施例的用于对包括至少一个全数据flit的链式flit进行打包和解码的过程流程图。

[0015] 图10A是根据各种实施例的可以具有多于一个核心、可以具有集成存储器控制器并且可以具有集成图形的处理器的框图。

[0016] 图10B描绘了根据本公开的一个实施例的系统的框图。

[0017] 图11描绘了根据本公开的实施例的更具体的第一示例性系统的框图。

[0018] 图12描绘了根据本公开的实施例的更具体的第二示例性系统1300的框图。

[0019] 图13描绘了根据本公开的实施例的SoC的框图。

[0020] 图14是根据公开内容的实施例的对比使用软件指令变换器将源指令集中的二进制指令变换为目标指令集中的二进制指令的框图。

具体实施方式

[0021] 在以下说明书中阐述了许多具体细节，例如，特定类型的处理器和系统配置、特定硬件结构、特定架构和微架构细节、特定寄存器配置、特定指令类型、特定系统组件、特定处理器管线阶段、特定互连层、特定分组/事务配置、特定事务名称、特定协议交换、特定链路宽度、特定实现方式和操作等的示例，以便提供对本公开的透彻理解。然而，对于本领域技术人员可以显而易见的是，不一定需要采用这些具体细节来实践本公开的主题。在其他实例中，避免对以下已知的组件或方法进行详细描述以免不必要地模糊本公开：例如，特定和替代的处理器架构、用于所描述的算法的特定逻辑电路/代码、特定固件代码、低级别互连操作、特定逻辑配置、特定制造技术和材料、特定编译器实现方式、代码形式的特定算法表达、特定掉电和门控技术/逻辑以及计算机系统的其他特定操作细节。

[0022] 在下面的具体实施方式中，参考形成其一部分的附图，在附图中相同的附图标记始终表示相同的部分，并且附图通过可以实践的说明性实施例的方式示出。应该理解，在不脱离本公开的范围的情况下可以利用其他实施例并且可以进行结构或逻辑改变。因此，以下具体实施方式不应被视为具有限制意义，并且实施例的范围由所附示例及其等同物限定。

[0023] 在所附说明书中公开了公开内容的方面。在不脱离本公开的精神或范围的情况下，可以设想本公开的替代实施例及其等同物。应当注意，下面公开的相同元素在附图中由相同的附图标记指示。

[0024] 各种操作可以以最有助于理解示例的主题的方式依次描述为多个离散的动作或操作。然而，描述的次序不应被解释为暗指这些操作必须是次序相关的。特别地，这些操作可以不按照呈现的次序执行。所描述的操作可以以与所描述的实施例不同的次序来执行。在附加实施例中可以执行各种附加操作和/或可以省略所描述的操作。

[0025] 出于本公开的目的，短语“A和/或B”表示(A)、(B)或(A和B)。出于本公开的目的，短语“A、B和/或C”表示(A)、(B)、(C)、(A和B)、(A和C)、(B和C)或(A、B和C)。

[0026] 说明书可以使用短语“在一个实施例中”或“在实施例中”，其可以各自指代相同实施例或不同实施例中的一个或多个。此外，如关于本公开的实施例使用的术语“包括”、“包含”、“具有”等是同义的。

[0027] 如本文中所使用的，术语“电路”可以指代以下各项，是以下各项的一部分或包括以下各项：专用集成电路(ASIC)、电子电路、执行一个或多个软件程序或固件程序的(共享的、专用的或组)处理器和/或(共享的、专用的或组)存储器、组合逻辑电路和/或提供所描述的功能的其他合适的组件。

[0028] 虽然可以参考特定集成电路中的能量节约、能量效率、处理效率等来描述以下实施例(例如，在计算平台或微处理器中)，但是其他实施例也适用于其他类型的集成电路和逻辑器件。本文描述的实施例的类似技术和教导可以适用于可以同样受益于这些特征的其他类型的电路或半导体器件。例如，所公开的实施例不限于服务器计算机系统、台式计算机系统、膝上型计算机或Ultrabooks™，而且还可以用于其他设备，例如，手持设备、智能电话、平板计算机、其他薄型笔记本计算机、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、数码照相机、个人数字助理(PDA)和手持PC。这里，用于高性能互连的类似技术可以适用于在低功耗互连中增强性能(或甚至节能)。嵌入式应用

典型地包括微控制器、数字信号处理器(DSP)、片上系统、网络计算机(NetPC)、机顶盒、网络中心、广域网(WAN)交换机或可以执行下面教导的功能和操作的任何其他系统。此外，本文描述的装置、方法和系统不限于物理计算设备，而且还可以涉及用于能量节约和效率的软件优化。如可以在下面的说明书中变得显而易见的，可以认为本文描述的方法、装置和系统的实施例(无论是参考硬件、固件、软件还是其组合)对于与性能考虑均衡的“绿色技术”未来是至关重要的。

[0029] 随着计算系统的发展，其中的组件变得更加复杂。用于在组件之间进行耦合和通信的互连架构的复杂性也已经增加，以确保满足最佳组件操作的带宽需求。此外，不同的细分市场需要互连架构的不同方面来满足相应市场。例如，服务器要求更高的性能，而移动生态系统有时能够牺牲整体性能以用于节能。然而，大部分结构的单一目的是提供最高可能性能和最大节能。进一步地，各种不同互连可以潜在地受益于本文描述的主题。

[0030] 外围组件互连(PCI)快速(PCIe)互连结构架构和快速路径互连(QPI)结构架构以及其他示例可以潜在地根据本文描述的一个或多个原理以及其他示例来改进。例如，PCIe的主要目标是使得来自不同供应商的组件和设备能够在开放式架构中互操作，跨越多个细分市场；客户端(台式和移动)、服务器(标准和企业)以及嵌入式设备和通信设备。PCI快速是高性能通用I/O互连，其定义用于各种未来计算和通信平台。一些PCI属性(例如，其使用模型，加载-存储架构和软件接口)已经通过其修订进行维护，而先前的并行总线实现方式已经由高度可扩展的完全串行接口取代。最近期的PCI快速版本利用点对点互连、基于交换机的技术和分组化的协议中的改进，实现新水平的性能和特征。PCI快速支持的高级特征中的一些是功率管理、服务质量(QoS)、热插拔/热调换支持、数据完整性以及错误处理。虽然本文的主要讨论内容参考新的高性能互连(HPI)架构，但是本文描述的公开内容的方面可以适用于其他互连架构，例如，符合PCIe的架构、符合QPI的架构、符合MIPI的架构、高性能架构或其他已知的互连架构。

[0031] 参考图1，示出了由互连一组组件的点对点链路组成的结构的实施例。系统100包括耦合到控制器中心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，例如，微处理器、主机处理器、嵌入式处理器、协处理器或其他处理器。处理器105通过前侧总线(FSB)106耦合到控制器中心115。在一个实施例中，FSB 106是如下面描述的串行点对点互连。在另一实施例中，链路106包括符合不同互连标准的串行差分互连架构。

[0032] 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)存储器或系统100中的设备可访问的其他存储器。系统存储器110通过存储器接口116耦合到控制器中心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存储器接口以及动态RAM(DRAM)存储器接口。

[0033] 在一个实施例中，控制器中心115可以包括例如PCIe互连层级中的根中心、根复合体或根控制器。控制器中心115的示例包括芯片组、存储器控制器中心(MCH)、北桥、互连控制器中心(ICH)、南桥和根控制器/中心。经常，术语芯片组指代两个物理上分离的控制器中心，例如，耦合到互连控制器中心(ICH)的存储器控制器中心(MCH)。注意，当前系统经常包括与处理器105集成的MCH，而控制器115以与下面描述的类似的方式与I/O设备通信。在一些实施例中，可选地通过根复合体115支持对等路由。

[0034] 这里，控制器中心115通过串行链路119耦合到交换机/桥接器120。输入/输出模块

117和121(也可以称为接口/端口117和121)可以包括/实现分层协议栈,以提供控制器中心115与交换机120之间的通信。在一个实施例中,多个设备能够耦合到交换机120。

[0035] 交换机/桥接器120将分组/消息从设备125向上游(即,朝向根复合体向上的层级)路由到控制器中心115,以及从处理器105或系统存储器110向下游(即,远离根控制器向下的层级)路由到设备125。在一个实施例中,交换机120被称为多个虚拟PCI到PCI桥接器设备的逻辑组件。设备125包括要耦合到电子系统的任何内部或外部的设备或组件,例如,I/O设备、网络接口控制器(NIC)、插入卡、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、Firewire设备、通用串行总线(USB)设备、扫描仪以及其他输入/输出设备。经常,在PCIe中,诸如设备之类的术语被称为端点。虽然没有具体示出,但是设备125可以包括桥接器(例如,PCIe到PCI/PCI-X桥接器),以支持传统设备或其他版本的设备,或者这些设备支持的互连结构。

[0036] 图形加速器130也可以通过串行链路132耦合到控制器中心115。在一个实施例中,图形加速器130耦合到MCH,该MCH耦合到ICH。然后将交换机120以及因此I/O设备125耦合到ICH。I/O模块131和118还用于实现分层协议栈,以在图形加速器130与控制器中心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本身可以集成在处理器105中。

[0037] 转到图2,示出了分层协议栈的实施例。分层协议栈200可以包括任何形式的分层通信栈,例如,QPI栈、PCIe栈、下一代高性能计算互连(HPI)栈或其他分层栈。在一个实施例中,协议栈200可以包括事务层205、链路层210和物理层220。接口(例如,图1中的接口117、118、121、122、126和131)可以表示为通信协议栈200。作为通信协议栈的表示也可以称为实现/包括协议栈的模块或接口。

[0038] 分组可以用于在组件之间传送信息。可以在事务层205和数据链路层210中形成分组,以将信息从发送组件携带到接收组件。当被发送的分组流过其他层时,这些分组利用用于在这些层处处理分组的附加信息进行扩展。在接收侧,发生反向过程,并且分组从其物理层220表示变换为数据链路层210表示,并且最后(对于事务层分组)变换为可以由接收设备的事务层205处理的形式。

[0039] 在一个实施例中,事务层205可以提供设备的处理核心与互连架构(例如,数据链路层210和物理层220)之间的接口。在这方面,事务层205的主要职责可以包括分组(即,事务层分组或TLP)的组包和拆包。事务层205还可以管理针对TLP的基于信用的流控制。在一些实现方式中,可以利用分离事务,即,请求和响应按时间分开的事务,允许链路在目标设备收集用于响应的数据时携带其他业务,以及其他示例。

[0040] 基于信用的流控制可以用于利用互连结构来实现虚拟信道和网络。在一个示例中,设备可以在事务层205中通告接收缓冲器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(例如,图1中的控制器中心115)可以对由每个TLP消耗的信用的数量进行计数。如果事务未超过信用限制,则可以发送事务。在接收到响应时,恢复一定量的信用。这样的信用方案的优点的一个示例在于,只要没有遇到信用限制,则信用返回的延迟不会影响性能,以及其他潜在优点。

[0041] 在一个实施例中,四个事务地址空间可以包括配置地址空间、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用于将数据传送到存储器映射的

位置或从存储器映射的位置传送数据的读取请求和写入请求中的一个或多个。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(例如，32比特地址)或长地址格式(例如，64比特地址)。配置空间事务可以用于访问连接到互连的各种设备的配置空间。针对配置空间的事务可以包括读取请求和写入请求。消息空间事务(或简称为消息)还可以被定义为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可以对分组报头/有效载荷206进行组包。

[0042] 数据链路层设备的示例可以包括桥接器、中继器、交换机等。桥接器可以类似于中继器，具有通过读取源和目的地的MAC地址来过滤内容的附加功能。桥接器还用于互连在相同协议(或者在一些实施例中，复用到单个flit中的不同协议)上工作的两个设备或系统或网络。交换机是具有缓冲器并且具有可以提高其效率(大量端口意味着更少的流量)和性能的设计的多端口桥接器。交换机是数据链路层设备。交换机可以在转发数据之前执行错误检查，这使得交换机非常高效，因为它不转发有错误的分组并且仅选择性地将好的分组转发到正确的端口。换言之，交换机划分主机的冲突域，但广播域保持相同。

[0043] 快速参考图3，示出了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可以是用于携带事务信息的机制。在这方面，事务描述符300支持对系统中的事务的识别。其他潜在用途包括跟踪对默认事务排序的修改以及事务与信道的关联。例如，事务描述符300可以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对于所有未完成的请求是唯一的。

[0044] 根据一种实现方式，本地事务标识符字段308是由请求代理生成的字段，并且其对于对该请求代理而言要求完成的所有未完成的请求可以是唯一的。此外，在该示例中，源标识符310唯一地标识互连层级内的请求者代理。因此，与源ID 310一起，本地事务标识符字段308提供层级域内的事务的全局标识。

[0045] 属性字段304指定事务的特性和关系。在这方面，属性字段304潜在地用于提供允许修改对事务的默认处理的附加信息。在一个实施例中，属性字段304包括优先级字段312、预留字段314、排序字段316和非监听字段318。这里，优先级子字段312可以由发起者修改以向事务指派优先级。预留属性字段314被预留以供将来使用或供应商定义的使用。可以使用预留属性字段来实现使用优先级或安全性属性的可能使用模型。

[0046] 在该示例中，排序属性字段316用于提供传达可以修改默认排序规则的排序类型的可选信息。根据一个示例实现方式，排序属性“0”表示要应用默认排序规则，其中排序属性“1”表示不严格的排序，其中写入可以在相同方向上传递写入，并且读取完成可以在相同方向上传递写入。监听属性字段318用于确定事务是否被监听。如所示出的，信道ID字段306标识事务与其相关联的信道。

[0047] 返回至图2的讨论，链路层210(也称为数据链路层210)可以用作事务层205与物理层220之间的中间阶段。在一个实施例中，数据链路层210的职责是提供用于在链路上的两个组件之间交换事务层分组(TLP)的可靠机制。数据链路层210的一侧接受由事务层205组包的TLP，应用分组序列标识符211(即，标识号或分组号)，计算并应用错误检测码(即，CRC 212)，并且将修改后的TLP提交给物理层220以用于跨物理到外部设备的传输。

[0048] 在一个示例中，物理层220包括逻辑子块221和电子块222，以物理地将分组发送到

外部设备。这里，逻辑子块221负责物理层220的“数字”功能。在这方面，逻辑子块可以包括用于准备传出信息以供由物理子块222进行发送的发送部分，以及用于在将接收到的信息传递到链路层210之前识别并准备接收到的信息的接收器部分。

[0049] 物理块222包括发送器和接收器。发送器由逻辑子块221提供符号，发送器将符号串行化并发送到外部设备上。向接收器提供来自外部设备的串行化符号，并且接收器将接收到的信号变换为比特流。比特流被解串行化并提供给逻辑子块221。在一个示例实施例中，采用8b/10b传输码，其中发送/接收十比特符号。这里，特殊符号用于利用帧223将分组组成帧。另外，在一个示例中，接收器还提供从传入的串行流中恢复的符号时钟。

[0050] 如上面陈述的，虽然参考协议栈的特定实施例(例如，PCIe协议栈)讨论了事务层205、链路层210和物理层220，但是分层协议栈不限于此。实际上，任何分层协议可以被包括/实现，并且适用本文讨论的特征。作为示例，表示为分层协议的端口/接口可以包括：(1)用于对分组进行组包的第一层，即，事务层；用于对分组进行排序的第二层，即，链路层；以及用于传输分组的第三层，即，物理层。作为具体示例，使用如本文描述的高性能互连分层协议。

[0051] 接下来参考图4，示出了串行点对点结构的示例实施例。串行点对点链路可以包括用于传输串行数据的任何传输路径。在所示的实施例中，链路可以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包括用于将数据发送到设备410的发送逻辑406以及用于从设备410接收数据的接收逻辑407。换言之，在链路的一些实现方式中，包括两个发送路径(即，路径416和417)以及两个接收路径(即，路径418和419)。

[0052] 传输路径指代用于传输数据的任何路径，例如，传输线路、铜线路、光线路、无线通信信道、红外通信链路或其他通信路径。两个设备(例如，设备405和设备410)之间的连接被称为链路，例如，链路415。链路可以支持一个通道——每个通道表示一组差分信号对(一对用于发送，一对用于接收)。为了扩展带宽，链路可以聚合由xN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。

[0053] 差分对可以指代两个传输路径，例如，线路416和417，用于发送差分信号。作为示例，当线路416从低电压电平切换到高电压电平(即，上升沿)时，线路417从高逻辑电平驱动到低逻辑电平(即，下降沿)。差分信号潜在地表现出更好的电特性，例如，更好的信号完整性，即，交叉耦合、电压过冲/下冲、振铃以及其他示例优点。这允许更好的定时窗口，其实现更快的传输频率。

[0054] INTEL®加速器链路(IAL)或其他技术(例如，GenZ、CAPI)定义了通用存储器接口，其允许与诸如加速器之类的分立设备相关联的存储器用作相干存储器。在许多情况下，分立设备和相关联的存储器可以是连接的卡，或者在与(多个)核心处理器分开的机箱中。引入与设备相关联的相干存储器的结果是设备存储器不会与CPU或平台紧密耦合。不能预期平台特定的固件知晓设备细节。出于模块化和互操作性的原因，存储器初始化职责必须在平台特定的固件和设备特定的固件/软件之间公平划分。

[0055] 本公开描述了对现有Intel加速器链路(IAL)架构的扩展。IAL使用称为IAL.io、IAL.cache和IAL.mem的三种单独协议的组合来实现IAL的基于偏差的相干性模型(以下称为相干偏差模型)。相干偏差模型可以促进加速器中的高性能，同时将相干开销最小化。

[0056] IAL.io是IAL用于诸如发现、配置、初始化、中断、错误处理、地址转换服务等之类

的功能的PCIe兼容的输入/输出(IO)协议。IAL.io本质上是非相干的,支持可变有效载荷大小并且遵循PCIe排序规则。IAL.io的功能类似于Intel片上系统结构(IOSF)。IOSF是重新封装用于复用的PCIe协议,用于发现、寄存器访问、中断等。

[0057] IAL.mem是主机用于从设备附接的存储器访问数据的I/O协议。IAL.mem允许将设备附接的存储器映射到系统相干地址空间。IAL.mem还具有监听和元数据语义,以管理设备侧高速缓存的相干性。IAL.mem类似于控制存储器流的SMI3。

[0058] IAL.cache是设备用于从主机附接的存储器请求可缓存数据的I/O协议。IAL.cache是非报告(non-posted)且无序的,并且支持高速缓存行粒度有效载荷大小。IAL.cache类似于用于相干请求和存储器流的管芯内互连(IDI)协议。

[0059] 本公开使用IAL附接的存储器(IAL.mem协议)作为示例实现方式,但是也可以扩展到其他技术,例如,由GenZ联盟或者CAPI或OpenCAPI规范、CCIX、NVLink等发布的那些技术。IAL建立在PCIe之上,并且加入了对相干存储器附接的支持。然而,一般而言,本文描述的系统、设备和程序可以使用促进相干存储器的附接的其他类型的输入/输出总线。

[0060] 图5是根据本公开的实施例的包括连接的加速器的处理系统500的示意图。处理系统500可以包括主机设备501和连接的设备530。连接的设备530可以是跨基于IAL的互连来连接或通过另一类似互连来连接的分立设备。连接的设备530可以与主机设备501集成在同一机箱内,或者可以容纳在单独的机箱中。

[0061] 主机设备501可以包括处理器核心502(标记为CPU 502)。处理器核心502可以包括一个或多个硬件处理器。处理器核心502可以耦合到存储器模块505。存储器模块505可以包括双倍数据速率(DDR)交错式存储器,例如,双列直插式存储器模块DIMM1 506和DIMM2 508,但是也可以包括更多存储器和/或其他存储器类型。主机设备501可以包括以硬件、软件或固件中的一个或组合实现的存储器控制器504。存储器控制器504可以包括逻辑电路,用于管理去往以及来自主机设备501和存储器模块505的数据流。

[0062] 连接的设备530可以跨互连耦合到主机设备501。作为示例,连接的设备530可以包括加速器ACC1 532和ACC2 542。ACC1 532可以包括能够控制相干存储器ACC1_MEM 536的存储器控制器MC1 534。ACC2 542可以包括能够控制相干存储器ACC2_MEM 546的存储器控制器MC2 544。连接的设备530可以包括另外的加速器、存储器等。ACC1_MEM 536和ACC2_MEM 546可以是主机处理器使用的相干存储器;同样,存储器模块505也可以是相干存储器。ACC1_MEM 536和ACC2_MEM 546可以是或包括主机管理的设备存储器(HDM)。

[0063] 主机设备501可以包括用于执行一个或多个存储器初始化过程的软件模块520。软件模块520可以包括操作系统(OS)522、平台固件(FW)524、一个或多个OS驱动程序526以及一个或多个EFI驱动程序528。软件模块520可以包括在体现在非暂时性机器可读介质上的逻辑,并且可以包括当被执行时使一个或多个软件模块对相干存储器ACC1_MEM 536和ACC2_MEM 546进行初始化的指令。

[0064] 例如,平台固件524可以通过标准硬件寄存器或使用指定的供应商特定扩展能力寄存器(DVSEC)在启动期间提前确定相干存储器ACC1_MEM536和ACC2_MEM 546的大小以及存储器的总特性。平台固件524将设备存储器ACC1_MEM 536和ACC2_MEM 546映射到相干地址空间。设备固件或软件550执行设备存储器初始化并发信号通知平台固件524和/或系统软件520(例如,OS 522)。然后,设备固件550经由软件协议将详细的存储器特性传送到平台

固件524和/或系统软件520(例如,OS 522)。

[0065] 图6示出了可以表示各种实施例的操作环境600的示例。图6中描绘的操作环境600可以包括可操作以提供处理能力和/或存储器能力的设备602。例如,设备602可以是经由互连650通信地耦合到主机612的加速器或处理器设备,互连650可以是单个互连、总线、迹线等。设备602和主机612可以通过链路650进行通信,以使数据和消息能够在其间传递。在一些实施例中,链路650可操作以用于支持多种协议以及经由多种互连协议进行的数据和消息的通信。例如,链路650可以支持各种互连协议,包括但不限于非相干互连协议、相干互连协议和存储器互连协议。支持的互连协议的非限制性示例可以包括PCI、PCIe、USB、IDI、IOSF、SMI、SMI3、IAL.io、IAL.cache和IAL.mem等。例如,链路650可以支持相干互连协议(例如,IDI)、存储器互连协议(例如,SMI3)和非相干互连协议(例如,IOSF)。

[0066] 在实施例中,设备602可以包括加速器逻辑604,其包括电路605。在一些实例中,加速器逻辑604和电路605可以提供处理能力和存储器能力。在一些实例中,加速器逻辑604和电路605可以结合主机612提供的处理能力来提供附加处理能力。设备602的示例可以包括生产者-消费者设备、生产者-消费者加设备、软件辅助设备存储器设备、自主设备存储设备和大型高速缓存设备,如先前讨论的。加速器逻辑604和电路605可以基于设备来提供处理能力和存储器能力。例如,加速器逻辑604和电路605可以使用互连经由接口逻辑606和电路607与主机612进行通信,该互连使用例如用于诸如相干请求和存储器流之类的各种功能的相干互连协议(例如,IDI)。接口逻辑606和电路607可以基于用于通信的消息和数据来确定互连协议。在另一示例中,加速器逻辑604和电路605可以包括相干逻辑,该相干逻辑包括或访问偏置模式信息。包括相干逻辑的加速器逻辑604可以使用存储器互连协议(例如,SMI3)经由接口逻辑606和电路607与主机612传送访问偏置模式信息以及相关的消息和数据。接口逻辑606和电路607可以基于用于通信的数据和消息来确定使用存储器互连协议。

[0067] 此外,接口逻辑606、电路607或设备602的其他部分可以包括翻转(rollover)计数器,以对翻转值进行计数。翻转值是由设备602发送器和接收器(即,接口逻辑606和电路607的TX组件和RX组件)的链路层维护的计数。翻转值是与数据传输相关联的离散化数据量的计数。

[0068] 在一些实施例中,加速器逻辑604和电路605可以包括并处理利用诸如基于结构的协议(例如,IOSF)和/或外围组件互连快速(PCIe)协议之类的非相干互连的指令。在各种实施例中,非相干互连协议可以用于各种功能,包括但不限于发现、寄存器访问(例如,设备602的寄存器)、配置、初始化、中断、直接存储器访问和/或地址转换服务(ATS)。注意,设备602可以包括各种加速器逻辑604和电路605以处理信息,并且可以基于设备的类型,例如,生产者-消费者设备、生产者-消费者加设备、软件辅助设备存储器设备、自主设备存储器设备和大型高速缓存设备。此外,并且如先前讨论的,取决于设备的类型,包括接口逻辑606、电路607、(多个)协议队列609和多协议复用器608的设备602可以根据一种或多种协议来进行通信,例如,非相干协议、相干协议和存储器互连协议。实施例不限于此方式。

[0069] 在各种实施例中,主机612可以类似于处理器105,如图1中所讨论的,并且包括类似或相同的电路以提供类似的功能。主机612可以可操作地耦合到主机存储器626,并且可以包括相干逻辑(或相干和高速缓存逻辑)614,相干逻辑614可以包括高速缓存层级并且具有较低级别高速缓存(LLC)。相干逻辑614可以使用各种互连与包括电路623的接口逻辑622

和一个或多个核心618a-n进行通信。在一些实施例中，相干逻辑614可以经由相干互连协议和存储器互连协议中的一个或多个来实现通信。在一些实施例中，相干LLC可以包括主机存储器626和加速器存储器610的至少一部分的组合。实施例不限于此方式。

[0070] 主机612可以包括总线逻辑616，总线逻辑616可以是或可以包括PCIe逻辑。在各种实施例中，总线逻辑616可以使用非相干互连协议(例如，IOSF)和/或外围组件互连快速(PCIe或PCI-E)协议通过互连进行通信。在各种实施例中，主机612可以包括多个核心618a-n，每个核心具有高速缓存。在一些实施例中，核心618a-n可以包括Intel®架构(IA)核心。核心618a-n中的每一个可以经由互连与相干逻辑614通信。在一些实施例中，与核心618a-n以及相干和高速缓存逻辑614耦合的互连可以支持相干互连协议(例如，IDI)。在各种实施例中，主机处理器可以包括可操作以通过互连与总线逻辑616通信的设备620。在一些实施例中，设备620可以包括I/O设备，例如，PCIe I/O设备。

[0071] 在实施例中，主机612可以包括接口逻辑622和电路623，以实现主机612的组件与设备602之间的多协议通信。接口逻辑622和电路623可以根据诸如非相干互连协议、相干互连协议和存储器互连协议之类的一种或多种互连协议动态地处理和实现主机612与设备602之间的消息和数据的通信。在实施例中，接口逻辑622和电路623可以支持能够根据多种互连协议动态地处理数据和消息的单个互连、链路或总线。

[0072] 在一些实施例中，接口逻辑622可以耦合到多协议复用器624，多协议复用器624具有一个或多个协议队列625以与设备602发送和接收消息和数据，设备602包括多协议复用器608并且也具有一个或多个协议队列609。协议队列609和625可以是协议特定的。因此，每种互连协议可以与特定协议队列相关联。接口逻辑622和电路623可以处理利用多协议复用器624从设备602接收以及发送到设备602的消息和数据。例如，当发送消息时，接口逻辑622和电路623可以基于该消息根据互连协议之一来处理消息。接口逻辑622和电路623可以将消息发送到多协议复用器624和链路控制器。多协议复用器624或仲裁器可以将消息存储在协议队列625中，协议队列625可以是协议特定的。基于在设备602处的多协议复用器608处的协议队列609的协议特定的协议队列中的资源可用性，多协议复用器624和链路控制器可以确定何时将消息发送到设备602。当接收消息时，多协议复用器624可以基于该消息将消息放置在队列625的协议特定的队列中。接口逻辑622和电路623可以根据互连协议之一来处理消息。

[0073] 在实施例中，接口逻辑622和电路623可以动态地处理去往和来自设备602的消息和数据。例如，接口逻辑622和电路623可以确定每个消息的消息类型，并且确定多种互连协议中的哪种互连协议用于处理消息中的每个消息。可以使用不同的互连协议来处理消息。

[0074] 另外，接口逻辑622、电路623或主机612的其他部分可以包括翻转计数器，以对翻转值进行计数。翻转值是由设备612发送器和接收器(即，接口逻辑622和电路623的TX组件和RX组件)的链路层维护的计数。翻转值是与数据传输相关联的离散化数据量的计数。

[0075] 在示例中，接口逻辑622可以检测要经由互连650传送的消息。在实施例中，消息可能已经由核心618或另一I/O设备620生成并且用于传送到设备602。接口逻辑622可以确定消息的消息类型，例如，非相干消息类型、相干消息类型和存储器消息类型。在一个特定示例中，接口逻辑622可以基于在地址映射中的查找来确定诸如请求之类的消息是针对耦合的设备的I/O请求还是存储器请求。如果与消息相关联的地址映射为I/O请求，则接口逻辑

622可以利用非相干互连协议来处理消息，并且将消息作为非相干消息发送到链路控制器和多协议复用器624以用于传送到耦合的设备。多协议复用器624可以将消息存储在协议队列625的互连特定的队列中，并且当资源在设备602处可用时使消息被发送到设备602。在另一示例中，接口逻辑622可以基于地址表中的查找来确定与消息相关联的地址指示该消息是存储器请求。接口逻辑622可以利用存储器互连协议来处理消息，并且将消息发送到链路控制器和多协议复用器624以用于传送到耦合的设备602。多协议复用器624可以将消息存储在协议队列625的互连协议特定的队列中，并且当资源在设备602处可用时使消息被发送到设备602。

[0076] 在另一示例中，接口逻辑622可以基于执行的一个或多个高速缓存相干性和存储器访问动作来确定消息是相干消息。更具体地，主机612可以接收由耦合的设备602源送的相干消息或请求。可以执行高速缓存相干性和存储器访问动作中的一个或多个来处理消息，并且基于这些动作；接口逻辑622可以确定响应于请求而发送的消息可以是相干消息。接口逻辑622可以根据相干互连协议来处理消息，并且将相干消息发送到链路控制器和多协议复用器624以便发送到耦合的设备602。多协议复用器624可以将消息存储在队列625的互连协议特定的队列中，并且当资源在设备602处可用时使消息被发送到设备602。实施例不限于此方式。

[0077] 在一些实施例中，接口逻辑622可以基于与消息相关联的地址、由消息引起的动作、消息内的信息(例如，标识符)、消息的源、消息的目的地等来确定消息的消息类型。接口逻辑622可以基于该确定来处理接收到的消息，并且将消息发送到主机612的适当组件以进行进一步处理。接口逻辑622可以基于该确定来处理要发送到设备602的消息，并且将消息发送到链路控制器(未示出)和多协议复用器624以进行进一步处理。可以针对从主机612发送的消息或由主机612接收的消息来确定消息类型。

[0078] 当前的IAL架构可以使用称为IAL.io、IAL.cache和IAL.mem的三种单独协议的组合来实现IAL的基于偏差的相干性模型(此后称为“相干偏差模型”)。相干偏差模型可以促进加速器实现高性能，同时将相干开销最小化。

[0079] 在实施例中，IAL架构可以支持如下定义的至少五种类型的加速器模型。

[0080]

加速器类	描述	示例
生产者-消费者	基本 PCIe 设备	网络 加速器 密码 压缩
生产者-消费者加	具有附加能力的 PCIe 设备 示例：特殊数据操作，例如，原子化	Storm Lake 数据中心结构 Infiniband HBA
SW 辅助设备存储器	具有附接的存储器使用的加速器，其中软件“数据放置”是实际的	分立 FPGA 图形
自主设备存储器	具有附接的存储器使用的加速器，其中软件“数据放置”不是实际的	密集计算卸载 GPGPU
大型高速缓存	具有附接的存储器使用的加速器，其中数据占位面积大于附接的存储器	密集计算卸载 GPGPU

[0081] 本公开描述了用于高速缓存行大小固定的(例如,64B)相干数据传输的高效率(例如,>90%)链路层的系统、设备和方法。这种效率将可实现的链路带宽最大化,同时保持低延迟。基于本公开的方面,由于以下4个因素中的一个或多个,IAL可以实现高效率和低延迟:

[0082] 1.高流量控制单元(flit)宽度:IAL flit的大小可以是528比特,与输入/输出(I/O)链路宽度无关,从而在更大的有效载荷上分摊循环冗余校验(CRC)的成本。

[0083] 2.对多种协议的子flit复用:可以采用在同一个flit上动态地混合和匹配来自多种协议的消息的能力。

[0084] 3.全数据flit格式:可以采用发送没有报头的flit的能力。

[0085] 4.多数据报头:可以采用将多个不相关的数据传输链接在一起的能力,这针对每次传输分摊报头开销。

[0086] 链路带宽是性能指标,通过它可以评估竞争标准。I/O标准(例如,外围组件互连快速(PCIe))可以支持高达4千字节(KB)的有效载荷大小,这有助于I/O标准实现大约95%的效率。然而,相干传输的有效载荷大小可以限于小得多的高速缓存行粒度(64B)。这可能是链路效率和最大可实现带宽的限制因素。本文描述的技术可以显著地减小PCIe级别链路效率与相干传输之间的差距。高效率链路(例如,本文描述的那些)可以用于带宽敏感的应用中,例如,直接存储器访问(DMA)。在下表中,BW可以指代带宽,并且GT/s可以指代千兆传输每秒。

[0087]

传输类型	链路效率	BW(32GT/s链路)
IAL.$主机到设备数据	94%	60.4
IAL.$设备到主机数据	94%	60.4
IAL.Mem从端到主端数据	91%	58.5
IAL.Mem主端到从端数据	80%	51.2

[0088] 具有16比特CRC的固定Flit宽度

[0089] 图7A中示出了通用IAL.cache和IAL.mem流控制单元(flit)格式的示例。图7A是根据本公开的实施例的具有固定宽度和16比特循环冗余校验(CRC)的示例flit格式700的示意图。在实施例中,IAL.cache和IAL.mem flit大小可以固定为528比特,其中512比特是有效载荷702,并且16比特是滚动的循环冗余校验(CRC)704。flit大小可以不改变,与物理链路宽度或分叉无关。具有16比特CRC的固定flit大小在大的有效载荷上分摊CRC的成本,而不会影响错误检测或校正能力。可以检测所有单比特、双比特和三比特错误以及一些4比特错误,并且可以使用重放对这些错误进行校正。

[0090] 对多种协议的子flit复用

[0091] 如可以从图7A中看出的,可以将flit划分为相等大小的时隙(例如,报头时隙712以及通用时隙714、716和718)。报头时隙712可以包含报头信息,包括包含在报头时隙的其余部分以及该flit中的其他时隙中的协议级别消息的定义。IAL.cache和IAL.mem协议可以共享公共链路层。来自两种协议的消息和数据传输可以以动态打包格式复用到共享flit中,该动态打包格式允许对flit时隙进行低粒度共享,并且同时足够灵活以允许单个活动协议获得全带宽。由于来自两种协议的消息被打包到子时隙粒度中,因此可以允许最大flit利用率。

[0092] 简要返回图6,多协议MUX 608可以用于复用来自IAL.cache协议和IAL.mem协议两者(或在其他实施例中,IDI协议和SMI3+协议或其他类似协议)的数据和消息。

[0093] 全数据flit格式

[0094] 对于高效率数据传输,IAL.cache和IAL.mem可以使用没有报头时隙的flit格式,如图7B所示。图7B是根据本公开的实施例的全数据flit 750的示意图。每个时隙762、764、766和768仅包含数据而不携带任何报头,并且除了CRC之外,全数据flit 750可以将链路利用率最大化。然而,由于接收器没有接收到用于对全数据flit 750解码的报头,因此出于此目的描述了跟踪机制。跟踪机制可以基于“翻转”的概念。翻转是与给定传输相关联的数据块的计数,该计数可以由发送器(Tx)链路层和接收器(Rx)链路层两者来维护。每当Tx要发送512b数据时,如翻转计数指示的,Tx可以发送全数据flit 750。类似地,由于Rx对它接收的消息保持跟踪,所以Rx可以维护Tx的翻转计数的影子。使用该计数,可以确定性地知晓何时预期全数据flit 750。Tx组件和Rx组件可以是接口硬件和软件的一部分,例如,接口逻辑606和接口逻辑622。

[0095] 接收器(Rx)将接收只具有数据的flit(即,没有报头信息的全数据flit)。Rx必须具有帮助接收器理解它接收的内容的一些信息。RX维护翻转,翻转是由rx和tx两者维护的与给定传输相关联的数据块数量的计数。每当Tx要发送512b数据时,如Tx的翻转计数指示的,Tx发送全数据flit 750。由于该规则,Rx例如通过维护Tx的翻转的影子计数来对它接收的消息保持跟踪。Rx可以用于翻转以在包括报头的先前传输与作为全数据flit的后续传输之间进行区分。这就是Rx如何可以在没有附加信息的情况下对全数据flit进行解码。当数据跨Tx与Rx之间的链路传输时,翻转计数动态地变化。

[0096] 在重放的场景期间(例如,由于CRC错误),可能需要从全数据flit中对控制flit消除歧义。这可以通过在没有任何混叠可能性的情况下创建接收器可以解码的控制flit的序列来完成。例如,控制flit的序列或系列可以被Rx解释为重放场景。

[0097] 多数据报头(MDH)Flit

[0098] 图8A是根据本公开的实施例的示例flit打包环境800的示意图。在实施例中，IAL.cache和IAL.mem链路层可以具有将多个全数据flit链接在一起的能力。这对于数据流送使用(例如，大型DMA传输)可能很有用。这里的概念是与多个单独传输相关联的数据报头可以紧密挤压到公共报头时隙中。在此之后，所有后续时隙和flit可能只包含与这些报头相关联的数据。这可以在大得多的有效载荷上分摊报头信息的开销。下面示意性地示出了这种情况的示例。每个“DH”表示数据报头，而每个“DC”表示与报头相关联的16字节传输。

[0099] 图8A示出了一组高速缓存数据阵列802和存储器数据阵列812。高速缓存数据阵列802包括高速缓存数据报头阵列804和4x16B高速缓存数据块阵列806。存储器数据阵列812包括存储器数据报头阵列814和4x16B存储器数据块阵列816。高速缓存数据报头阵列804和存储器数据报头阵列814可以各自累积数据事务信息，并且可以取决于其中存在的数据的完整高速缓存行价值的数量而生成“待定”消息或“多个待定”消息。高速缓存数据报头阵列804和存储器数据报头阵列814可以将待定消息或多个待定消息发送到flit打包决策逻辑820。flit打包决策逻辑820(或者flit打包器820)可以以硬件、软件或硬件和软件的组合来实现。

[0100] flit打包器820可以优化使得能够将多个数据报头打包在一起的语义。flit打包器820可以从来自高速缓存数据报头阵列804和存储器数据报头阵列814的接收到的报头信息转换到被定义为携带多个数据报头的多数据报头(MDH)时隙格式类型。MDH时隙格式的示例在图8B中的Flit ID:0 852中示出，其包括数据报头(DH0、DH1、DH2和DH3)。然后，flit打包器820可以基于报头在报头时隙内的布置将高速缓存数据块和存储器数据块封装到空闲时隙中。在一些情况下，flit是全数据flit。例如，Flit 0 852是图8B中的带报头的flit，其包括具有报头DH0、DH1、DH2和DH3的MDH以及DC0_0、DC0_1和DC0_2。Flit 1 854a是全数据flit，其包括数据内容DC0_3、DC1_0、DC1_1和DC1_2。关于图8B的更多细节在下面描述。

[0101] 取决于打包的数据报头的数量，flit打包器必须调度与数据报头相关联的4x16B数据块传输，同时维持其优先次序。这些阵列以这样的方式构建：允许对多达4个数据报头的同时访问能力，以及读取和操纵4x16B数据块中的每一个的单独可控性。

[0102] 链式数据传输可能导致需要通过链路发送一个或多个全数据flit。远程接收器必须解码MDH时隙格式并维护翻转计数器，以确定多少全数据flit将跟随包含数据报头信息的flit。

[0103] 图8B-E示出了根据本公开的实施例的示例flit封装场景850、860、870和880。图8B-E中所示的示例是示例，并且可以适用于覆盖可以打包在一起的数据报头的数量，连同这些数据报头在flit内的位置(时隙#)。

[0104] 图8B示出了根据本公开的实施例的第一示例flit封装场景850。在该第一示例中，发送器将报头和数据块封装到链接在一起的flit中。发送器可以使用flit打包器820将四个数据报头(DH0、DH1、DH2和DH3)封装到带报头的flit的单个时隙(在这种情况下是flit 0 852的时隙0)中。Flit 0852是图8B中的带报头的flit，其包括具有时隙0中的报头DH0、DH1、DH2和DH3的MDH以及时隙1中的DC0_0、时隙2中的DC0_1和时隙3中的DC0_2。Flit1 854a是包括分别在时隙0-3中的数据内容DC0_3、DC1_0、DC1_1和DC1_2的全数据flit。Flit 2 854b和Flit 3 854c也是全数据flit，类似于Flit 1 854a。Flit 4 856是带报头的flit，其包括各种时隙中的其他Txn类型，以及时隙1中的DC3数据(DC3_3)的其余部分。

[0105] 图8C示出了根据本公开的实施例的第二示例flit封装场景860。在该第二示例中，发送器将报头和数据块封装到链接在一起的flit中。发送器可以使用flit打包器820将两个数据报头(DH0和DH1)封装到带报头的flit的单个时隙(在这种情况下是flit 0 862的时隙0)中。Flit 0 862是图8C中的带报头的flit，其包括具有时隙0中的报头DH0和DH1的MDH以及时隙1中的DC0_0、时隙2中的DC0_1和时隙3中的DC0_2。Flit 1 864是包括分别在时隙0-3中的数据内容DC0_3、DC1_0、DC1_1和DC1_2的全数据flit。Flit 3 866是带报头的flit，其包括各种时隙中的其他Txn类型，以及时隙1中的DC1数据(DC1_3)的其余部分。

[0106] 图8D示出了根据本公开的实施例的第三示例flit封装场景870。在该第三示例中，发送器将报头和数据块封装到链接在一起的flit中。发送器可以使用flit打包器820将四个数据报头(DH0、DH1、DH2和DH3)封装到带报头的flit的单个时隙(在这种情况下是flit 0 872的时隙2)中。Flit 0852是图8D中的带报头的flit，其包括具有时隙2中的报头DH0、DH1、DH2和DH3的MDH以及时隙3中的DC0_0。其他Txn数据可以处于时隙0和1中。Flit 1 874a是包括分别在时隙0-3中的数据内容DC0_1、DC0_2、DC0_3和DC1_0的全数据flit。Flit 2 874b和Flit 3 874c也是全数据flit，类似于Flit 1 874a。Flit 4 876是带报头的flit，其包括时隙0中的其他Txn类型，以及时隙1中的DC3_1、时隙2中的DC3_2和时隙3中的DC3_3。

[0107] 图8E示出了根据本公开的实施例的第四示例flit封装场景880。在该第四示例中，发送器将报头和数据块封装到链接在一起的flit中。发送器可以使用flit打包器820将两个数据报头(DH0和DH1)封装到带报头的flit的单个时隙(在这种情况下是flit 0 882的时隙2)中。Flit 0 882是图8E中的带报头的flit，其包括具有时隙2中的报头DH0和DH1的MDH以及时隙3中的DC0_0。其他Txn数据可以处于时隙0和1中。Flit 1 884是包括分别在时隙0-3中的数据内容DC0_1、DC0_2、DC0_3的全数据flit。Flit 2886是带报头的flit，其包括时隙0中的其他Txn类型，以及时隙1中的DC1_1、时隙2中的DC1_2和时隙3中的DC1_3。

[0108] 图9A-B是根据本公开的实施例的用于对包括至少一个全数据flit的链式flit进行打包和解码的过程流程图。

[0109] 图10A-图14是示例性计算机架构的框图。用于膝上型计算机、台式计算机、手持PC、个人数字助理、工程工作站、服务器、网络设备、网络中心、交换机、嵌入式处理器、数字信号处理器(DSP)、图形设备、视频游戏设备、机顶盒、微控制器、蜂窝电话、便携式媒体播放器、手持设备和各种其他电子设备的本领域已知的其他系统设计和配置也适用于执行本公开中描述的方法。通常，能够包含如本文公开的处理器和/或其他执行逻辑的各种系统或电子设备通常是合适的。

[0110] 图10A是根据各种实施例的可以具有多于一个核心、可以具有集成存储器控制器并且可以具有集成图形的处理器1000的框图。图10A中的实线框示出了具有单个核心1002A、系统代理1010和一组一个或多个总线控制器单元1016的处理器1000；而虚线框的可选添加示出了具有多个核心1002A-N、系统代理单元1010中的一组一个或多个集成存储器控制器单元1014以及专用逻辑1008的替代处理器1000。

[0111] 因此，处理器1000的不同实现方式可以包括：1)CPU，其中专用逻辑1008是集成图形和/或科学(吞吐量)逻辑(其可以包括一个或多个核心)，并且核心1002A-N是一个或多个通用核心(例如，通用有序核心、通用无序核心或两者的组合)；2)协处理器，其中核心1002A-N是旨在主要用于图形和/或科学(吞吐量)的大量专用核心；3)协处理器，其中核心

1002A-N是大量通用有序核心。因此，处理器1000可以是通用处理器、协处理器或专用处理器，例如，网络或通信处理器、压缩和/或解压缩引擎、图形处理器、GPGPU(通用图形处理单元)、高吞吐量的许多集成核心(MIC)协处理器(例如，包括30个或更多个核心)、嵌入式处理器或执行逻辑操作的其他固定逻辑或可配置逻辑。处理器可以在一个或多个芯片上实现。处理器1000可以是使用多种工艺技术(例如，BiCMOS、CMOS或NMOS)中的任何工艺技术的一个或多个衬底的一部分，和/或可以在使用多种工艺技术中的任何工艺技术的一个或多个衬底上实现。

[0112] 在各种实施例中，处理器可以包括任何数量的处理元件，这些处理元件可以是对称或不对称的。在一个实施例中，处理元件指代支持软件线程的硬件或逻辑。硬件处理元件的示例包括：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、核心和/或能够保持处理器的状态(例如，执行状态或架构状态)的任何其他元件。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、应用或其他代码)相关联的任何硬件。物理处理器(或处理器插座)典型地指代集成电路，其潜在地包括任何数量的其他处理元件，例如，核心或硬件线程。

[0113] 核心可以指代位于能够维持独立架构状态的集成电路上的逻辑，其中每个独立维持的架构状态与至少一些专用执行资源相关联。硬件线程可以指代位于能够维持独立架构状态的集成电路上的任何逻辑，其中独立维持的架构状态共享对执行资源的访问。如可以看出的，当共享特定资源而其他资源专用于架构状态时，硬件线程与核心的命名之间的界限重叠。但是经常地，操作系统将核心和硬件线程视为单独的逻辑处理器，其中操作系统能够单独地调度每个逻辑处理器上的操作。

[0114] 存储器层级包括核心内的一个或多个级别的高速缓存、一组或一个或多个共享高速缓存单元1006以及耦合到该组集成存储器控制器单元1014的外部存储器(未示出)。该组共享高速缓存单元1006可以包括一个或多个中间级别高速缓存，例如，级别2(L2)、级别3(L3)、级别4(L4)或其他级别的高速缓存、最后一级高速缓存(LLC)和/或其组合。虽然在一个实施例中，基于环的互连单元1012互连专用逻辑(例如，集成图形逻辑)1008、该组共享高速缓存单元1006以及系统代理单元1010/(多个)集成存储器控制器单元1014，但是替代实施例可以使用任何数量的公知技术来互连这些单元。在一个实施例中，在一个或多个高速缓存单元1006与核心1002A-N之间维持相干性。

[0115] 在一些实施例中，核心1002A-N中的一个或多个核心能够是多线程的。系统代理1010包括协调和操作核心1002A-N的那些组件。系统代理单元1010可以包括例如功率控制单元(PCU)和显示单元。PCU可以是或包括调节核心1002A-N和专用逻辑1008的功率状态所需的逻辑和组件。显示单元用于驱动一个或多个外部连接的显示器。

[0116] 就架构指令集而言，核心1002A-N可以是同构的或异构的；也就是说，核心1002A-N中的两个或更多个核心能够执行相同的指令集，而其他核心仅能够执行该指令集的子集或不同的指令集。

[0117] 图10B描绘了根据本公开的一个实施例的系统1050的框图。系统1050可以包括一个或多个处理器1052、1054，其耦合到控制器中心1056。在一个实施例中，控制器中心1056包括图形存储器控制器中心(GMCH)1058和输入/输出中心(IOH)1060(其可以在分开的芯片上或在同一芯片上)；GMCH 1058包括耦合到存储器1062和协处理器1064的存储器控制器和

图形控制器;IOH 1060将输入/输出(I/O)设备1066耦合到GMCH 1058。可替代地,存储器控制器和图形控制器中的一者或两者集成在处理器内(如本文描述的),存储器1062和协处理器1064直接耦合到处理器1052,并且控制器中心1020是包括IOH 1060的单个芯片。

[0118] 附加处理器1054的可选性质在图10B中用短划线表示。每个处理器1052、1054可以包括本文描述的处理核心中的一个或多个处理核心,并且可以是处理器1000的某个版本。

[0119] 存储器1062可以是例如动态随机存取存储器(DRAM)、相变存储器(PCM)、其他合适的存储器或其任何组合。存储器1062可以存储任何合适的数据,例如,由处理器1052、1054用来提供计算机系统1050的功能的数据。例如,与执行的程序相关联的数据或者由处理器1052、1054访问的文件可以存储在存储器1062中。在各种实施例中,存储器1062可以存储由处理器1052、1054使用或执行的数据和/或指令序列。

[0120] 在至少一个实施例中,控制器中心1056经由诸如前端总线(FSB)之类的多点总线、诸如快速路径互连(QPI)之类的点对点接口或类似的连接1095与(多个)处理器1052、1054通信。

[0121] 在一个实施例中,协处理器1064是专用处理器,例如,高吞吐量MIC处理器、网络或通信处理器、压缩和/或解压缩引擎、图形处理器、GPGPU、嵌入式处理器等。在一个实施例中,控制器中心1056可以包括集成图形加速器。

[0122] 就包括架构特性、微架构特性、热特性、功耗特性等在内的优点的度量范围而言,物理资源1052、1054之间可能存在各种差异。

[0123] 在一个实施例中,处理器1052执行控制一般类型的数据处理操作的指令。嵌入在指令内的可以是协处理器指令。处理器1052将这些协处理器指令识别为属于应由附接的协处理器1064执行的类型。因此,处理器1052在协处理器总线或其他互连上向协处理器1064发布这些协处理器指令(或表示协处理器指令的控制信号)。(多个)协处理器1064接受并执行接收到的协处理器指令。

[0124] 图11描绘了根据本公开的实施例的更具体的第一示例性系统1100的框图。如图11所示,多处理器系统1100是点对点互连系统,并且包括经由点对点互连1150耦合的第一处理器1170和第二处理器1180。处理器1170和1180中的每一个可以是处理器的某个版本。在公开内容的一个实施例中,处理器1170和1180分别是处理器1110和1115,而协处理器1138是协处理器1145。在另一实施例中,处理器1170和1180分别是处理器1110和协处理器1145。

[0125] 示出处理器1170和1180分别包括集成存储器控制器(IMC)单元1172和1182。处理器1170还包括点对点(P-P)接口1176和1178,作为其总线控制器单元的一部分;类似地,第二处理器1180包括P-P接口1186和1188。处理器1170、1180可以使用P-P接口电路1178、1188经由点对点(P-P)接口1150交换信息。如图11所示,IMC 1172和1182将处理器耦合到相应的存储器,即,存储器1132和存储器1134,它们可以是本地附接到相应的处理器的主存储器的部分。

[0126] 处理器1170、1180可以各自使用点对点接口电路1176、1194、1186、1198经由各个P-P接口1152、1154与芯片组1190交换信息。芯片组1190可以可选地经由高性能接口1139与协处理器1138交换信息。在一个实施例中,协处理器1138是专用处理器,例如,高吞吐量MIC处理器、网络或通信处理器、压缩和/或解压缩引擎、图形处理器、GPGPU、嵌入式处理器等。

[0127] 共享高速缓存(未示出)可以包括在任一处理器中或者在两个处理器外但是经由

P-P互连与处理器连接，使得如果任一个处理器或两个处理器被置于低功率模式，则处理器的本地高速缓存信息可以存储在共享高速缓存中。

[0128] 芯片组1190可以经由接口1196耦合到第一总线1116。在一个实施例中，第一总线1116可以是外围组件互连(PCI)总线，或者诸如PCI快速总线或另一第三代I/O互连总线之类的总线，但是本公开的范围不限于此。

[0129] 如图11所示，各种I/O设备1114可以连同总线桥1118一起耦合到第一总线1116，总线桥1118将第一总线1116耦合到第二总线1120。在一个实施例中，诸如协处理器、高吞吐量MIC处理器、GPGPU、加速器(例如，图形加速器或数字信号处理(DSP)单元)、现场可编程门阵列或任何其他处理器之类的一个或多个附加处理器1115耦合到第一总线1116。在一个实施例中，第二总线1120可以是低引脚数(LPC)总线。在一个实施例中，各种设备可以耦合到第二总线1120，这些设备包括例如键盘和/或鼠标1122、通信设备1127和可以包括指令/代码和数据1130的诸如磁盘驱动器或其他大容量存储设备之类的存储单元1128。此外，音频I/O 1124可以耦合到第二总线1120。注意，本公开设想到其他架构。例如，代替图11的点对点架构，系统可以实现多点总线或其他这样的架构。

[0130] 图12描绘了根据本公开的实施例的更具体的第二示例性系统1200的框图。图11和图12中的相似元件具有相似的附图标记，并且图12中省略了图11的某些方面，以避免模糊图12的其他方面。

[0131] 图12示出了处理器1270、1280可以分别包括集成存储器和I/O控制逻辑(“CL”)1272和1282。因此，CL 1272、1282包括集成存储器控制器单元并且包括I/O控制逻辑。图12示出了不仅存储器1232、1234耦合到CL 1272、1282，而且I/O设备1214也耦合到控制逻辑1272、1282。传统I/O设备1215耦合到芯片组1290。

[0132] 图13描绘了根据本公开的实施例的SoC 1300的框图。此外，虚线框是更高级的SoC上的可选特征。在图13中，(多个)互连单元1302耦合到：应用处理器1310，其包括一组一个或多个核心1004A-N和(多个)共享高速缓存单元1006；系统代理单元1010；(多个)总线控制器单元1016；(多个)集成存储器控制器单元1014；一组或一个或多个协处理器1320，其可以包括集成图形逻辑、图像处理器、音频处理器和视频处理器；静态随机存取存储器(SRAM)单元1330；直接存储器访问(DMA)单元1332；以及显示单元1340，其用于耦合到一个或多个外部显示器。在一个实施例中，(多个)协处理器1320包括专用处理器，例如，网络或通信处理器、压缩和/或解压缩引擎、GPGPU、高吞吐量MIC处理器、嵌入式处理器等。

[0133] 在一些情况下，指令变换器可以用于将指令从源指令集变换为目标指令集。例如，指令变换器可以将指令转换(例如，使用静态二进制转换，包括动态编译的动态二进制转换)、变形、模拟或以其他方式变换为要由核心处理的一个或多个其他指令。指令变换器可以以软件、硬件、固件或其组合来实现。指令变换器可以在处理器上，在处理器外，或者部分在处理器上而部分在处理器外。

[0134] 图14是根据公开内容的实施例的对比使用软件指令变换器将源指令集中的二进制指令变换为目标指令集中的二进制指令的框图。在所示实施例中，指令变换器是软件指令变换器，但是可替代地，指令变换器可以以软件、固件、硬件或其各种组合来实现。图14示出了可以使用x86编译器1404来编译高级语言1402形式的程序，以生成可以由具有至少一个x86指令集核心的处理器1416在本地执行的x86二进制代码1406。具有至少一个x86指令

集核心的处理器1416表示任何这样的处理器:能够通过兼容地执行或以其他方式处理(1)Intel x86指令集核心的指令集的大部分或(2)以在具有至少一个x86指令集核心的Intel处理器上运行为目标的应用或其他软件的目标代码版本,以便实现与具有至少一个x86指令集核心的Intel处理器基本上相同的结果,来执行与具有至少一个x86指令集核心的Intel处理器基本上相同的功能。x86编译器1404表示可操作以用于生成x86二进制代码1406(例如,目标代码)的编译器,该x86二进制代码1406可以在具有或不具有附加链接处理的情况下在具有至少一个x86指令集核心的处理器1416上执行。类似地,图14示出了可以使用替代指令集编译器1408来编译高级语言1402形式的程序,以生成可以由不具有至少一个x86指令集核心的处理器1414(例如,具有执行MFS Technologies(Sunnyvale,CA)的MIPS指令集和/或执行ARM Holdings(Sunnyvale,CA)的ARM指令集的核心的处理器)在本地执行的替代指令集二进制代码1410。指令变换器1412用于将x86二进制代码1406变换为可以由不具有x86指令集核心的处理器1414在本地执行的代码。该变换后的代码不太可能与替代指令集二进制代码1410相同,因为能够做到这一点的指令变换器很难实现;然而,变换后的代码将完成一般操作,并且由来自替代指令集的指令组成。因此,指令变换器1412表示通过模拟、仿真或任何其他过程来允许不具有x86指令集处理器或核心的处理器或其他电子设备执行x86二进制代码1406的软件、固件、硬件或其组合。

[0135] 设计可以经历从创建到仿真到制造的各种阶段。表示设计的数据可以以多种方式来表示设计。首先,如在仿真中有用的,可以使用硬件描述语言(HDL)或另一种功能描述语言来表示硬件。另外,可以在设计过程的某些阶段产生具有逻辑和/或晶体管栅极的电路级模型。此外,大多数设计在某个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在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)、开放艺术品系统交换标准(OASIS)之类的数据库文件格式或类似格式来存储。

[0136] 在一些实现方式中,基于软件的硬件模型以及HDL和其他功能描述语言对象可以包括寄存器传输语言(RTL)文件以及其他示例。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,以及根据对象来确定物理电路和/或片上布局。设计工具的输出可以用于制造物理设备。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(包括大小和类型)、存储器块、物理链路路径、结构拓扑以及被实现以便实现在HDL对象中建模的系统的其他属性。设计工具可以包括用于确定片上系统(SoC)和其他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对象本身可以作为输入提供给制造系统软件,以使得制造所描述的硬件。

[0137] 在设计的任何表示中,表示设计的数据可以存储在任何形式的机器可读介质中。诸如盘之类的存储器或者磁性或光学存储装置可以是这样的机器可读介质:用于存储经由光波或电波传输的信息,该光波或电波被调制或以其他方式生成以传输这样的信息。当传输指示或携带代码或设计的电载波时,就执行对电信号的复制、缓冲或重传而言,做出新的副本。因此,通信提供商或网络提供商可以至少临时地在有形的机器可读介质上存储物品

(例如,编码成载波、体现本公开的实施例的技术的信息)。

[0138] 在各种实施例中,可以将存储设计的表示的介质提供给制造系统(例如,能够制造集成电路和/或相关组件的半导体制造系统)。设计表示可以指示系统制造能够执行上面描述的功能的任何组合的设备。例如,设计表示可以关于要制造哪些组件、组件应该如何耦合在一起、组件应该放置在设备上的位置和/或关于与要制造的设备有关的其他合适的规范来指示系统。

[0139] 因此,至少一个实施例的一个或多个方面可以通过存储在机器可读介质上的代表性指令来实现,该代表性指令表示处理器内的各种逻辑,其在由机器读取时使得机器制造用于执行本文描述的技术的逻辑。经常称为“IP核心”的这种表示可以存储在非暂时性有形机器可读介质上,并且提供给各种客户或制造设施,以加载到制造逻辑或处理器的制造机器中。

[0140] 本文公开的机制的实施例可以以硬件、软件、固件或这些实现方法的组合来实现。公开内容的实施例可以被实现为在可编程系统上执行的计算机程序或程序代码,该可编程系统包括至少一个处理器、存储系统(包括易失性和非易失性的存储器和/或存储元件)、至少一个输入设备以及至少一个输出设备。

[0141] 诸如图11中所示的代码1130之类的程序代码可以应用于输入指令以执行本文描述的功能并生成输出信息。输出信息可以以已知的方式应用于一个或多个输出设备。出于本申请的目的,处理系统包括具有诸如数字信号处理器(DSP)、微控制器、专用集成电路(ASIC)或微处理器之类的处理器的任何系统。

[0142] 程序代码可以以高级过程编程语言或面向对象的编程语言实现,以与处理系统通信。如果需要,程序代码还可以以汇编语言或机器语言实现。实际上,本文描述的机制的范围不限于任何特定编程语言。在各种实施例中,语言可以是编译语言或解释语言。

[0143] 上面阐述的方法、硬件、软件、固件或代码的实施例可以经由存储在能够由处理元件执行(或能够以其他方式访问)的机器可访问、机器可读、计算机可访问或计算机可读介质上的指令或代码来实现。机器可访问/可读介质包括以机器(例如,计算机或电子系统)可读的形式提供(即,存储和/或传输)信息的任何机制。例如,机器可访问介质包括随机存取存储器(RAM),例如,静态RAM(SRAM)或动态RAM(DRAM);ROM;磁性或光学存储介质;闪速存储器设备;电存储设备;光学存储设备;声学存储设备;用于保持从暂时性(传播)信号(例如,载波、红外信号、数字信号)接收的信息的其他形式的存储设备等,这些介质与可以从其接收信息的非暂时性介质区分开。

[0144] 用于将逻辑编程为执行公开内容的实施例的指令可以存储在系统中的存储器(例如,DRAM、高速缓存、闪速存储器或其他存储装置)内。此外,指令可以经由网络或通过其他计算机可读介质来分发。因此,机器可读介质可以包括用于以机器(例如,计算机)可读的形式存储或传输信息的任何机制,但不限于软盘、光盘、压缩盘只读存储器(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、闪速存储器,或者用于经由电、光、声或其他形式的传播信号(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适合于以机器(例如,计算机)可读的形式存储或传输电子指令或信息的任何类型的有形机器可读介质。

[0145] 逻辑可以用于实现各种组件的功能中的任何功能。“逻辑”可以指代硬件、固件、软件和/或其中每个的组合，以用于执行一个或多个功能。作为示例，逻辑可以包括诸如微控制器或处理器之类的与非暂时性介质相关联的硬件，该非暂时性介质用于存储适于由微控制器或处理器执行的代码。因此，在一个实施例中，对逻辑的引用指代特别地配置为识别和/或执行要保留在非暂时性介质上的代码的硬件。此外，在另一实施例中，逻辑的使用指代包括代码的非暂时性介质，该代码特别适于由微控制器执行以执行预定的操作。并且如可以推断的，在又一实施例中，术语逻辑(在该示例中)可以指代硬件和非暂时性介质的组合。在各种实施例中，逻辑可以包括可操作以用于执行软件指令的微处理器或其他处理元件、诸如专用集成电路(ASIC)之类的分立逻辑、诸如现场可编程门阵列(FPGA)之类的编程逻辑器件、包含指令的存储器设备、逻辑器件的组合(例如，如可以在印刷电路板上找到的)或其他合适的硬件和/或软件。逻辑可以包括一个或多个门或其他电路组件，其可以由例如晶体管实现。在一些实施例中，逻辑还可以完全体现为软件。软件可以体现为记录在非暂时性计算机可读存储介质上的软件包、代码、指令、指令集和/或数据。固件可以体现为在存储器设备中被硬编码(例如，非易失性)的代码、指令或指令集和/或数据。经常，被示为分离的逻辑边界通常变化并且潜在地重叠。例如，第一逻辑和第二逻辑可以共享硬件、软件、固件或其组合，同时潜在地保留一些独立的硬件、软件或固件。

[0146] 在一个实施例中，短语“用于”或“被配置为”的使用指代对装置、硬件、逻辑或元件进行布置、放置在一起、制造、提供销售、导入和/或设计，以执行指定的任务或确定的任务。在该示例中，如果装置或其元件被设计、耦合和/或互连以执行指定的任务，则未被操作的装置或其元件仍“被配置为”执行所述指定的任务。作为纯粹说明性示例，逻辑门可以在操作期间提供0或1。但是逻辑门“被配置为”向时钟提供使能信号不包括每个潜在逻辑门可以提供1或0。相反，逻辑门是以某种方式耦合的，在操作期间1或0输出用于使能时钟。再次注意，术语“被配置为”的使用不要求操作，而是关注于装置、硬件和/或元件的隐藏状态，其中在隐藏状态下，装置、硬件和/或元件被设计用于在装置、硬件和/或元件操作时执行特定任务。

[0147] 此外，在一个实施例中，短语“能够/能够用于”和/或“可操作用于”的使用指代一些装置、逻辑、硬件和/或元件以这样的方式设计：使得能够以指定的方式使用装置、逻辑、硬件和/或元件。注意，如上面在一个实施例中对用于、能够用于或可操作用于的使用指代装置、逻辑、硬件和/或元件的隐藏状态，其中装置、逻辑、硬件和/或元件不操作但以这种方式设计为使得能够以指定方式使用装置。

[0148] 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。经常，逻辑电平、逻辑值或逻辑的值的使用也称为1和0的使用，其仅表示二进制逻辑状态。例如，1指代高逻辑电平，并且0指代低逻辑电平。在一个实施例中，诸如晶体管或闪存单元之类的存储单元能够保持单个逻辑值或多个逻辑值。然而，已经在计算机系统中使用值的其他表示。例如，十进制数十也可以表示为二进制值1010和十六进制字母A。因此，值包括在计算机系统中能够保存的信息的任何表示。

[0149] 此外，状态可以由值或值的部分表示。作为示例，第一值(例如，逻辑一)可以表示默认或初始状态，而第二值(例如，逻辑零)可以表示非默认状态。另外，在一个实施例中，术语复位和置位分别指默认值或状态和更新的值或状态。例如，默认值潜在地包括高逻辑值

(即,复位)而更新的值潜在地包括低逻辑值(即,置位)。注意,可以使用值的任何组合来表示任何数量的状态。

[0150] 系统、方法、计算机程序产品和装置可以包括以下示例中的一个或组合:

[0151] 示例1可以包括一种计算机设备,该计算机设备包括:存储器设备;高速缓存设备;以及链路层组件,其耦合到存储器设备和高速缓存设备,链路层组件用于:识别从存储器设备接收的第一数据,第一数据采用第一协议格式;识别从高速缓存设备接收的第二数据,第二数据采用第二协议格式;以及基于第一数据和第二数据来生成用于发送的流控制单元(flit),其中,flit的大小为528比特。

[0152] 示例2可以包括示例1的计算机设备,其中,生成flit包括:复用第一数据的部分和第二数据的部分以生成flit。

[0153] 示例3可以包括示例1的计算机设备,其中,flit包括用于有效载荷的512比特和用于循环冗余校验的16比特。

[0154] 示例4可以包括示例3的计算机设备,其中,512比特由第一数据的部分和第二数据的部分组成。

[0155] 示例5可以包括示例1的计算机设备,其中,flit被划分为相等大小的时隙,并且其中,flit的第一时隙包含报头信息。

[0156] 示例6可以包括示例1的计算机设备,其中,链路层组件包括发送器链路层和接收器链路层,并且其中,发送器链路层维护翻转,并且接收器链路层维护与翻转相对应的影子翻转。

[0157] 示例7可以包括示例6的计算机设备,其中,flit包括全数据flit,其中,发送器链路层发送flit,并且其中,接收器链路层基于发送器链路层对flit的发送和影子翻转来确定何时预期全数据响应flit。

[0158] 示例8可以包括示例1的计算机设备,其中,flit包括全数据flit。

[0159] 示例9可以包括示例1的计算机设备,其中,flit是第一flit,其中,第一flit包括第一数据的第一部分和第二数据的第一部分,其中,链路层组件还用于生成第二flit,第二flit包括第一数据的第二部分和第二数据的第二部分,其中,第一flit包括与第一flit和第二flit相关联的报头,其中,第二flit包括全数据flit,并且其中,第二flit在第一flit之后被发送。

[0160] 示例10可以包括示例1的计算机设备,其中,链路层组件包括flit打包器电路。

[0161] 示例11可以包括示例1的计算机设备,其中,链路层组件包括桥接器或交换机。

[0162] 示例12可以包括一种方法,该方法包括:由链路层识别以第一协议格式接收的第一数据;由链路层识别以第二协议格式接收的第二数据;由链路层复用第一数据的部分和第二数据的部分以产生复用数据;以及由链路层生成包括复用数据的流控制单元(flit)。

[0163] 示例13可以包括示例12的方法,其中,flit的大小为528比特。

[0164] 示例14可以包括示例12的方法,其中,flit包括用于有效载荷的512比特和用于循环冗余校验的16比特,并且其中,有效载荷包括复用数据。

[0165] 示例15可以包括示例14的方法,其中,有效载荷由复用数据组成。

[0166] 示例16可以包括示例12的方法,其中,flit被划分为相等大小的时隙,并且其中,flit的第一时隙包含报头信息。

[0167] 示例17可以包括示例12的方法，还包括：由链路层的发送器链路层维护翻转；以及由链路层的接收器链路层维护与翻转相对应的影子翻转。

[0168] 示例18可以包括示例17的方法，还包括：由发送器链路层发送flit，其中，flit包括全数据flit；以及由接收器链路层基于发送器链路层对flit的发送和影子翻转来确定何时预期全数据响应flit。

[0169] 示例19可以包括示例12的方法，其中，flit包括全数据flit。

[0170] 示例20可以包括示例12的方法，其中，第一数据的部分是第一数据的第一部分，其中，第二数据的部分是第二数据的第一部分，其中，复用数据是第一复用数据，其中，flit是第一flit，并且其中，该方法还包括：由链路层复用第一数据的第二部分和第二数据的第二部分以产生第二复用数据；由链路层生成包括第二复用数据的第二flit，其中，第二flit包括全数据flit，并且其中，第一flit包括与第一flit和第二flit相关联的报头信息；由链路层发送第一flit；以及由链路层在第一flit之后发送第二flit。

[0171] 示例21是一种系统，该系统包括发送器设备和接收器设备，发送器设备包括：用于存储数据的存储器，用于缓存数据的高速缓存，至少部分地以硬件实现的链路层逻辑，以及至少部分地以硬件实现的复用器逻辑；接收器设备包括：用于对从发送器接收的数据包进行解码的解码器设备。链路层逻辑用于：由链路层设备识别以第一协议格式从存储器接收的第一数据；由链路层设备识别以第二协议格式从高速缓存接收的第二数据；由链路层设备复用第一数据的部分和第二数据的部分以产生复用数据；以及由链路层设备生成包括复用数据的流控制单元(flit)。

[0172] 示例22可以包括示例21的主题，其中，flit的大小为528比特。

[0173] 示例23可以包括示例21或22的主题，其中，flit包括用于有效载荷的512比特和用于循环冗余校验的16比特，并且其中，有效载荷包括复用数据。

[0174] 示例24可以包括示例23的主题，其中，有效载荷由复用数据组成。

[0175] 示例25可以包括示例21的主题，其中，flit被划分为相等大小的时隙，并且其中，flit的第一时隙包含报头信息。

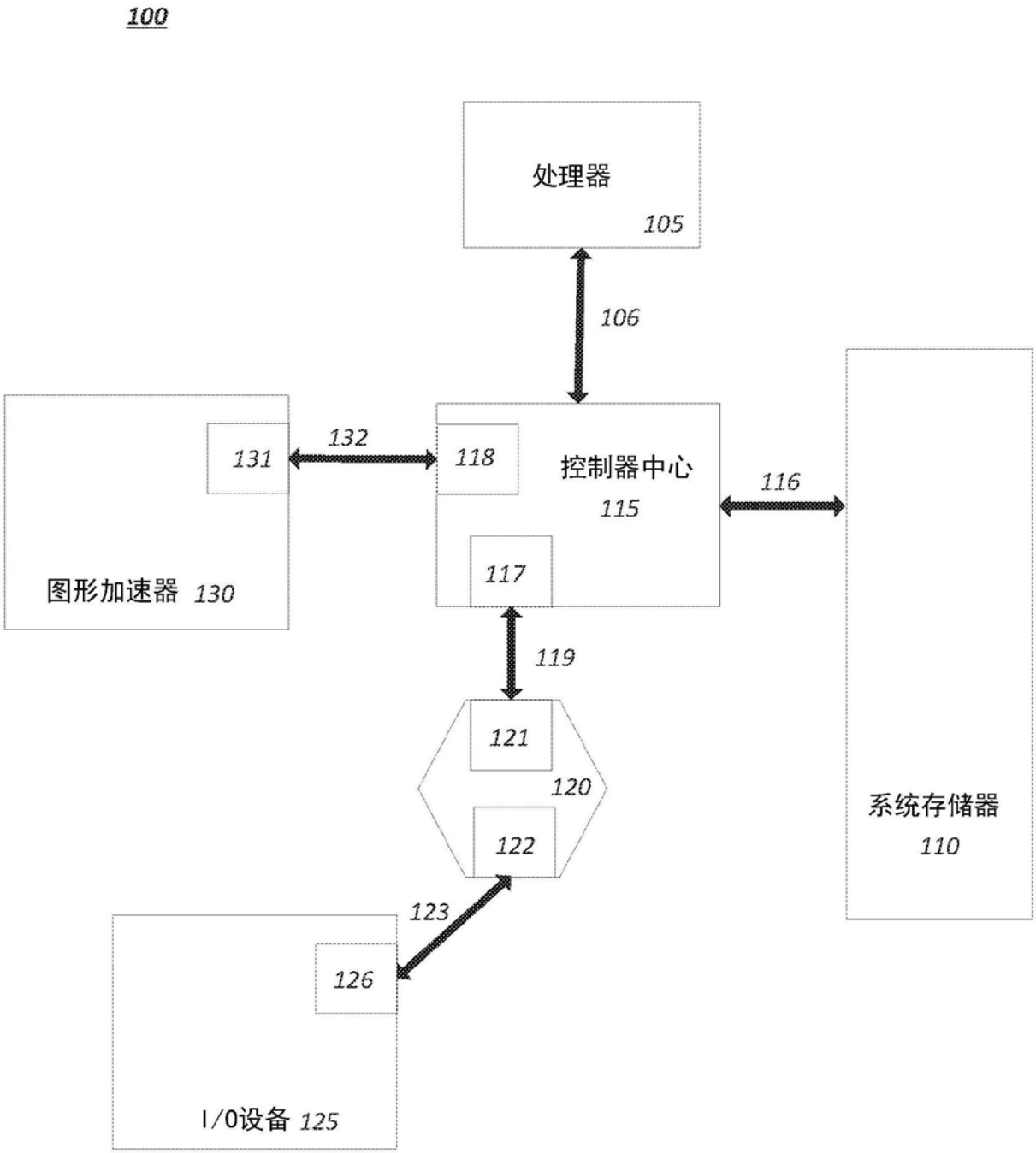

100
处理器
105
106
131
132
118
控制器中心
115
116
图形加速器 130
117
119
121
120
122
系统存储器
110
123
126
I/O设备 125

图1

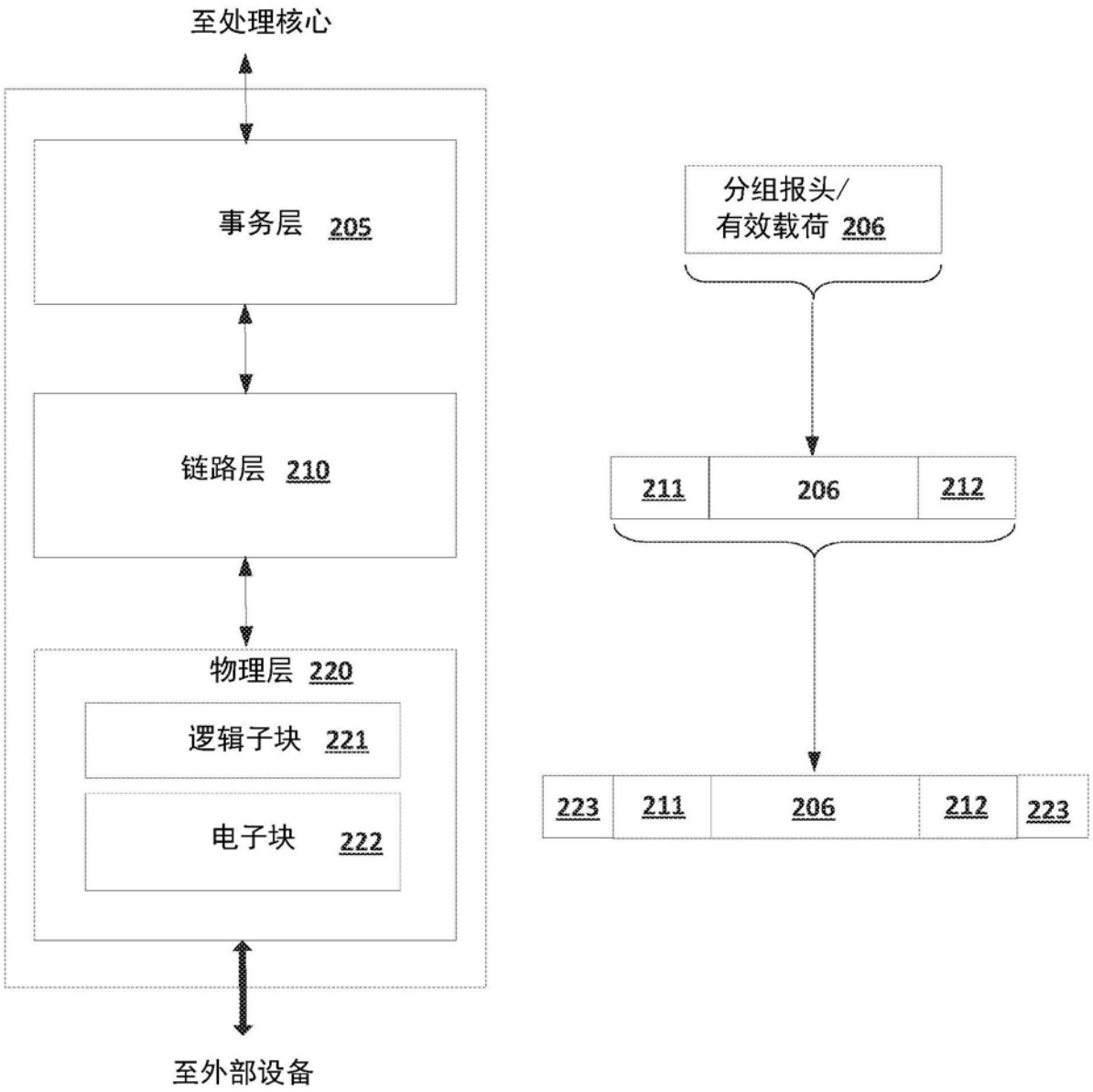

分层协议栈 200
至处理核心
事务层 205
链路层 210
物理层 220
逻辑子块 221
电子块 222
至外部设备
分组报头/
有效载荷 206
211
206
212
223
211
206
212
223

图2


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

图3

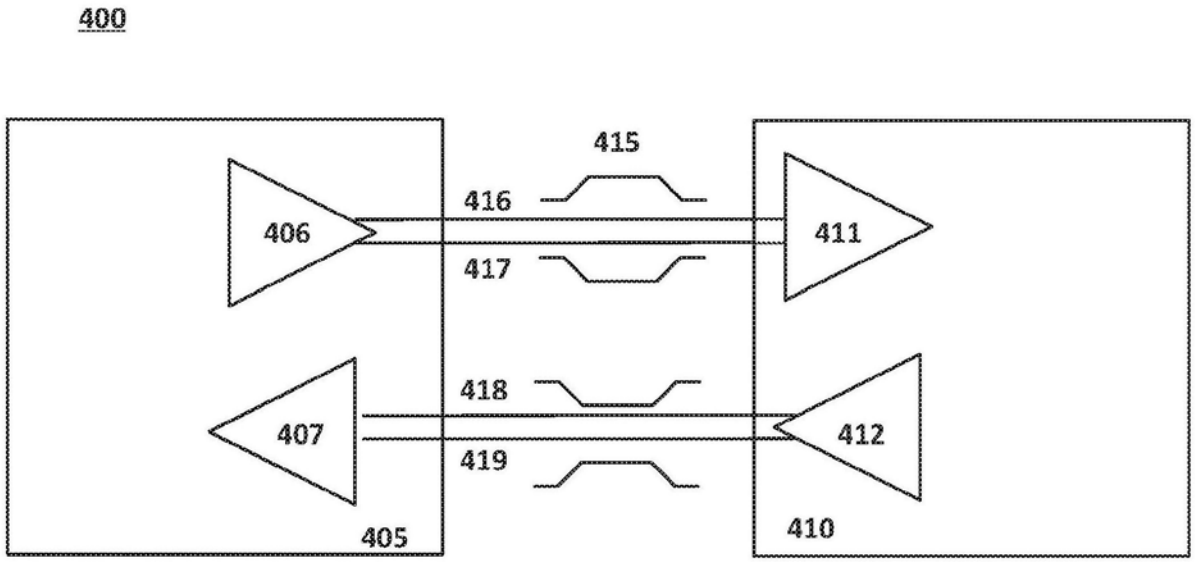

400
415
416
417
406
411
418
419
407
412
405
410

图4

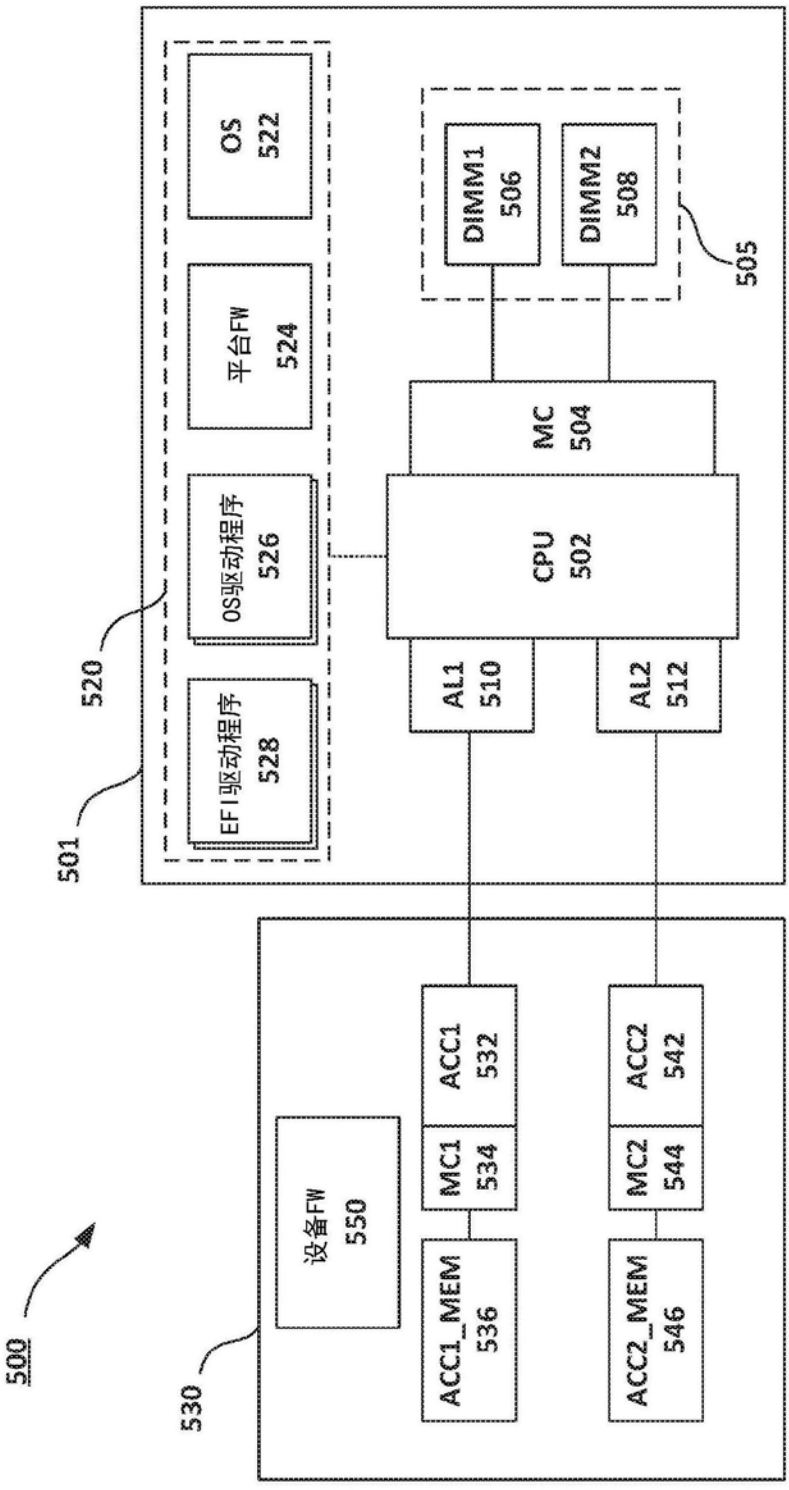

500
501
520
530
505
OS
522
平台FW
524
OS驱动程序
526
EFI驱动程序
528
DIMM1
506
DIMM2
508
MC
504
CPU
502
AL1
510
AL2
512
设备FW
550
ACC1
532
MC1
534
ACC1_MEM
536
ACC2
542
MC2
544
ACC2_MEM
546

图5

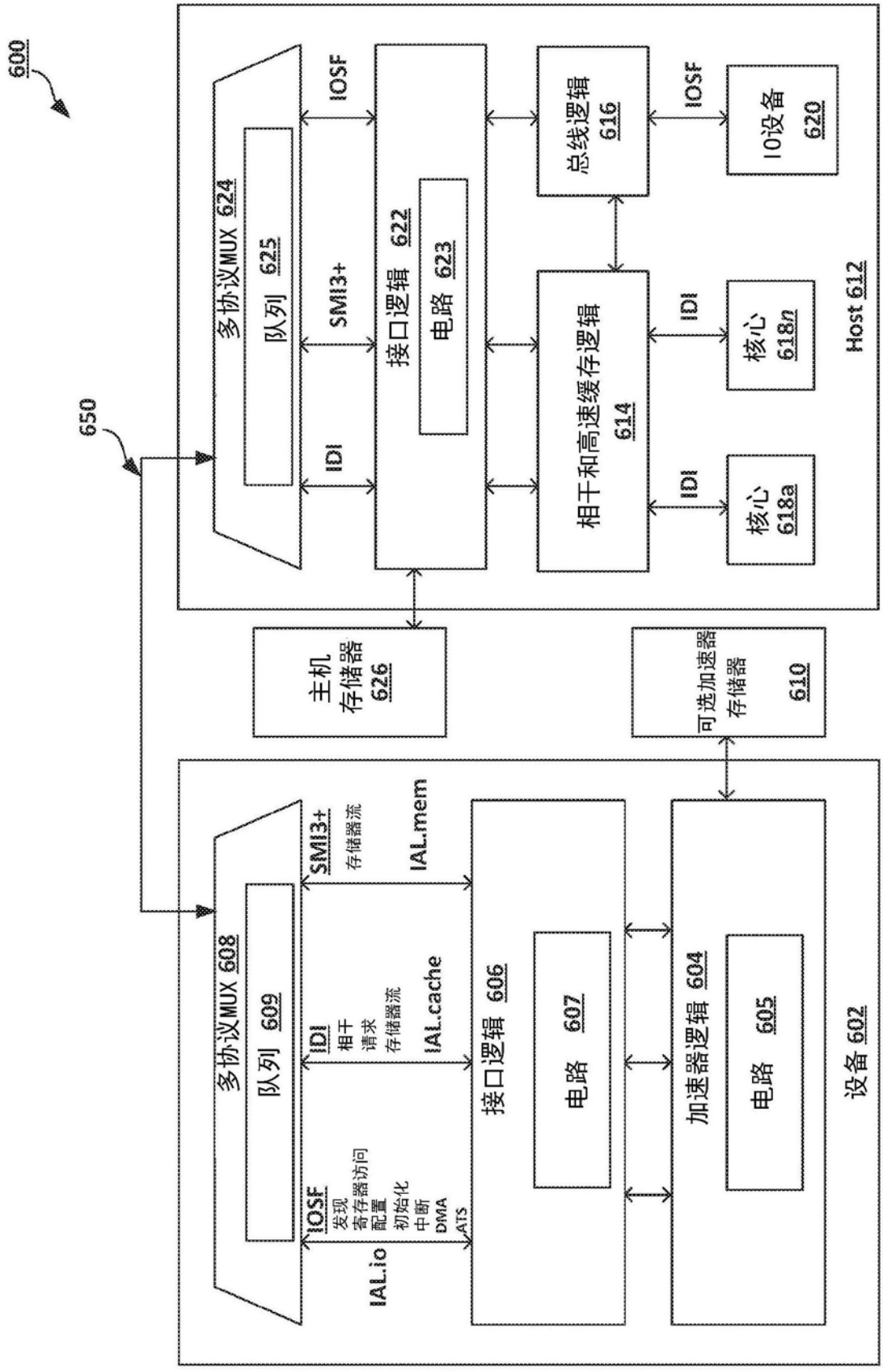

600
650
多协议MUX 624
队列 625
IOSF
SMI3+
IDI
接口逻辑 622
电路 623
总线逻辑 616
IOSF
IO设备 620
相干和高速缓存逻辑 614
IDI
核心 618n
IDI
核心 618a
Host 612
主机存储器 626
可选加速器存储器 610
多协议MUX 608
队列 609
SMI3+
存储器流
IAL.mem
IDI
相干请求
存储器流
IAL.cache
IOSF
发现
寄存器访问
配置
初始化
中断
DMA
ATS
IAL.io
接口逻辑 606
电路 607
加速器逻辑 604
电路 605
设备 602

图6

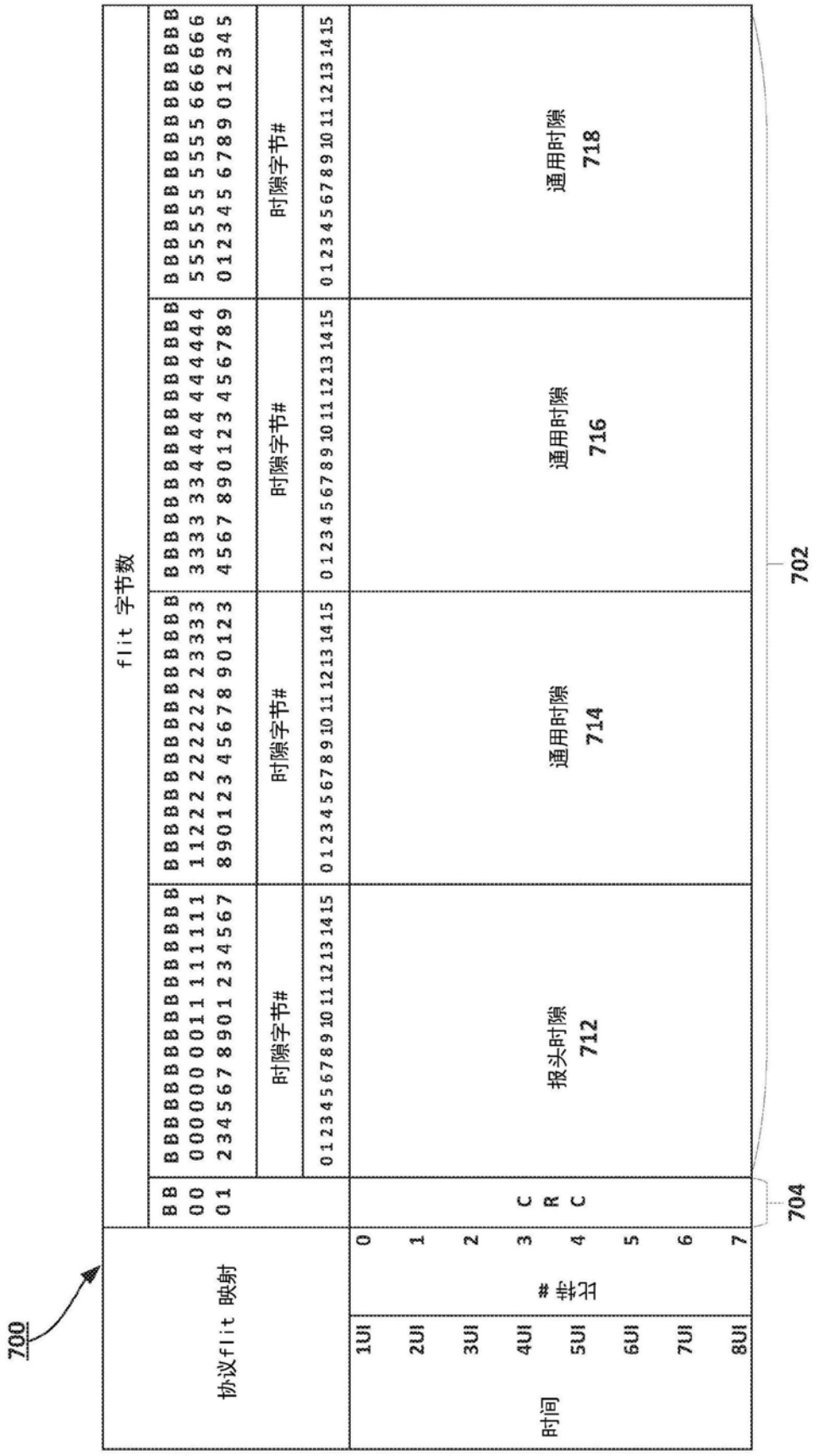

700
协议flit 映射
flit 字节数
BB 00 01
BBBBBBBBBBBBBBBB 0000000011111111 2345678901234567
BBBBBBBBBBBBBBBB 1122222222233333 8901234567890123
BBBBBBBBBBBBBBBB 3333334444444444 4567890123456789
BBBBBBBBBBBBBBBB 5555555555666666 0123456789012345
时隙字节#
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
报头时隙 712
通用时隙 714
通用时隙 716
通用时隙 718
时间
1UI 2UI 3UI 4UI 5UI 6UI 7UI 8UI
比特#
0 1 2 3 4 5 6 7
CRC
704
702

图7A

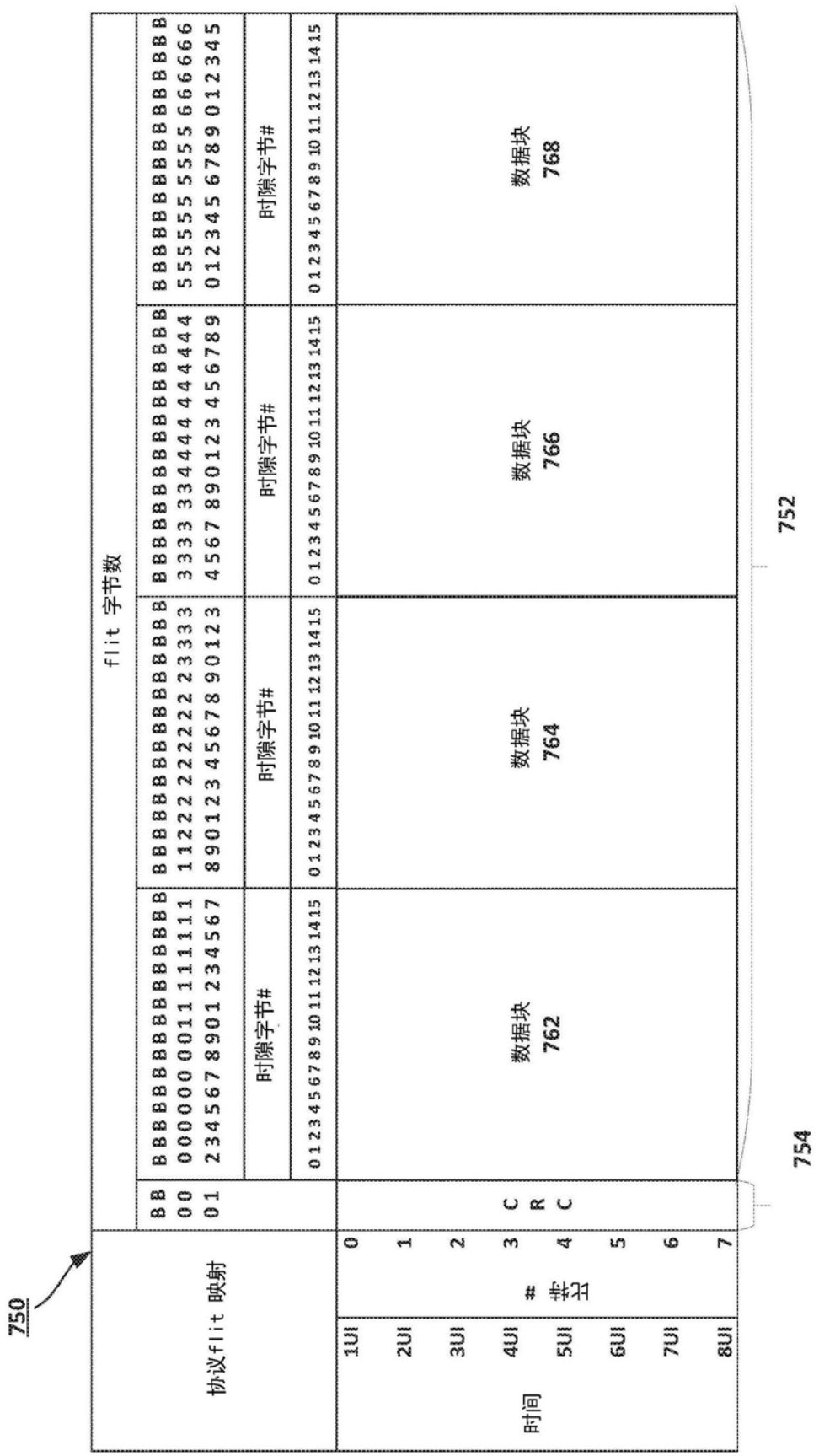

750
协议flit 映射
flit 字节数
BB 00 01
BBBBBBBBBBBBBBBB 0000000011111111 2345678901234567
BBBBBBBBBBBBBBBB 1122222222223333 8901234567890123
BBBBBBBBBBBBBBBB 3333334444444444 4567890123456789
BBBBBBBBBBBBBBBB 5555555555666666 0123456789012345
时隙字节#
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
数据块 762
数据块 764
数据块 766
数据块 768
CRC
时间
比特 #
0 1 2 3 4 5 6 7
1UI 2UI 3UI 4UI 5UI 6UI 7UI 8UI
752
754

图7B

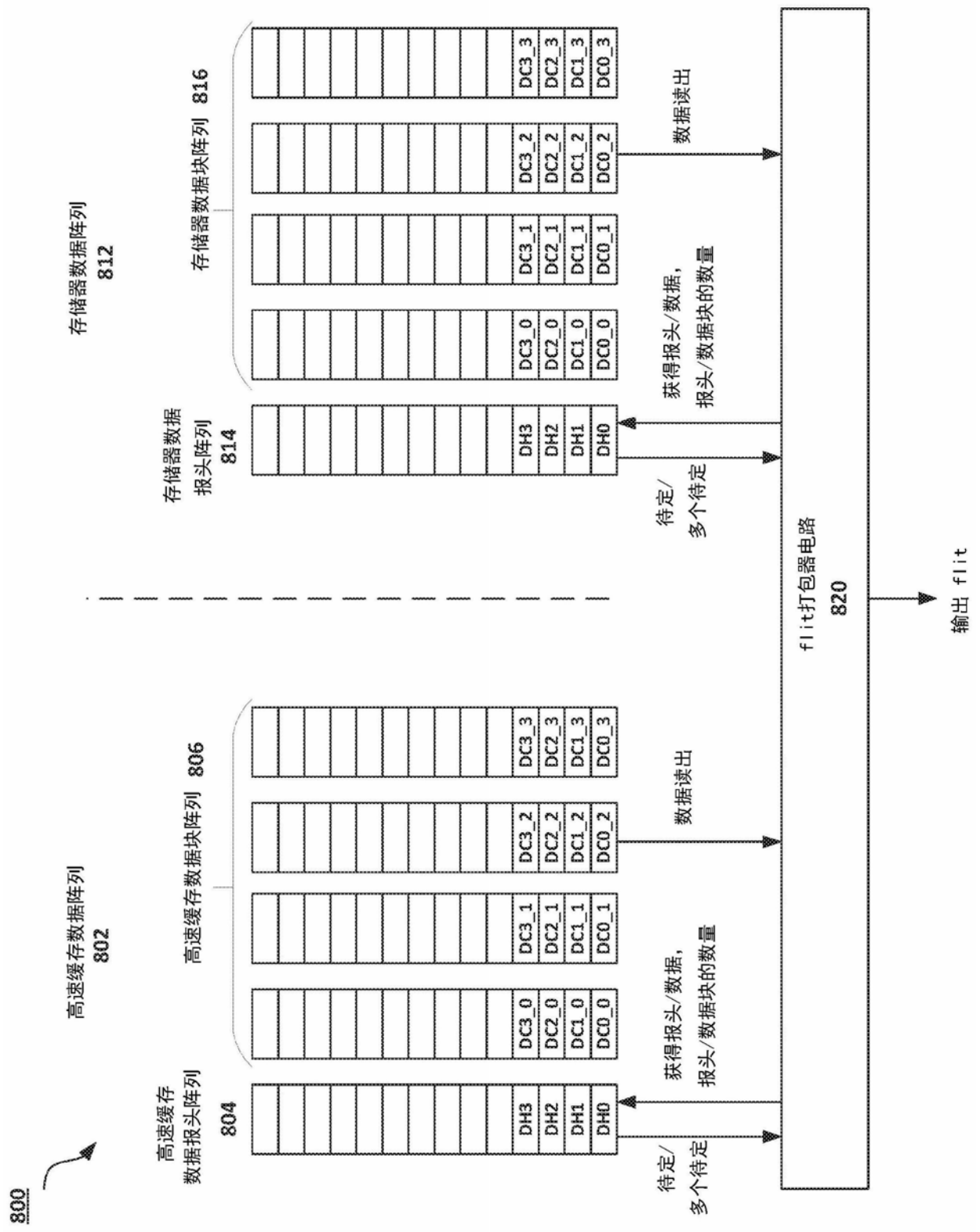
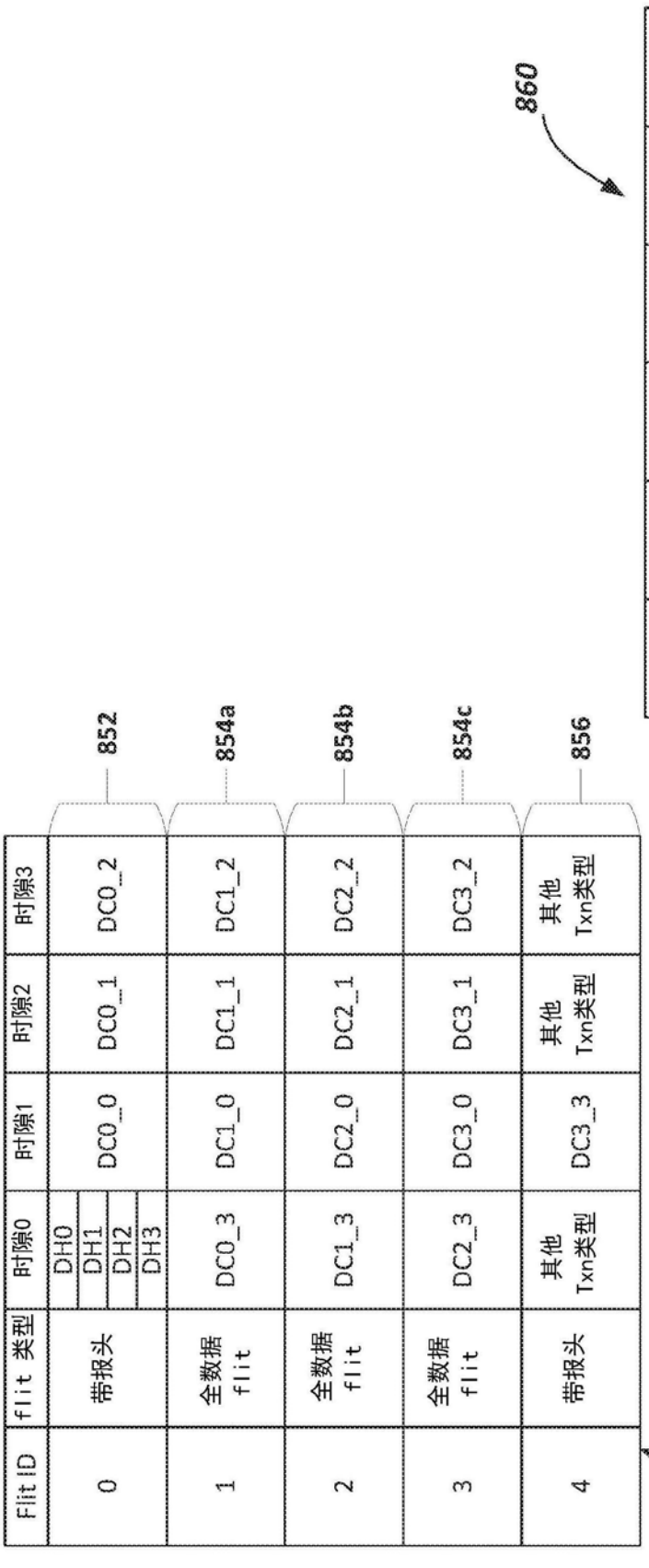

800
高速缓存数据阵列
802
高速缓存
数据报头阵列
804
高速缓存数据块阵列 806
存储器数据阵列
812
存储器数据
报头阵列
814
存储器数据块阵列 816
DH3
DH2
DH1
DH0
DC3_0
DC2_0
DC1_0
DC0_0
DC3_1
DC2_1
DC1_1
DC0_1
DC3_2
DC2_2
DC1_2
DC0_2
DC3_3
DC2_3
DC1_3
DC0_3
待定/
多个待定
获得报头/数据，
报头/数据块的数量
数据读出
flit打包器电路
820
输出 flit

图8A



Flit ID	flit 类型	时隙0	时隙1	时隙2	时隙3	
0	带报头	DH0 DH1 DH2 DH3	DC0_0	DC0_1	DC0_2	852
1	全数据 flit	DC0_3	DC1_0	DC1_1	DC1_2	854a
2	全数据 flit	DC1_3	DC2_0	DC2_1	DC2_2	854b
3	全数据 flit	DC2_3	DC3_0	DC3_1	DC3_2	854c
4	带报头	其他 Txn类型	DC3_3	其他 Txn类型	其他 Txn类型	856

850

图8B

860

Flit ID	flit 类型	时隙0	时隙1	时隙2	时隙3	
0	带报头	DH0 DH1	DC0_0	DC0_1	DC0_2	862
1	全数据 flit	DC0_3	DC1_0	DC1_1	DC1_2	864
2	带报头	其他 Txn类型	DC1_3	其他 Txn类型	其他 Txn类型	866

图8C

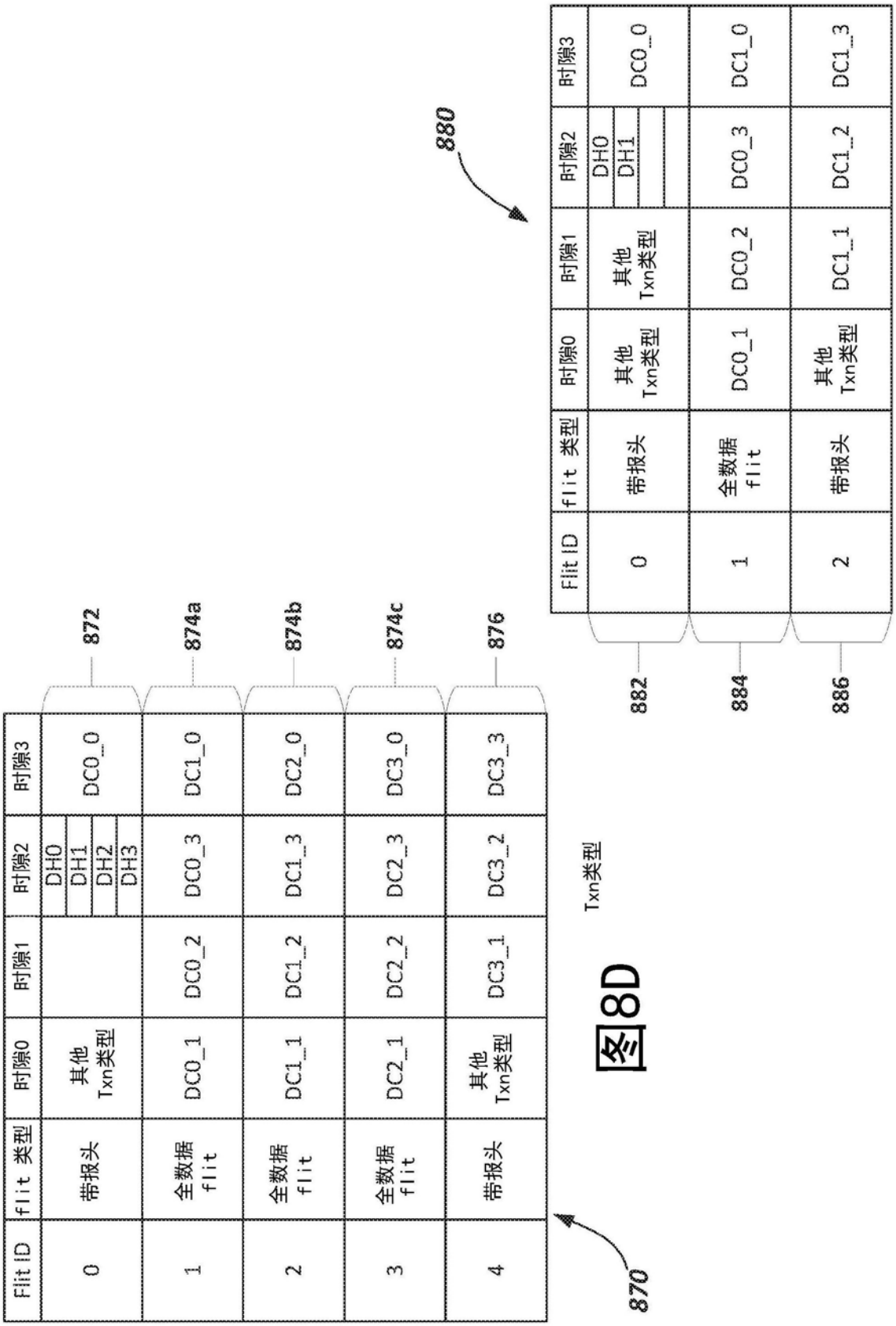

Flit ID
flit 类型
时隙0
时隙1
时隙2
时隙3
0
带报头
其他 Txn类型
DH0
DH1
DH2
DH3
DC0_0
1
全数据 flit
DC0_1
DC0_2
DC0_3
DC1_0
2
全数据 flit
DC1_1
DC1_2
DC1_3
DC2_0
3
全数据 flit
DC2_1
DC2_2
DC2_3
DC3_0
4
带报头
其他 Txn类型
DC3_1
DC3_2
DC3_3
872
874a
874b
874c
876
870
Txn类型
880
882
884
886
DH0
DH1
DC0_0
DC1_0
DC0_3
DC1_3
DC2_0
DC2_3
DC3_0
DC3_3

图8D

图8E

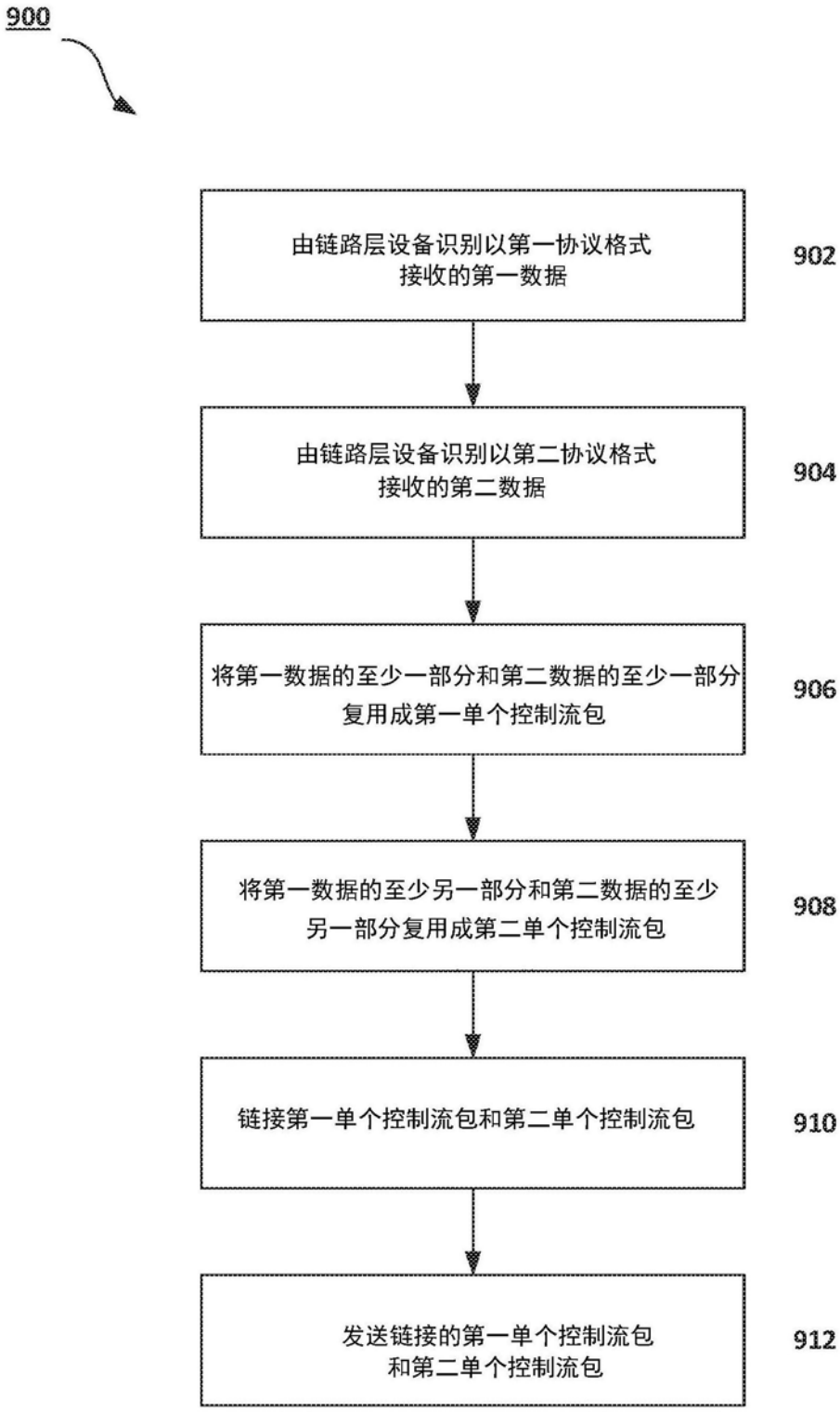

900
由链路层设备识别以第一协议格式接收的第一数据
902
由链路层设备识别以第二协议格式接收的第二数据
904
将第一数据的至少一部分和第二数据的至少一部分复用成第一单个控制流包
906
将第一数据的至少另一部分和第二数据的至少另一部分复用成第二单个控制流包
908
链接第一单个控制流包和第二单个控制流包
910
发送链接的第一单个控制流包和第二单个控制流包
912

图9A

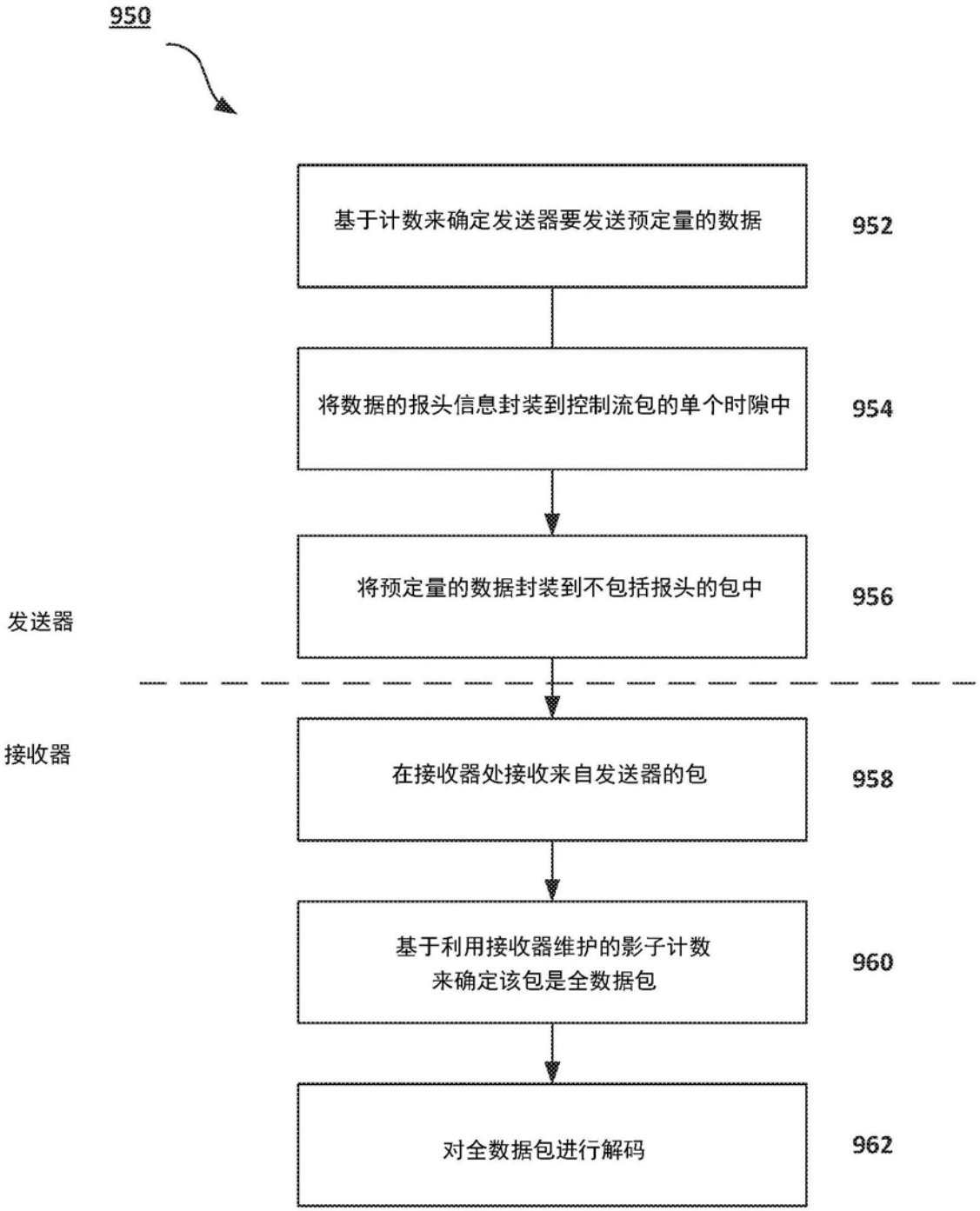

950
基于计数来确定发送器要发送预定量的数据
952
将数据的报头信息封装到控制流包的单个时隙中
954
将预定量的数据封装到不包括报头的包中
956
发送器
接收器
在接收器处接收来自发送器的包
958
基于利用接收器维护的影子计数
来确定该包是全数据包
960
对全数据包进行解码
962

图9B

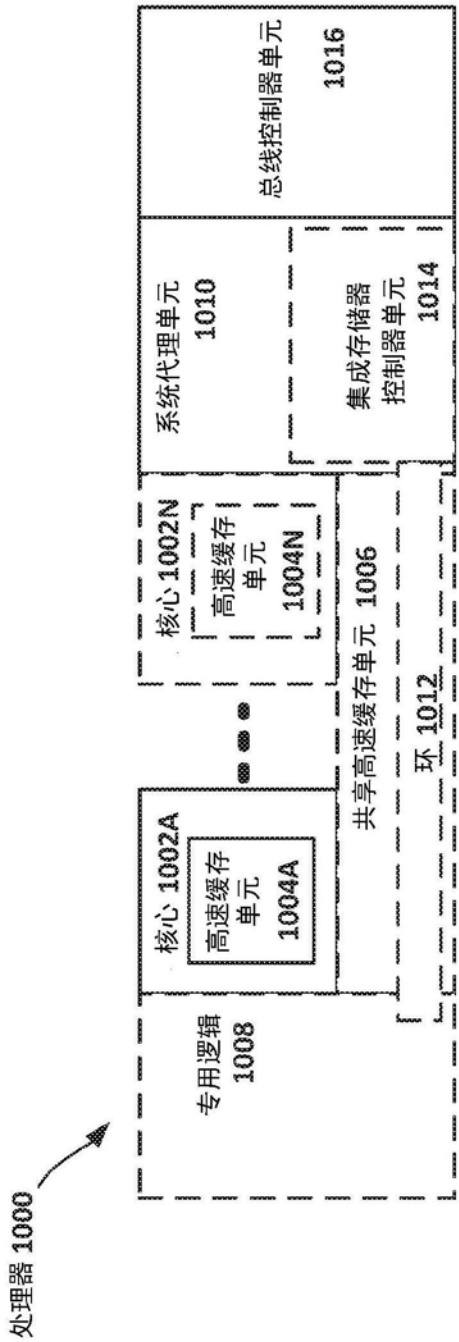

处理器 1000
专用逻辑 1008
核心 1002A
高速缓存单元 1004A
核心 1002N
高速缓存单元 1004N
共享高速缓存单元 1006
环 1012
系统代理单元 1010
集成存储器控制器单元 1014
总线控制器单元 1016

图10A

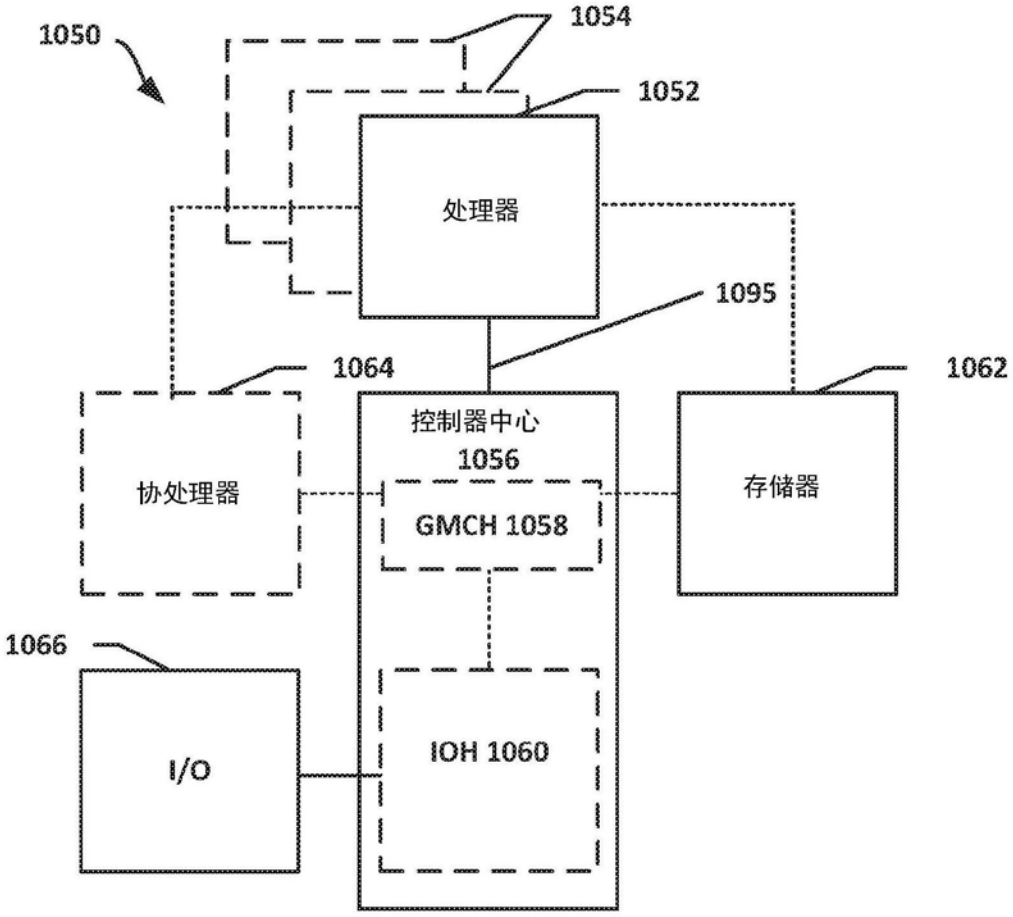

1050
1054
1052
处理器
1095
1064
1062
控制器中心
1056
协处理器
GMCH 1058
存储器
1066
I/O
IOH 1060

图10B

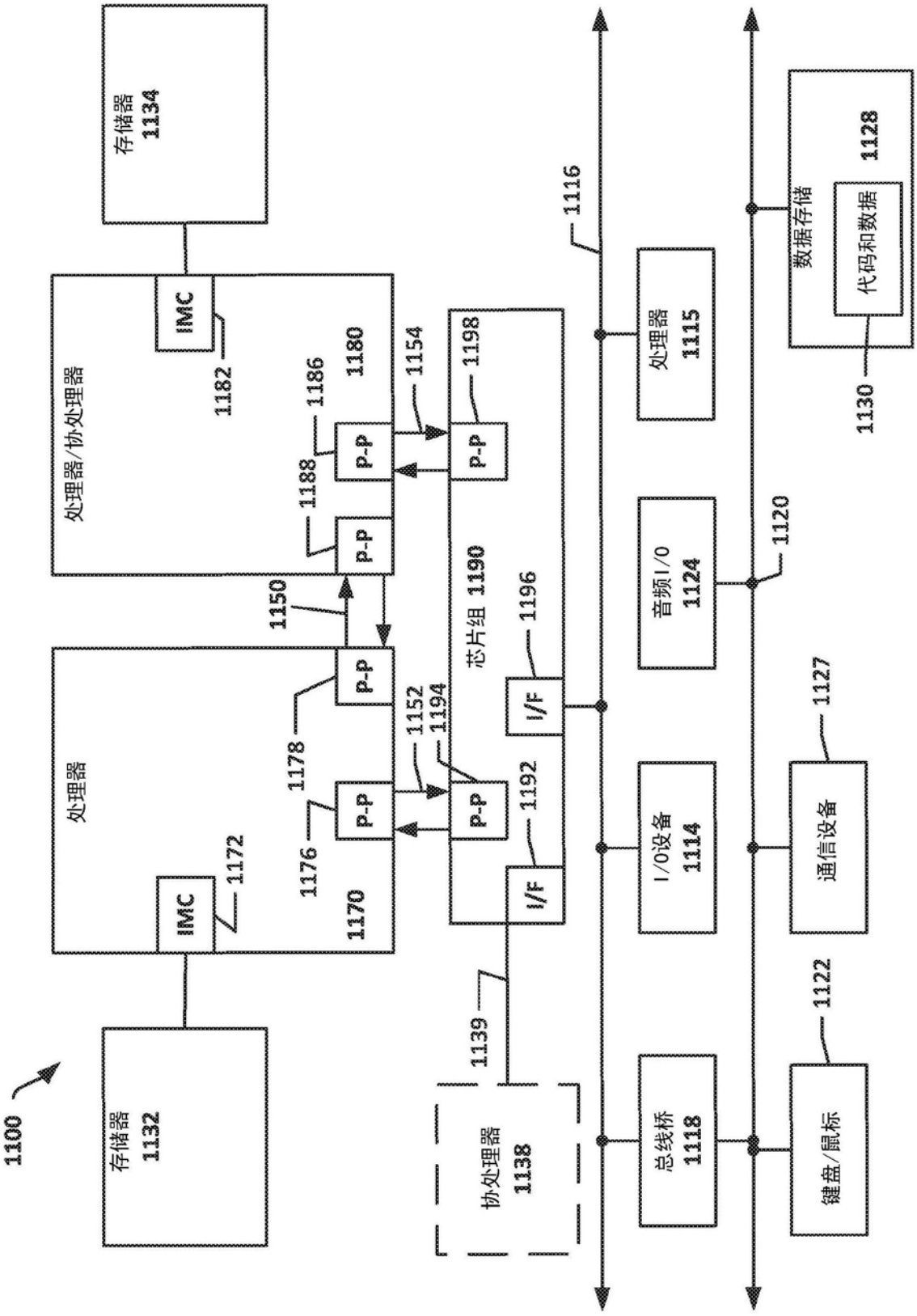

1100
存储器 1132
处理器 1170
IMC 1172
处理器/协处理器 1180
IMC 1182
存储器 1134
1176
1178
1186
1188
1150
1152
1154
1194
1198
芯片组 1190
1196
1192
1139
协处理器 1138
I/F
P-P
1116
处理器 1115
音频I/O 1124
I/O设备 1114
总线桥 1118
1120
1127
通信设备
1122
键盘/鼠标
数据存储 1128
代码和数据
1130

图11

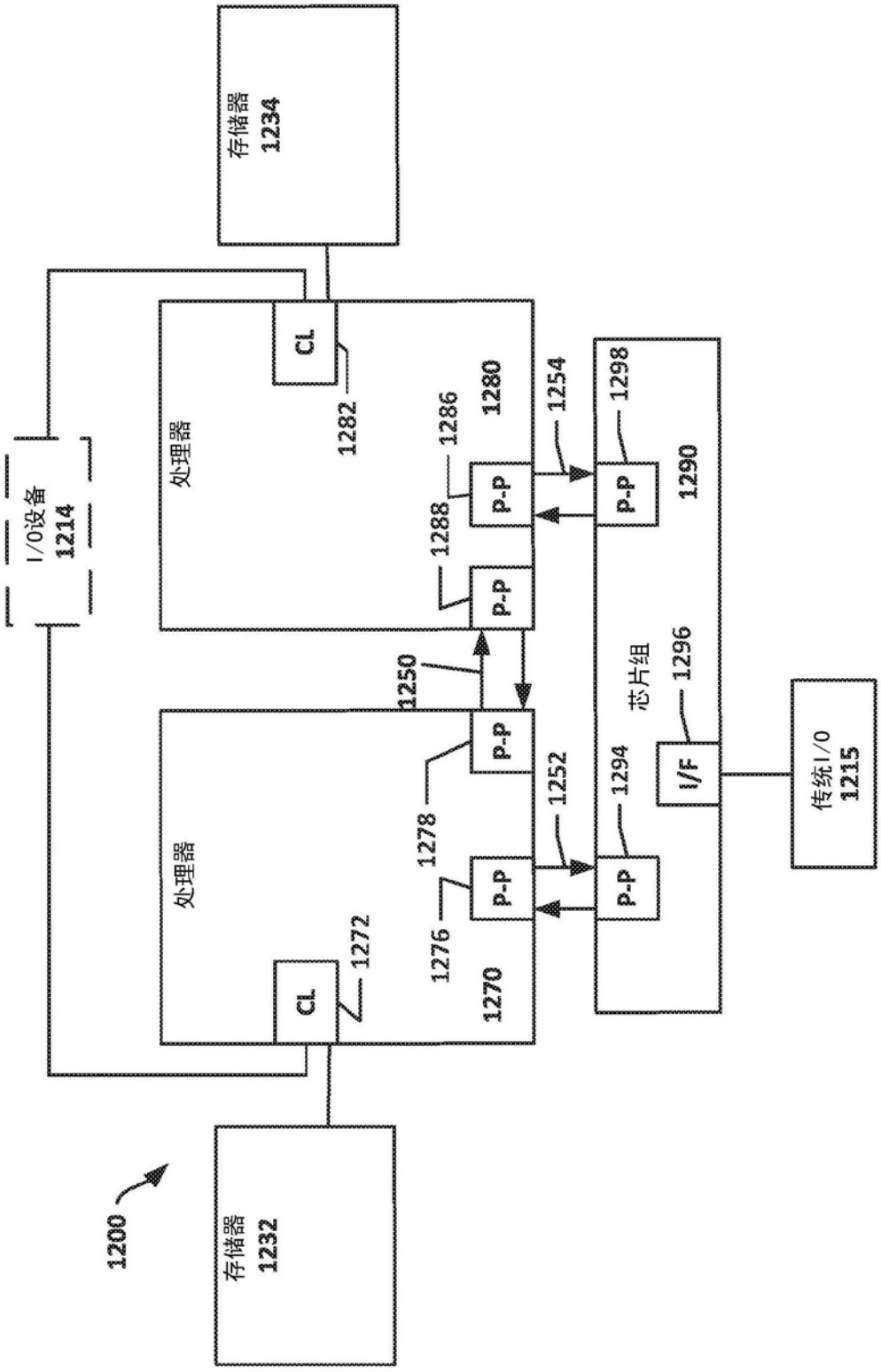

存储器
1234
CL
1282
处理器
1280
1286
P-P
1288
P-P
1254
1298
P-P
1290
芯片组
1296
I/F
传统I/O
1215
1294
P-P
1252
1278
P-P
1276
处理器
1270
CL
1272
1250
I/O设备
1214
1200
存储器
1232

图12

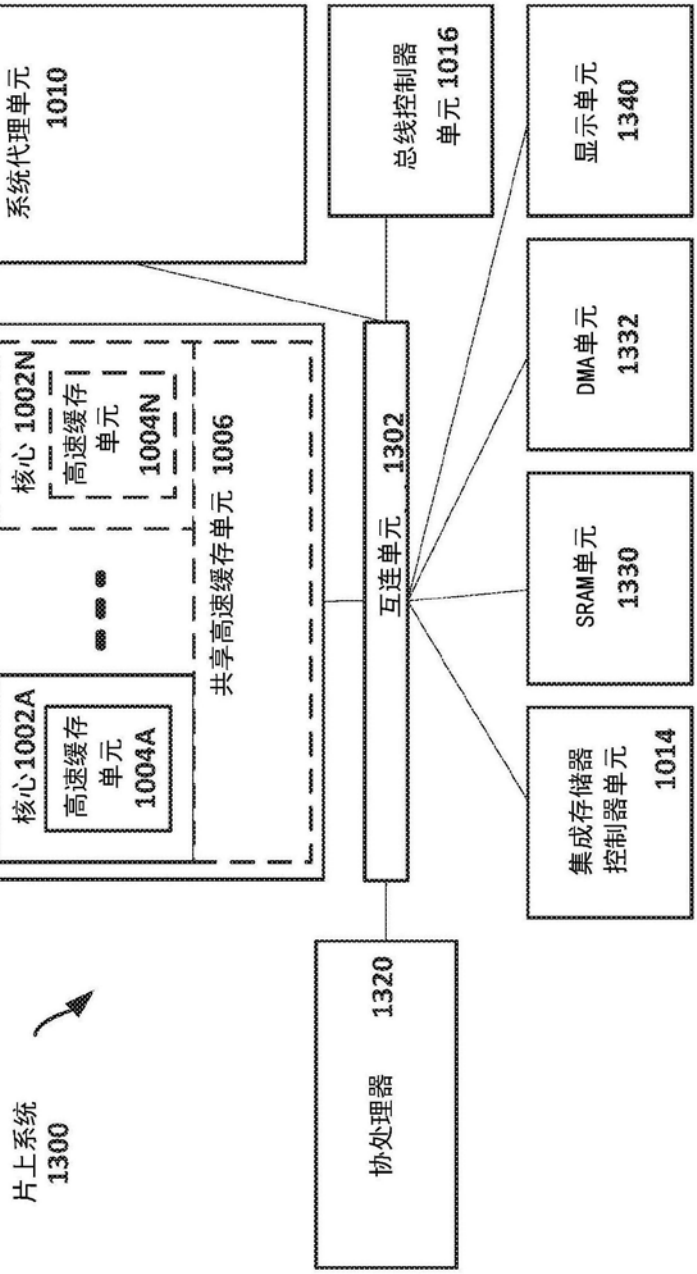

片上系统
1300
应用处理器 1310
核心1002A
高速缓存单元 1004A
核心 1002N
高速缓存单元 1004N
共享高速缓存单元 1006
协处理器 1320
互连单元 1302
系统代理单元 1010
总线控制器单元 1016
集成存储器控制器单元 1014
SRAM单元 1330
DMA单元 1332
显示单元 1340

图13

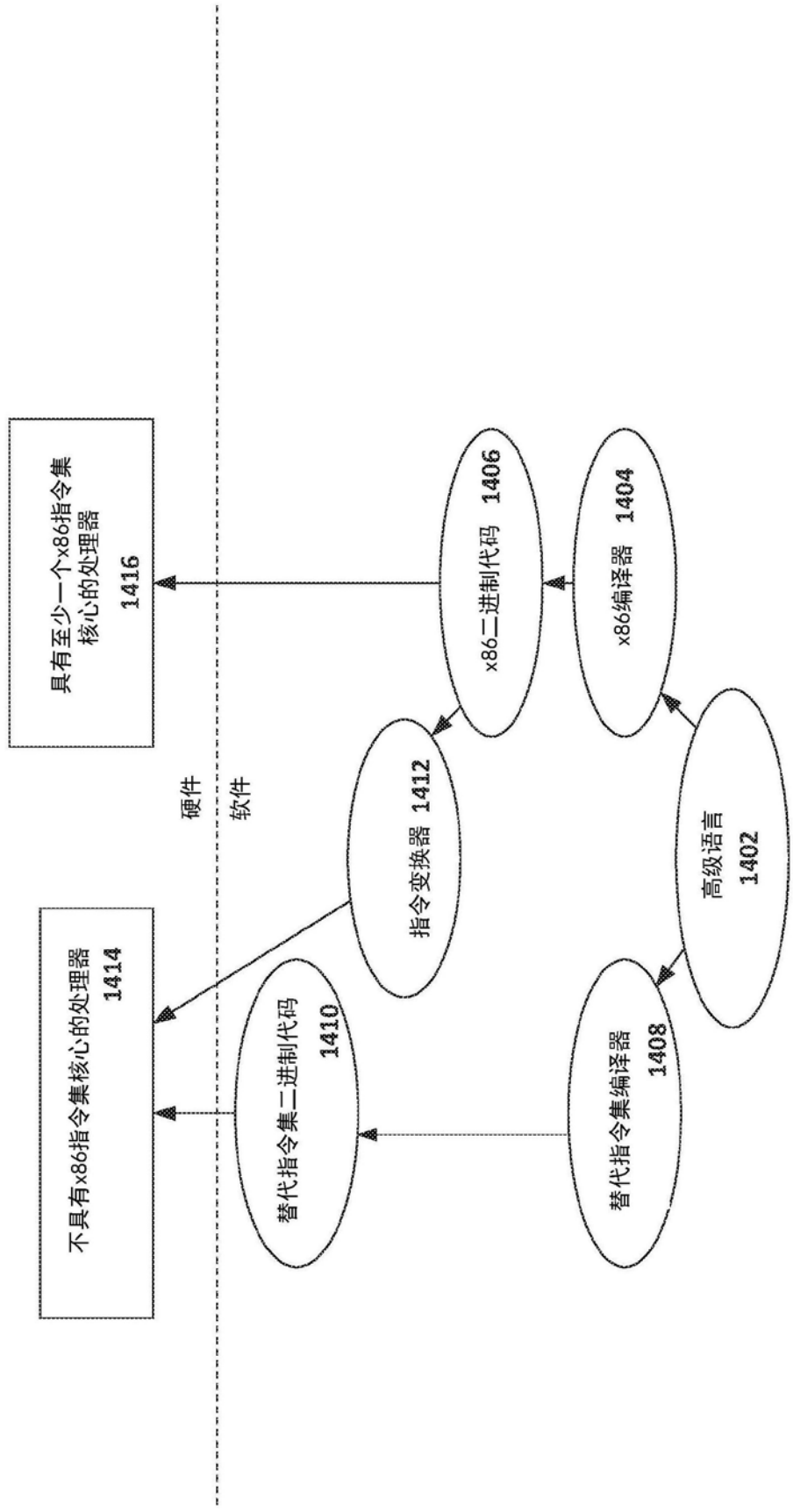

具有至少一个x86指令集核心的处理器 1416
不具有x86指令集核心的处理器 1414
硬件
软件
x86二进制代码 1406
x86编译器 1404
指令变换器 1412
高级语言 1402
替代指令集二进制代码 1410
替代指令集编译器 1408

图14