(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 113868173 A
(43)申请公布日 2021.12.31

(21)申请号 202111150075.4

(22)申请日 2017.02.02

(30)优先权数据
62/303,487 2016.03.04 US
15/281,318 2016.09.30 US

(62)分案原申请数据
201780009811.X 2017.02.02

(71)申请人 英特尔公司
地址 美国加利福尼亚

(72)发明人 D·J·哈里曼 R·罗齐奇
M·达恩 P·塞蒂 R·E·高夫
S·K·拉宾达拉纳特

(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

(51)Int.Cl.
G06F 13/40(2006.01)
G06F 13/42(2006.01)

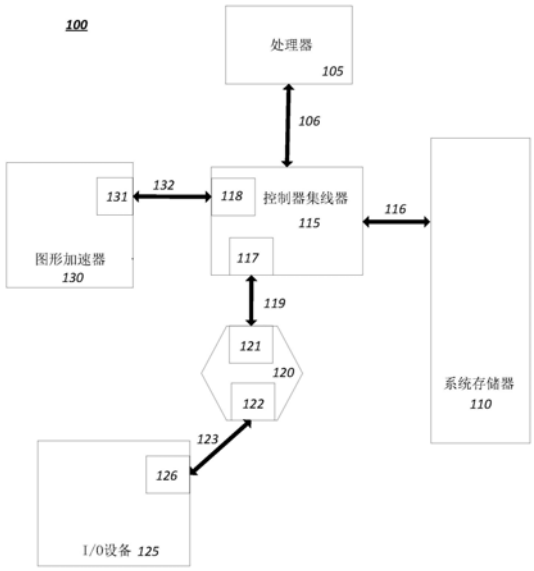
权利要求书2页 说明书38页 附图12页

(54)发明名称

扁平化端口桥

(57)摘要

提供扁平化端口桥(FPB)以支持根据第一寻址方案和第二替代寻址方案的寻址。FPB包括主要侧和辅助侧,主要侧连接到根据第一寻址方案寻址的第一组设备,并且辅助侧连接到根据第二寻址方案寻址的第二组设备。对于第一组设备中的每个设备,第一寻址方案使用总线/设备/功能(BDF)地址空间中的唯一的总线编号,而对于第二组设备中的每个设备,第二总线寻址方案使用唯一的总线设备编号。



CN 113868173 A

1.一种装置,包括:

具有主要侧和辅助侧的扁平化端口桥(FPB),所述FPB支持类型1桥接功能以解码事务层分组(TLP),其中,所述类型1桥接功能包括非FPB分组解码/路由机制和FPB分组解码/路由机制,其中,所述FPB分组解码/路由机制允许在非连续范围内分配路由标识符(ID)和存储器空间资源;以及

用于基于对所述FPB分组解码/路由机制的输出和所述非FPB分组解码/路由机制的输出执行逻辑或运算来生成解码结果,以确定是否将所述TLP与所述FPB的主要侧或所述FPB的辅助侧相关联的逻辑。

2.如权利要求1所述的装置,其中,所述FPB分组解码/路由机制包括路由ID(RID)辅助开始、矢量开始以及粒度寄存器以执行对所述TLP的基于路由ID的解码。

3.如权利要求1所述的装置,其中,所述FPB分组解码/路由机制包括存储器低向量开始、粒度、存储器低向量、存储器高向量开始以及存储器高向量寄存器以执行对所述TLP的基于存储器的解码。

4.如权利要求1所述的装置,还包括根复合体,其中,所述根复合体的至少一个端口包括所述FPB。

5.如权利要求1所述的装置,还包括交换机,其中,所述交换机的至少一个端口包括所述FPB。

6.如权利要求1所述的装置,其中,所述非FPB分组解码/路由机制包括辅助/从属总线编号寄存器以执行对所述TLP的基于路由ID的解码。

7.如权利要求1所述的装置,其中,所述非FPB分组解码/路由机制包括存储器基址/限制寄存器和可预取基址/限制寄存器以执行对所述TLP的基于存储器的解码。

8.一种方法,包括:

识别计算系统中的多个设备,其中,所述多个设备中的每个设备通过多个逻辑总线中的至少相应的一个逻辑总线连接在所述计算系统中;以及

向所述多个设备中的每一个分配相应的地址,其中,将所述地址分配给设备包括:

确定是根据第一寻址方案还是根据第二总线寻址方案来分配所述地址,其中,所述第一寻址方案将总线/设备/功能(BDF)地址空间中的唯一总线编号分配给在所述第一寻址方案中寻址的每个设备,并且所述第二总线寻址方案分配所述BDF地址空间中的唯一的总线设备编号,其中,每个总线设备编号包括八位总线编号和五位设备编号。

9.一种用于处理事务的系统,包括:

耦合到交换机的多个设备;

所述交换机,其包括输入/输出模块,所述输入/输出模块实现包含事务层的分层协议栈,其中,所述事务层组装包括事务描述符的分组报头/有效载荷,并且所述事务描述符包括:

全局标识信息,其包括源标识信息和本地事务标识信息,所述源标识信息用于唯一地标识所述多个设备中的请求所述事务的设备,而所述本地事务标识信息用于提供所述系统中的事务的全局标识;

属性信息,其用于提供允许修改所述事务的默认处理的附加信息;以及

信道标识符信息。

10.如权利要求9所述的系统,其中,所述属性信息包括优先级信息,其用于由请求所述事务的所述设备修改以向所述事务分配优先级。

11.如权利要求9或10所述的系统,其中,所述属性信息包括排序属性信息,其用于指示修改默认排序规则的排序类型。

扁平化端口桥

[0001] 本申请为分案申请，其原申请是于2018年08月03日(国际申请日为2017年02月02日)向中国专利局提交的专利申请，申请号为201780009811，发明名称为“扁平化端口桥”。

[0002] 本申请要求于2016年3月4日提交的美国临时专利申请序号NO.62/303,487的权益，该临时专利申请通过引用方式整体并入本文。

技术领域

[0003] 本公开涉及计算系统，并且特别地(但非排他地)涉及地址空间映射。

背景技术

[0004] 外围组件互连(PCI)配置空间由采用PCI、PCI-X和快速PCI(PCIe)的系统用于执行基于PCI的设备的配置任务。基于PCI的设备具有用于设备配置寄存器的地址空间，称为配置空间，并且快速PCI针对设备引入了扩展的配置空间。配置空间寄存器通常由主处理器映射到存储器映射的输入/输出位置。设备驱动器、操作系统和诊断软件访问配置空间，并可以向配置空间寄存器读取和写入信息。

[0005] PCI本地总线相对于其他I/O架构的改进之一是其配置机制。除了正常的存储器映射和I/O端口空间外，总线上的每个设备功能都有256字节长的配置空间，通过知道设备的8位PCI总线、5位设备和3位功能编号可以寻址(通常称为BDF或B/D/F，缩写为总线/设备/功能)。这允许多达256个总线，每个总线最多32个设备，每个总线支持8个功能。单个PCI扩展卡可以作为设备进行响应，并且至少可以实现编号为零的功能。配置空间的前64个字节是标准化的；其余部分是可用规范定义的扩展和/或供应商定义的目的。

[0006] 为了允许标准化配置空间的更多部分而不与现有用途冲突，可以存在在外围组件接口配置空间的高192字节内定义的能力列表。每个功能都有一个描述它的能力的字节，以及一个字节用于指向下一个能力。附加字节数取决于能力ID。如果正在使用能力，则会设置状态寄存器中的位，并提供指向链路的能力列表中第一个的指针。已经向PCIe的版本提供了类似的特征，包括扩展的配置空间，将配置空间的总大小扩展到4096字节，以及诸如PCIe扩展的能力结构。

附图说明

[0007] 图1示出了包括互连架构的计算系统的实施例。

[0008] 图2示出了包括分层栈的互连架构的实施例。

[0009] 图3示出了要在互连架构内生成或接收的请求或分组的实施例。

[0010] 图4示出了用于互连架构的发送器和接收器对的实施例。

[0011] 图5示出了系统总线的表示。

[0012] 图6示出了系统中的总线标识符的枚举的表示。

[0013] 图7A示出了采用扁平化端口桥(FPB)的实例的系统的表示。

[0014] 图7B示出了FPB的示例实现方式。

[0015] 图8示出了示例FPB的详细表示。

[0016] 图9示出了BDF空间和支持的粒度中的示例地址。

[0017] 图10示出了FPB MEM Low机制应用于4GB以下的存储器地址空间中的地址布局以及粒度对这些地址的影响。

[0018] 图11示出了包括多核处理器的计算系统的框图的实施例。

[0019] 图12示出了计算系统的框图的另一实施例。

具体实施方式

[0020] 在以下描述中,阐述了许多具体细节,诸如特定类型的处理器和系统配置的示例,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、特定的测量值/高度、特定的处理器流水线阶段和操作等,以便提供对本发明的透彻理解。然而,对于本领域技术人员显而易见的是,不需要采用这些具体细节来实施本发明。在其他实例中,为了避免不必要地模糊本发明,没有详细描述众所周知的组件或方法,例如特定和替代处理器架构、用于所描述的算法的特定逻辑电路/代码、特定固件代码、特定互连操作、特定逻辑配置、特定制造技术和材料、特定编译器实现、代码形式的算法的特定表达、特定的断电和门控技术/逻辑以及计算机系统的其他特定操作细节。

[0021] 尽管可以参考特定集成电路中(例如,在计算平台或微处理器中)的节能和能量效率来描述以下实施例,但是其他实施例也适用于其他类型的集成电路和逻辑设备。本文描述的实施例的类似技术和教导可以应用于其他类型的电路或半导体设备,其也可以受益于更好的能量效率和节能。例如,所公开的实施例不限于台式计算机系统或Ultrabooks™。并且还可以用于其他设备,例如手持设备、平板电脑、其他薄型笔记本电脑、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)交换机或可执行下面教导的功能和操作的任何其他系统。此外,这里描述的装置、方法和系统不限于物理计算设备,还可以涉及用于节能和效率的软件优化。

[0022] 随着计算系统的发展,其中的组件变得更加复杂。结果,在组件之间耦合和通信的互连架构在复杂性方面也在增加,以确保满足针对最佳组件操作的带宽要求。此外,不同的细分市场需要互连架构的不同方面以满足市场需求。例如,服务器需要更高的性能,而移动生态系统有时会牺牲整体性能以节省电力。然而,大多数结构的独特目的是提供最高性能和最大功率节省。下面,讨论了许多互连,这些互连将潜在地受益于本文所述的本发明的各方面。

[0023] 一种互连结构架构包括外围快速组件互连(PCI)(PCIe)架构。PCIe的主要目标是使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(台式机和移动)、服务器(标准和企业)以及嵌入式和通信设备。快速PCI是一种针对各种未来的计算和通信平台的高性能、通用I/O互连。一些PCI属性,例如其使用模型、加载-存储架构和软件接口,已通过其修订版进行维护,而先前的并行总线实现已被高度可扩展的、完全串行接口所取代。最近的快速PCI版本利用了点对点互连、基于交换机的技术和分组协议的优势以提供更高水平的性能和特征。电源管理、服务质量(QoS)、热插拔/热交换支持、数据

完整性和错误处理是快速PCI支持的高级功能当中的一些。

[0024] 参见图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包括耦合到控制器集线器115的处理器105和系统存储器110。处理器105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。

[0025] 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。

[0026] 在一个实施例中,控制器集线器115是快速外围组件互连(PCIe或PCIE)互连层级中的根集线器、根复合体或根控制器。控制器集线器115的示例包括芯片组、存储器控制器集线器(MCH)、北桥、互连控制器集线器(ICH)、南桥和根控制器/集线器。通常,术语芯片组指的是两个物理上分离的控制器集线器,即耦合到互连控制器集线器(ICH)的存储器控制器集线器(MCH)。注意,当前系统通常包括与处理器105集成的MCH,而控制器115以与下面描述的类似方式与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。

[0027] 这里,控制器集线器115通过串行链路119耦合到交换机/桥120。输入/输出模块117和121(也可以称为接口/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。

[0028] 交换机/桥120将来自设备125分组/消息向上游(即,朝向根复合体顺着层次结构向上)路由到控制器集线器115并且向下游(即远离根控制器顺着层次结构向下)从处理器105或系统存储器110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组件,例如I/O设备、网络接口控制器(NIC)、附加卡、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、扫描仪和其他输入/输出设备。通常在PCIe本地语言(vernacular)中,例如设备,被称为端点。尽管没有具体示出,但是设备125可以包括PCIe到PCI/PCI-X桥,用于支持传统或其他版本的PCI设备。PCIe中的端点设备通常被分类为传统、PCIe或根复合体集成端点。

[0029] 图形加速器130还通过串行链路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。然后将交换机120和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和控制器集线器115之间进行通信。类似于上面的MCH讨论,图形控制器或图形加速器130本身可以集成在处理器105中。

[0030] 转到图2,图2示出了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,例如快速路径互连(QPI)栈、PCIe栈、下一代高性能计算互连栈或其他分层栈。尽管下面参考图1-4进行的讨论涉及PCIe栈,但是相同的概念可以应用于其他互连栈。在一个实施例中,协议栈200是PCIe协议栈,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。

[0031] 快速PCI使用分组在组件之间传递信息。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传送到接收组件。当发送的分组流经其他层时，利用用于处理这些层的分组所需的附加信息来对它们进行扩展。在接收侧，发生反向过程，并且分组从其物理层220表示变换到数据链路层210表示，并且最终(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。

[0032] 事务层

[0033] 在一个实施例中，事务层205用于提供设备的处理核和互连架构之间的接口，例如数据链路层210和物理层220。在这方面，事务层205的主要职责是分组的汇编和反汇编(即事务层分组或TLP)。转换层205通常管理TLP的基于信用的流控制。PCIe实现分离事务，即具有由时间分隔的请求和响应的事务，允许链路在目标设备收集响应的数据时携带其他业务。

[0034] 此外，PCIe利用基于信用的流控制。在该方案中，设备在事务层205中为每个接收缓冲器通告初始信用量。在链路的相对端处的外部设备，例如图1中的控制器集线器115，计算由每个TLP消费的信用数。如果事务未超过信用限制，则可以发送事务。在接收到响应后，将恢复一定量的信用。信用方案的优点是，如果没有遇到信用限制，则信用返还的延迟不会影响性能。

[0035] 在一个实施例中，四个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用于将数据传送到存储器映射位置或从存储器映射位置传送数据的读请求和写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位地址，或长地址格式，例如64位地址。配置空间事务用于访问PCIe设备的配置空间。到配置空间的事务包括读请求和写请求。定义消息事务被定义为支持PCIe代理之间的带内通信。

[0036] 因此，在一个实施例中，事务层205组装分组报头/有效载荷156。当前分组报头/有效载荷的格式可以在PCIe规范网站的PCIe规范中找到。

[0037] 快速参考图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。在这方面，事务描述符300支持系统中的事务的识别。其他潜在用途包括跟踪默认事务排序的修改以及事务与信道的关联。

[0038] 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。

[0039] 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，并且对于需要完成该请求代理的所有未完成请求它是唯一的。此外，在该示例中，源标识符310唯一地标识PCIe层次结构中的请求者代理。因此，与源ID 310一起，本地事务标识符308字段提供层次结构域中的事务的全局标识。

[0040] 属性字段304指定事务的特征和关系。在这方面，属性字段304可能用于提供允许修改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、保留字段314、排序字段316和非窥探字段318。这里，优先级子字段312可以由发起者修改以向事务分配优先级。保留属性字段314被保留用于将来或供应商定义的使用。可以使用保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。

[0041] 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”表示要应用的默认排序规则，其中排序属性“1”表示放宽排序，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同一个方向上传递写入。窥探属性字段318用于确定是否窥探了事务。如图所示，信道ID字段306标识与事务相关联的信道。

[0042] 链路层

[0043] 链路层210(也称为数据链路层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的职责是提供用于在链路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组装的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，并将修改的TLP提交给物理层220用于跨物理到外部设备的传输。

[0044] 物理层

[0045] 在一个实施例中，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221的“数字”功能。在这方面，逻辑子块包括用于准备用于由物理子块222发送的传出信息的发送部分，以及用于在将所接收的信息传递到链路层210之前识别和准备所接收的信息的接收部分。

[0046] 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。接收器被提供来自外部设备的串行化的符号，并将接收的信号转换为比特流。比特流被反序列化并提供给逻辑子块221。在一个实施例中，采用8b/10b传输码，其中发送/接收10比特符号。这里，特殊符号用于利用帧223对分组进行成帧。此外，在一个示例中，接收器还提供从传入串行流中恢复的符号时钟。

[0047] 如上所述，尽管参考PCIe协议栈的特定实施例讨论了事务层205、链路层210和物理层220，但是分层协议栈不限于此。实际上，可以包括/实现任何分层协议。作为示例，表示为分层协议的端口/接口包括：(1)用于组装分组的第一层，即事务层；用于对分组进行排序的第二层，即链路层；以及用于发送分组的第三层，即物理层。作为具体示例，使用公共标准接口(CSI)分层协议。

[0048] 接下来参考图4，示出了PCIe串行点对点结构的实施例。尽管示出了PCIe串行点对点链路的实施例，但是串行点对点链路不限于此，因为它包括用于发送串行数据的任何传输路径。在所示的实施例中，基本PCIe链路包括两个低压差分驱动信号对：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的传输逻辑406以及用于从设备410接收数据的接收逻辑407。换句话说，在PCIe链路中包含两个发送路径，即路径416和417，以及两个接收路径，即路径418和419。

[0049] 传输路径指的是用于发送数据的任何路径，例如传输线、铜线、光线、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-每个通道代表一组差分信号对(一对用于传输，一对用于接收)。为了对带宽进行缩放，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。

[0050] 差分对指的是两个传输路径，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、振铃等。这允许更好的定时窗口,这使得更快的传输频率成为可能。

[0051] 新的和不断增长的使用模型(例如,基于PCIe的存储阵列和雷电接口(雷电接口))正在推动PCIe层次结构深度和宽度的显着增加。快速PCI(PCIe)架构基于PCI,它定义了“配置空间”,系统固件和/或软件在其中发现功能并启用/禁用/控制它们。该空间中的寻址基于16位地址(通常称为“BDF”或总线设备功能编号),由8位总线编号、5位设备编号和3位功能编号组成。在PCIe中,总线编号可以指逻辑总线而不是物理总线。除了用于在配置空间中寻址PCI功能,以及为诸如错误报告和IO虚拟化之类的目的识别特定功能外,空间本身也可以被视为一种类似于其他资源受分配和管理问题影响的资源。

[0052] PCI允许系统提供多个独立的BDF空间,其被称为“段”。每个段可能具有某些资源要求,例如用于生成PCI/PCIe配置请求的机制,包括PCIe规范中定义的增强配置访问机制(ECAM)。此外,输入/输出(I/O)存储器管理单元(IOMMU)(如Intel VT-d)可以使用BDF空间作为索引,但可能无法被配置为直接理解段。因此,在某些实例中,针对系统中定义的每个段复制单独的ECAM和IOMMU。图5示出了包括多个段(例如,505a-c)的系统的示例。例如,在该示例中,针对连接到根复合体525的三个交换机510、515、520中的每一个定义段。在该示例中,可以在根复合体525处实现单独的IOMMU和ECAM(例如,530a-c)以促进段(例如,505a-c)中的每个。此外,在该示例中,各种端点(EP)连接到每个段中的各种总线。在某些实例中,段的配置空间可以针对潜在的热插拔事件保留多个总线地址,从而限制每个段中可用的总线地址总数。更进一步地,在段中的一个或多个中分配总线编号可以根据一种算法,该算法本身很少涉及密集填充地址和紧凑使用可用总线地址空间。在某些实例中,这会导致浪费的配置地址(即BDF)空间。

[0053] 传统的PCIe系统被配置为以这样的方式分配地址空间:当应用于现代和新兴的用例时,倾向于低效地使用BDF空间并且特别是总线编号。虽然相对较少的实现方式实际上可能涉及单个系统消耗唯一BDF值的所有64K(例如,在PCIe下定义),但深层次结构(例如,在PCIe交换机的深层次结构中出现的层次结构)可能会非常快速耗尽可用的总线编号。此外,在支持热插拔的应用中,BDF空间的大部分通常可以保留用于将来的潜在用途(即,当未来的设备被热插入系统时),从池中获取系统立即可用的额外的几行总线编号。虽然段机制可用于解决此问题,但段本身的可缩放性较差,因为如上所述,额外的硬件资源(例如,IOMMU)将内置到CPU、平台控制器集线器(PCH)、片上系统(SoC)、根复合体等,以支持每个段。因此,使用段来解决深层次结构导致对系统进行缩放以满足最坏情况的系统要求,这通常远远超过大多数系统所需的系统要求,导致平台资源的显著浪费。此外,在系统的根复合体之外创建段可能是困难的(并且在某些情况下,基本上是不可能的)。

[0054] 在一些实现方式中,可提供系统以实现BDF空间的更高效使用且解决上文中的至少一些示例性问题。这可以允许扩展PCIe、雷电接口、片上系统结构(例如,英特尔片上系统结构(IOSF)以及其他),以及与非常大拓扑的其他互连,但不需要根复合体中的专用资源,如在完全依赖于段或其他替代方案的解决方案中就是这种情况。图6示出了根据示例PCIe BDF分配将总线编号分配给系统中的总线的示例。在该示例中,使用传统的BDF分配(如通过圆形标签(例如,650a-d等)指定的那样),具有直接连接到根复合体615的两个设备605、610和两个基于交换机的层次结构(对应于交换机620、625)的系统利用近似最密集的总线编号

分配被列举。在深层次结构中，可以快速消耗单个BDF空间中的可用总线编号。此外，现实世界系统通常不太高效地分配总线编号，导致BDF空间的稀疏(或“浪费”)分配。

[0055] 支持热添加/移除的用例(例如雷电接口，在某些情况下，基于PCIe的存储)的另一个问题是BDF空间中的总线编号分配被“重新平衡”以解决在运行系统中发生的硬件拓扑变化。然而，重新平衡对于系统软件来说可能非常困难，因为在典型情况下，所有PCI功能随后被强制进入与重新平衡相关的静止状态，以便允许系统重新枚举BDF空间，然后重新启用PCI功能。然而，该过程可能非常慢，并且通常导致系统在非常长的时间段内冻结(例如，足够长以破坏运行的应用程序，并且对最终用户可见)。还可以提供改进的系统以缩短应用修订的BDF空间所花费的时间，使得重新平衡过程可以在百分之一毫秒或更快的范围内执行，并且没有明确地将PCI功能置于静止状态。最后，可以定义非常大的系统或具有用于支持多个根复合体的(专有)机制的系统以要求使用段。

[0056] 如上所述，扁平化端口桥(portal bridge)(FPB)可以是可选机制，可用于解决至少一些上述示例问题，包括改进总线/设备/功能(BDF)和存储器映射IO(MMIO)空间的可缩放性和运行时重新分配。“BDF空间”的概念与配置地址空间有关，但概括为认识到BDF是请求者和完成者ID的基础，完成的路由，并且除了对配置请求的路由之外，可以作为若干机制中的基本元素。对于与上游端口相关联的功能，BDF空间地址的功能编号部分(例如，3位功能编号)可以通过上游端口硬件的结构来确定，而总线和设备编号部分可以由上游端口上方的下游端口确定。示例FPB可以维持现有架构，其中上游端口确定BDF的3位功能编号部分中的功能的映射，并且仅在13位总线/设备编号部分内操作。在这种实例中，“BD空间”可以指BDF的13位总线/设备编号部分。这里的MMIO可以特指通过根端口、交换机端口或逻辑桥的存储器读和写请求，其中FPB能力提供额外的机制来确定这些请求的地址解码。实现FPB能力的桥本身也可以称为FPB。

[0057] 图7A是示出了在系统中的一个或多个交换机的每个端口处提供FPB逻辑的简化框图。某些交换机可能不包括FPB逻辑，并且仅支持根据传统BDF或MMIO空间分配的总线枚举。此外，根复合体705的端口还可以包括FPB逻辑，诸如设计用于潜在地支持动态情况热插拔或灵活地支持各种架构(例如，在根复合体的设计时未预定义)的端口，以及其他示例。此外，根复合体705的一些端口可以省略FPB逻辑，诸如设计用于支持静态情况的端口，例如在图7A的示例中连接到端点710、715的端口，以及其他示例。

[0058] 可以在拥有FPB逻辑的每个端口处启用或禁用FPB逻辑。FPB逻辑可以用硬件、固件和/或软件实现，以支持使用“扁平化”方法分配BDF或MMIO空间。例如，互连交换机、端点和根复合体的每个总线可以被分配唯一的总线编号(例如，如结合图6所讨论的)，其中多个设备能够由每个总线编号下的32个可能的设备编号中的相应一个来寻址，以及多个可能的功能编号能够由每个设备编号下的8个可能功能编号中的相应一个来寻址。通过FPB的扁平化可以允许总线中的至少一些通过总线设备(BD)编号组合(而不是通过唯一的总线编号)唯一地寻址，从而将唯一潜在总线地址的最大数量从256(根据传统PCIe总线编号枚举)扩展至8192(根据BD总线枚举)。因此，在FPB下，总线编号可以重复用于几个不同的总线，尽管每个总线编号具有唯一的BD编号(即，BD的总线编号部分是相同的但是该总线编号下的设备编号不同)。系统的一些分支可以利用传统的总线寻址(例如，通过BDF总线编号)，而其他分支利用基于FPB的寻址。实际上，可以采用总线枚举方案的混合，结果仍然是在系统内使用

唯一的总线地址。

[0059] 转到图7B,该图是示出了FPB逻辑块的示例实现方式的高级架构的简化框图(例如在图7A的示例中所示的端口处所示出的)。可以通过每个FPB模块提供类型1桥功能。桥功能可以支持传统分组解码/路由机制(例如,传统PCIe BDF解码和路由)以及FPB分组解码/路由机制。

[0060] 通过使交换机内部和下游端口使用总线编号的方式“扁平化”,FPB改变了交换机消耗BDF资源的方式以减少浪费。FPB定义了系统软件在非连续范围内分配BDF和MMIO资源的机制,使系统软件能够分配BDF/MMIO池,从中可以将“箱”分配给FPB以下的功能。这允许系统软件分配设备热添加所需的BDF/MMIO,而不必重新平衡其他、已分配的资源范围,并返回释放的池资源,例如,通过热删除事件。FPB被定义为允许传统机制和新机制同时操作,例如,使得系统固件/软件可以实现一种策略,其中传统机制继续在系统的部分中使用,其中可以不需要FPB机制。在图7B的示例中,当给定TLP被解码为与桥的辅助侧相关联时,可以假设解码逻辑提供“1”输出。传统解码机制可以如前所述地应用,因此例如,除了其他示例之外,仅可以由传统BDF解码逻辑来测试BDF地址的总线编号部分(比特15:8)。

[0061] 如图8的示例中所示,FPB逻辑的实例可以包括传统分组解码/路由机制和FPB分组解码/路由机制。用于传统分组解码/路由机制。可以在FPB被提供的端口处识别TLP,并且传统的分组解码/路由机制可以基于传统的BDF路由来确定是将TLP路由到端口的辅助侧还是将TLP路由保持在主要侧。同样,FPB分组解码/路由机制可以基于扁平化的BDF路由来确定是将TLP路由到端口的辅助侧还是将TLP路由保持在主要侧。如果传统分组解码/路由机制或FPB分组解码/路由机制输出分组应该进入辅助侧的指示,则分组的路由遍历桥到辅助侧以便路由到其目的地。传统分组解码/路由机制可以包括,对于BDF解码,辅助/从属总线编号寄存器,以及对于存储器(例如,MMIO)解码,存储器基址/限制寄存器、可预取基址/限制寄存器、VGA使能位、增强分配,以及FPB逻辑可用的其他机制。FPB分组解码/路由机制可以包括对于BD空间解码、BD辅助开始、矢量开始、粒度和相关寄存器,以与BD矢量结合使用。存储器解码还可以利用FPB中的矢量,例如MEM低矢量,用于与MEM低矢量开始、粒度以及相关MEM低寄存器结合使用,以及MEM高矢量,用于与MEM高矢量开始、粒度和相关的MEM高寄存器结合使用,以及其他示例寄存器、机制、功能和特征。

[0062] 在一些情况下,尽管FPB可以针对特定桥添加额外的方式来解码给定的TLP,但是FPB可能不会改变关于桥在交换机和根复合体架构结构内操作的基本方式的任何信息。在一个示例中,FPB使用相同的架构概念来为三种不同的资源类型提供管理机制:BDF(“BD”)的总线/设备空间,比特15:3;低于4GB的存储器(“MEM低”);以及超过4GB的存储器(“MEM高”)。允许FPB的硬件实现方式支持这三种机制的任何组合。对于每种机制,FPB使用位矢量来指示对于所选资源类型的特定子集范围,该范围中的资源是否与FPB的主要侧或辅助侧相关联。可以允许硬件实现方式来实现这些矢量的小范围的大小,并且通过选择在增加BD/地址顺序中应用矢量的初始偏移以及矢量内各个比特的粒度来指示给定矢量中的比特应用的BD/地址资源集的大小,来使系统固件/软件能够最有效地使用可用矢量。

[0063] 对于每个BD/Mem低/Mem高机制,特别是对于根复合体,可能需要提供一种机制,例如,配置寄存器,硬件或系统特定的固件/软件可以通过该机制限制允许系统软件分配给根端口桥的辅助侧的BD/MMIO的允许范围。这可以简化多组件根复合体的构建,例如,通过确

保系统软件不会尝试将FPB机制应用于在多组件根复合体的不同组件上实现的根端口之间的BD/MMIO空间的重叠范围，例如，使得允许一个组件上的根端口在给定的BD/MMIO资源范围内工作，并允许另一个组件上的根端口工作在BD/MMIO资源的不同的、与第一范围不重叠的范围内，以及其他示例。

[0064] 在静态用例(或简称为“静态情况”)中，由于针对交换机的PCI/PCIe架构定义以及下游端口将整个总线编号与其链路相关联的传统要求导致的总线和设备编号“浪费”，因此层次结构的大小和端点的数量存在限制。在一些实现方式中，可以通过“扁平化”BDF空间的使用来解决这类问题，使得交换机和下游端口能够更高效地使用可用空间。对于动态使用案例(或简称为“动态用例”)，通过在相关端点上方的桥中保留大范围的总线编号和存储器映射I/O(MMIO)来避免重新平衡，以尝试满足预先分配的范围中的任何需求。然而，这种方法导致额外的浪费，这放大了传统BDF分配的缺点。此外，这种方法在一般情况下可能难以实现，即使对于相对简单的情况，例如，可能有一个实现单个端点的固态驱动器(SSD)替换为具有交换机的单元，创建单元内部的内部层次结构，使得虽然只有一个总线的初始分配就足够了，但初始分配会立即中断新单元。此外，对于MMIO，当热插拔端点可能需要分配低于4GB的MMIO空间(由于其本质上是有限的资源)而这很快被甚至相对少量的预先分配用尽时，预先分配方法可能是有问题的，并且由于多个系统元件要求系统地址空间分配低于4GB，因此预先分配不具吸引力。取决于包括给定系统的物理存储器寻址能力在中的多种因素，在某些情况下，MMIO空间中的资源限制也可能高于4GB。适用于低于4GB的MMIO空间的约束可能与应用于4GB以上的约束不同(因此可以针对每个约束来优化单独的机制)。

[0065] 在一些实现方式中，静态和动态使用案例中的至少一些问题可以通过定义用于针对BDF和MMIO两者启用不连续的资源范围(重新/)分配的机制来解决。系统软件可以拥有维护资源“池”的能力，可以在运行时分配(并释放回)，而不会像重新平衡那样需要中断正在进行的其他操作。因此，扁平化端口桥(FPB)可以作为可选能力提供，可以通过根和交换机端口中的类型1(桥接)功能来实现，以支持更高效和密集的BDF分配，从而无需重新平衡系统中其他地方分配的资源即可重新分配BDF资源，以及启用不连续的MMIO区域，并避免重新平衡MMIO资源的需要。在一些实现方式中，IO空间分配可以保持原样并且不由FPB修改。在示例FPB提供的潜在示例优势中，BDF空间分配可以更高效和密集，从而实现更大的层次结构，可以针对热添加/删除案例启用资源的运行时重新分配，而无需全局重新平衡资源，可以停用对在连续范围内分配的BDF和MMIO的需要，可以支持混合系统，包括支持FPB的组件以及不支持FPB的组件，同时不允许更改现有的离散端点。例如，传统根复合体、交换机、桥和端点可以在混合系统环境与实现FPB的RC和交换机一起使用。

[0066] 在一些实现方式中，FPB可以包括提供支持FPB的新硬件和软件。但是，可以任选地启用此附加硬件和/或软件，除非启用它，否则它将无效，默认情况下禁用。在某些实例中，用于实现FPB的硬件更改可能包括涉及类型1功能的硬件，同时允许支持类型0功能的端点和硬件保持不受影响。启用FPB的硬件可以通过现有的合规性和互操作性测试，并且可以开发新的测试来明确评估额外的FPB功能。打算与实现FPB功能的设备一同工作的软件可以被配置为理解新扩展的能力。传统软件将继续使用FPB硬件来运作，但无法使用FPB特征。

[0067] FPB可以继续允许使用用于BDF和MMIO的传统资源分配机制。在某些情况下，可能需要让系统固件仅使用传统机制继续执行初始系统资源分配，并且在操作系统启动后仅使

用FPB。FPB可以支持这一点，并且特别使系统能够继续使用遗留机制分配的资源。FPB专门用于使系统软件能够在运行时期间修改系统中的资源分配，仅需要使与被修改的资源相关联的硬件和进程保持静止，并允许所有其他硬件和进程继续正常操作。

[0068] 为了通过系统软件支持FPB的运行时使用，FPB硬件实现方式应该避免向飞行中的事务引入停顿或其他类型的中断，包括在系统软件正在修改FPB硬件的状态的时间期间。但是，不期望的是硬件尝试识别系统软件以确实影响飞行中的事务的方式错误地修改FPB配置的情况。与传统机制一样，系统软件负责确保系统操作不会因重新配置操作而被破坏。没有明确要求系统固件/软件以特定顺序执行FPB机制的启用和/或禁用，但是可以定义规则以在层次结构中实现资源分配操作，使得系统的硬件和软件元件不被破坏或导致故障。

[0069] 在一些实现方式中，如果系统软件违反关于FPB的任何规则，则可以不定义硬件行为。FPB可以在任何PCI桥(类型1)功能中实现，并且实现FPB的每个功能都实现FPB扩展能力。如果交换机实现FPB，则交换机的上游端口和所有下游端口实现FPB。可以允许根复合体在某些根端口上实现FPB，但在其他端口上不实现。可以允许根复合体在根复合体的内部逻辑总线上实现FPB。允许类型1功能实现适用于这些元素机制(BD、MEM低，MEM高)中的任何一个、两个或三个的FPB机制。可以允许系统软件启用特定FPB支持的元素机制的任何组合(包括全部或不包括任何一个)。除本节中明确修改的情况外，错误处理和报告机制可能不受FPB的影响。在FPB功能复位的情况下，FPB硬件清除所有实现的矢量中的所有位。一旦启用(例如，通过FPB BD矢量使能、FPB MEM低矢量使能和/或FPB MEM高矢量使能位)，如果系统软件随后禁用FPB机制，则相关联的矢量中的条目值未定义，并且如果系统软件随后重新启用FPB机制，则FPB硬件清除相关联的矢量中的所有位。

[0070] 在一些实现方式中，当启用相对应的FPB机制时，明确允许系统软件修改FPB矢量。如果FPB是在No_Soft_Reset位清零的情况下实现的，那么当FPB循环通过D0→D3hot→D0时，那么，与其他功能配置上下文一样，所有FPB机制都将被禁用，并且FPB将清除所有实现矢量的所有位。如果使No_Soft_Reset位置位来实现FPB，则当该FPB循环通过D0→D3hot→D0时，则与其他功能配置上下文一样，所有FPB配置状态都不会更改，并且FPB矢量中的条目将通过硬件保留。可以实现硬件，使得不存在对FPB计算执行任何类型的边界检查的要求，并且系统软件可以确保FPB参数被正确编程。例如，可以允许系统软件对矢量起始值进行编程，该矢量起始值导致相对应的矢量的高阶位超过与给定FPB相关联的资源范围，系统软件确保矢量的那些高阶位被清除。系统软件必须避免的错误的示例包括资源分配的重复，可以创建“环绕”或边界错误的起始偏移与置位矢量位的组合，以及其他示例。

[0071] 在FPB BD机制的一些实现方式中，如果特定BDF落入由与编程到BD矢量中相应条目的值进行逻辑“或”运算的辅助和从属总线编号寄存器中编程的值指示的总线编号的范围内，则FPB硬件将该BDF视为与FPB的辅助侧相关联。当仅使用FPB BD机制进行BDF解码时，可以利用系统软件来确保辅助和从属总线编号寄存器都为0。系统软件可以进一步确保配置FPB路由机制，使得FPB的配置请求目标功能辅助侧将由FPB从主要侧路由到FPB的辅助侧。FPB BD机制可以应用不同的粒度，可通过FPB BD矢量控制1寄存器中的FPB BD矢量粒度寄存器由系统软件编程。例如，图9示出了BDF空间和支持的粒度中的示例地址。图9中的表示示出了BDF空间中的地址布局与支持的粒度之间的关系。

[0072] 在一些实现方式中，系统软件根据在那些字段的描述中描述的约束来对FPB BD矢

量控制1寄存器中的FPB BD矢量粒度和FPB BD矢量开始字段进行编程。FPB(与交换机的上游端口相关联的FPB除外)可能受到约束,使得当不支持PCIe替换路由ID解释(ARI)转发时,或者当设备控制2寄存器中的ARI转发使能位为清除时,当类型1配置请求的BD地址(BDF的比特15:3)与FPB BD矢量控制2寄存器中的BD辅助开始字段中的值匹配时,FPB硬件将在FPB的主要侧上接收的类型1配置请求转换为FPB的辅助侧上的类型0配置请求,并且系统软件必须相应地配置FPB。当设备控制2寄存器中的ARI转发使能位被置位时,当类型1配置请求的总线编号地址(BDF的比特15:8)匹配FPB BD矢量控制2寄存器中的辅助开始字段的总线编号地址(仅比特15:8)中的值时,FPB硬件将FPB的主要侧接收的类型1配置请求转换为FPB的辅助侧的类型0配置请求,并且系统软件必须相应地配置FPB。

[0073] 在一些实现方式中,对于仅与交换机的上游端口相关联的FPB,FPB硬件可以使用FPB能力寄存器的FPB Num Sec Dev字段来指示与上游端口桥的辅助侧相关联的设备编号的数量,其可以是除了FPB BD矢量控制2寄存器中的BD辅助开始字段之外,FPB使用该数量来确定FPB的主要侧接收的配置请求针对交换机的下游端口中的一个,实际上确定该请求何时将从类型1配置请求转换为类型0配置请求,系统软件适当地配置FPB。如果在下游端口处启用ACS源验证,则FPB将检查端口收到的每个上游请求的请求者ID,以确定它是否映射到FPB的辅助侧,并且如果请求者ID不映射到FPB的辅助侧,那么这可以构成报告的与接收端口关联的错误(例如,ACS违规)。FPB可以进一步实现针对INTx虚拟线路的桥接映射。

[0074] 在一个示例中,为了确定FPB BD矢量中的哪个条目应用于给定的BDF地址,FPB提供的硬件和软件可以应用诸如以下的算法:

```
// “BDF”是要被测试的BDF地址
IF (BDF <= FPB_BD_Vector_Start) THEN
   EXIT;  // 在这种情况下，BFD不在范围内
// 否则，应用起始偏移
OffsetIndex := BDF – FPB_BD_Vector_Start;
// 针对矢量的粒度的下一个调整
//  （该操作是除法，此处完成为
//  利用0填充的右移
OI_Gran_Adjusted :=
   ShiftRightZeroFill(FPB_BD_Vector_Granularity,
                                    OffsetIndex);
IF (OI_Gran_Adjusted >= LENGTHOF(FPB_BD_Vector)) THEN
   EXIT;  // 在这种情况下，BDF不在范围内
// 否则，将该比特定位在（bit addressed)矢量中
SelectorBit := FPB_BD_Vector[OI_Gran_Adjusted];
//  如果SelectorBit被置位，则BDF被考虑为
//  在辅助侧，如果被清除，则BDF
//  被考虑为在主要侧
```

[0075]

[0076] 换句话说，为了确定FPB BD矢量中的哪个条目应用于给定的BDF地址，逻辑可以确定BD地址是否低于FPB BD矢量开始的值。如果BD地址低于FPB BD矢量开始的值，则BD不在范围内并且不与桥的辅助侧相关联。否则，逻辑可以通过首先减去FPB BD矢量开始的值，然后根据FPB BD矢量粒度的值将其除以该值来确定矢量中的比特索引，从而计算矢量中的偏移。如果位索引值大于由支持的FPB BD矢量大小指示的长度，则BD不在范围内(高于)并且不与桥的辅助侧相关联。然而，如果计算的位索引位置处的矢量中的位值是1b，则BD地址与桥的辅助侧相关联，否则BD地址与桥的主要侧相关联。

[0077] FPB MEM低机制可以应用不同的粒度，可通过FPB MEM低矢量控制寄存器中的FPB

MEM低矢量粒度寄存器由系统软件编程。图10示出了FPB MEM低机制应用于的4GB以下的存储器地址空间中的地址布局以及粒度对这些地址的影响。图10还涉及扁平化端口桥(FPB)扩展能力的定义。系统软件可以根据这些字段描述中描述的约束对FPB MEM低矢量控制寄存器中的FPB MEM低矢量粒度和FPB MEM低矢量开始字段进行编程。

[0078] 在FPB MEM低机制的实例中,如果该存储器地址落入由在与编程到MEM低矢量中相应条目的值进行逻辑或运算的其他桥接存储器解码寄存器(下面列举)中编程的值指示的任何范围内,则FPB硬件可以将特定存储器地址视为与FPB的辅助侧相关联。其他桥接存储器解码寄存器可以包括:类型1(桥接)报头中的存储器基址/限制寄存器;类型1(桥接)报头中的可预取的基址/限制寄存器;类型1(桥接)报头的桥接控制寄存器中的VGA使能位;增强分配(EA)能力;FPB MEM高机制(如果支持并启用)。在一个示例中,为了确定FPB MEM低矢量中的哪个条目适用于给定的存储器地址,硬件和软件可以应用诸如以下的算法:

```
// "Address" is the memory address to be tested
// expressed in MB units (i.e. bits [31:20])
IF (Address <= FPB_MEM_Low_Vector_Start) THEN
   EXIT;   // In this case the address is out of range
// Otherwise, apply Starting offset
OffsetIndex := Address – FPB_MEM_Low_Vector_Start;
// Next adjust for the granularity of the vector
// (this operation is a divide, done here
// as a right shift with zero fill)
OI_Gran_Adjusted :=
```

[0079]

```
   ShiftRightZeroFill(FPB_MEM_Low_Vector_Granularity,
                                        OffsetIndex);
IF (OI_Gran_Adjusted >=
      LENGTHOF(FPB_MEM_Low_Vector)) THEN
   EXIT;   // In this case the address is out of range
// Otherwise, locate the bit in the (bit addressed) vector
SelectorBit := FPB_MEM_Low_Vector[OI_Gran_Adjusted];
// If SelectorBit is Set, then the address is
// considered to be on the Secondary Side,
// if Clear, the address is considered to be on
// the Primary Side.
```

[0080] 换句话说，为了确定FPB MEM低矢量中的哪个条目应用于给定的存储器地址，硬件和软件可以确定存储器地址是否低于FPB MEM低矢量开始的值。如果是这样，则存储器地址

可能不在范围内(低于)并且不与桥的辅助侧相关联。逻辑可以通过首先减去FPB MEM低矢量开始的值,然后根据FPB MEM低矢量粒度的值除以该值以确定矢量中的比特索引来计算矢量中的偏移。如果位索引值大于支持的FPB MEM低矢量大小所指示的长度,则存储器地址不在范围内(高于),并且不与桥的辅助侧相关联。另一方面,如果计算的位索引位置处的矢量中的位值是1b,则存储器地址可以与桥的辅助侧相关联,否则存储器地址与桥的主要侧相关联。

[0081] 系统软件可以根据这些字段的描述中所描述的约束对FPB MEM高矢量控制1寄存器中的FPB MEM高矢量粒度和FPB MEM高矢量开始较低字段进行编程。在FPB MEM高机制的情况下,如果该存储器地址落在由与编程到MEM低矢量中相应条目中的值进行逻辑或运算的其他桥接存储器解码寄存器(下面枚举)中编程的值指示的任何范围内,则FPB硬件可以将特定的存储器地址视为与FPB的辅助侧相关联。其他桥接存储器解码寄存器可以包括类型1(桥接)报头中的存储器基址/限制寄存器;类型1(桥接)报头中的可预取的基址/限制寄存器;类型1(桥接)报头的桥接控制寄存器中的VGA使能位;增强分配(EA)能力;以及FPB MEM低机制(如果支持并启用)。在一个示例中,为了确定FPB MEM高矢量中的哪个条目适用于给定的存储器地址,硬件和软件可以应用诸如以下的算法:

```
// "Address" is the memory address to be tested
// expressed in 16MB units (i.e. bits [63:24])
// "FPB_MEM_High_Vector_Start" is the concatenation
// of FPB MEM High Vector Start Upper and
// FPB MEM High Vector Start Lower
```

[0082]

```
IF (Address <= FPB_MEM_High_Vector_Start) THEN
   EXIT;   // In this case the address is out of range
// Otherwise, apply Starting offset
OffsetIndex := Address – FPB_MEM_High_Vector_Start;
// Next adjust for the granularity of the vector
// (this operation is a divide, shown here
// as a right shift with zero fill)
// We have to apply an additional shift of 4 bits
// to account for the granularity units
OI_Gran_Adjusted :=
   ShiftRightZeroFill(
      ShiftRightZeroFill(FPB_MEM_High_Vector_Granularity,
                                    OffsetIndex), 4);
IF (OI_Gran_Adjusted >=
      LENGTHOF(FPB_MEM_High_Vector)) THEN
   EXIT;   // In this case the address is out of range
// Otherwise, locate the bit in the (bit addressed) vector
SelectorBit := FPB_MEM_High_Vector[OI_Gran_Adjusted];
// If SelectorBit is Set, then the address is
// considered to be on the Secondary Side,
// if Clear, the address is considered to be on
// the Primary Side.
```

[0083]

[0084] 换句话说，为了确定FPB MEM高矢量中的哪个条目应用于给定的存储器地址，硬件和软件可以确定存储器地址是否低于FPB MEM高矢量开始的值。如果是，则可以确定存储器

地址不在范围内(低于)并且不与桥的辅助侧相关联。否则,矢量中的偏移可以通过首先减去FPB MEM高矢量开始的值,然后根据FPB MEM高矢量粒度的值除以该值以通过该机制确定矢量中的比特索引来计算。如果位索引值大于由支持的FPB MEM高矢量大小指示的长度,则存储器地址不在范围内(高于),并且因此不与桥的辅助侧相关联。否则,如果计算的位索引位置处的矢量中的位值是1b,则存储器地址与桥的辅助侧相关联,或者存储器地址与桥的主要侧相关联。

[0085] 在一些实现方式中,FPB可以使用位矢量机制来描述地址空间(BD空间、MEM Lo和Mem Hi)。支持FPB的桥可以在支持FPB使用的每个地址空间中包含以下内容:位矢量;起始地址寄存器;以及粒度寄存器。桥可以使用这些值来确定给定地址是否是由FPB解码的与桥的辅助侧相关联的范围的一部分。使用传统解码机制和FPB解码机制中的任一个或两者确定不与桥的辅助侧相关联的地址(默认情况下)与桥的主要侧相关联。这里,术语“关联”可以表示,例如,桥将对TLP应用以下处理:

[0086] • 与主要侧相关联并在主要侧接收的TLP可以作为不支持的请求(UR)处理;

[0087] • 与主要侧相关联并在辅助侧接收的TLP可以作为前向上游处理;

[0088] • 与辅助侧相关联并在主要侧接收的TLP可以作为前向下游处理;

[0089] • 与辅助侧相关联并在辅助侧接收的TLP可以作为不支持的请求(UR)处理,等等。

[0090] 在FPB中,矢量中的每个位可以表示一系列地址,其中该范围的大小由所选择的粒度确定。如果矢量中的位被置位,则表示寻址到相应范围中的地址的分组将与桥的辅助侧相关联。每个位表示的特定地址范围取决于该位的索引,以及起始地址和粒度寄存器中的值。起始地址寄存器指示位矢量所描述的最低地址。粒度寄存器指示由每个位表示的区域的大小。矢量中的每个连续位应用于后续范围,根据粒度随每个位增加。

[0091] 在一些情况下,未启用ARI转发的下游端口仅将设备0与附接到表示来自端口的链路的逻辑总线的设备相关联。针对与指定设备编号0的链路相关联的总线编号的配置请求被传送到附接到该链路的设备。因此,指定所有其他设备编号(1-31)的配置请求可以由具有不支持的请求完成状态的交换机下行端口或根端口终止(相当于PCI中的主设备中止)。在某些情况下,非ARI设备可能不会假设设备编号0与其上游端口相关联,而是捕获其分配的设备编号并响应所有类型0配置读取请求,而不管请求中指定的设备编号。在一些示例中,当针对ARI设备并且其上方的下游端口被启用用于ARI转发时,设备编号被暗示为0,并且传统的设备编号字段被替代地用作8位功能编号字段的一部分。如果配置请求类型为1,则FPB逻辑可以确定总线编号和设备编号字段(在快速PCI-PCI桥的情况下)是否等于分配给辅助PCI总线的总线编号,或者在交换机或根复合体的情况下,是否等于分配给根(根复合体)或下游端口(交换机)之一的总线编号和已解码的设备编号。如果是,则可以将请求转发到该下游端口(或PCI总线,在快速PCI-PCI桥的情况下)。如果不等于任何下游端口或辅助PCI总线的总线编号,但是在分配给下游端口或辅助PCI总线的总线编号的范围内,则可以将请求转发到该下游端口接口而不进行修改。

[0092] 扁平化端口桥(FPB)扩展能力可以是可选的扩展能力,其将被提供用于实现FPB的任何桥功能或端口。如果交换机实现FPB,则交换机的上游端口和所有下游端口实现FPB扩展能力结构。允许根复合体在某些根端口上实现FPB扩展能力结构,但不在其他根端口上实现FPB扩展能力结构。在一些实现方式中,可以允许根复合体实现针对内部逻辑总线的FPB

能力。在以下描述中，凭借PCIe扩展能力来访问FPB寄存器，但是在其他示例实现方式中，可以通过其他手段来访问FPB寄存器，包括但不限于PCI能力结构或供应商定义的扩展能力。在一些实现方式中，寄存器可以托管在对应的交换机、桥、根复合体或系统中的其他设备的存储器元件中。

[0093] 表1示出了FPB扩展能力报头的一个示例实现方式。在一个示例中，FPB扩展能力报头可以具有00h的偏移。

[0094] 表1:FPB扩展能力报头

[0095]

位的位置	寄存器描述
15：0	**PCI 快速扩展能力 ID**-该字段标识以下结构为用于扁平化端口桥（FPB）的扩展能力结构

[0096]

19：16	**能力版本**-该字段是指示当前能力结构的版本的 PCI-SIG 定义的版本号。 针对规范的该版本必须是 1h。
31：20	**下一能力偏移**-该字段包含了在链接的能力列表中不存在其它项的情况下与下一快速 PCI 能力结构的偏移或 000h。 对于在配置空间中实现的扩展能力，该偏移相对于 PCI 兼容配置空间的开始并且因此必须总是为 000h（用于终止能力列表）或大于 0FFh。

[0097] 表2示出了FPB能力报头的一个示例实现方式。在一个示例中，FPB能力报头可以具有04h的偏移。

[0098] 表2:FPB能力寄存器

[0099]

位的位置	寄存器描述
0	**支持的 FPB BD 矢量**-如果置位，则指示支持 BD 矢量机制。
1	**支持的 FPB MEM 低矢量**-如果置位，则指示支持 MEM 低矢量机制。
2	**支持的 FPB MEM 高矢量**-如果置位，则指示支持 MEM 高机制。
7：3	**FPB Num Sec Dev**-对于仅交换机的上游端口，该字段指示与上游端口桥的辅助侧相关联的设备编号的数量。该数量是通过向该字段的数值加一来确定的。 虽然鼓励交换机实现方式高效地消费功能编号，但是显式地允许下游端口分配给在设备编号的所指示范围内不连续的功能编号，并且要求系统软件在与上游端口的辅助侧相关联的设备编号的所指示数量中的每个功能编号处扫描下游端口桥。该字段针对下游端口保留。
10：8	**支持的 FPB BD 矢量大小**-指示以硬件实现的 FPB BD 矢量的大小，并且约束软件允许写入 FPB BD 矢量粒度字段的允许值。

[0100]

	定义的编码为： 值 大小 允许的粒度 000b 256 位 8、16、32、64、128、256 001b 512 位 8、16、32、64、128 010b 1K 位 8、16、32、64 011b 2K 位 8、16、32 100b 4K 位 8、16 101b 8K 位 8 全部其它编码被保留 如果 FPB BD 矢量支持位为清除，则该字段中的值是未定义的并且必须由软件忽略。
15：11	保留
18：16	**支持的 FPB MEM 低矢量大小**-指示以硬件实现的 MEM 低矢量的大小，并且约束允许软件写入 FPB MEM 低矢量开始字段的允许值。 定义的编码为： 值 大小 允许的粒度 000b 256 位 1、2、4、8、16 001b 512 位 1、2、4、8 010b 1K 位 1、2、4 011b 2K 位 1、2 100b 4K 位 1 全部其它编码被保留 如果 FPB MEM 低矢量支持位为清除，则该字段中的值是未定义的并且必须由软件忽略。
23：19	保留
26：24	**支持的 FPB MEM 高矢量大小**-指示以硬件实现的 MEM 低矢量的大小。 定义的编码为： 000b 256 位

[0101]

	001b 512 位 010b 1K 位 011b 2K 位 100b 4K 位 101b 8K 位 全部其它编码被保留 如果 FPB MEM 高矢量支持位为清除，则该字段中的值是未定义的并且必须由软件忽略。
31：27	保留

[0102] 表3示出了FPB BD矢量控制1寄存器的一个示例实现方式。在一个示例中，FPB BD矢量控制1寄存器可以具有08h的偏移。

[0103] 表3:FPB BD矢量控制1寄存器

[0104]

位的位置	寄存器描述
0	**FPB BD 矢量使能**-当被置位时，使能 FPB BD 矢量机制 如果 FPB BD 矢量支持位为清除，则允许硬件将该位实现为只读（RO），并且在该情况下，该字段中的值是未定义的。该位的默认值为 0b。
3：1	保留
6：4	**FPB BD 矢量粒度**-由软件写入该字段的值控制 FPB BD 矢量的粒度和 FPB BD 矢量开始字段（下面）的要求对齐。定义的编码为： 值 粒度 开始对齐 000b 8BDF <无约束> 001b 16 BDF ...0b 010b 32 BDF ...00b 011b 64 BDF ...000b 100b 128 BDF ...0000b 101b 256 BDF ...00000b 所有其它编码被保留

[0105]

	基于实现的 FPB BD 矢量大小，允许硬件仅对该字段中的能够编程为非零值的那些位实现为 RW，在该情况下，允许高阶位，但是不要求硬连线为 0。 如果 FPB BD 矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值为 000b。
18：7	保留
31：19	FPB BD 矢量开始-由软件写入该字段的值控制 FPB BD 矢量所应用的 BD 空间中的偏移。 该值表示总线/设备编号（BDF 空间中的地址的位[15:3]），使得FPB BD矢量的位0表示从该寄存器中的值开始直至该值加上粒度减去 1 的范围，并且位 1 表示从该寄存器值加上粒度直至该值加上粒度减去 1 的范围，等等。功能编号偏移（位[2:0]由硬件固定为 000b 并且无法修改。 软件必须将该字段编程为根据如此处所指示的 FPB BD 矢量粒度字段中的值自然对齐的值： FPB BD 矢量粒度 开始对齐约束 0000b <无约束> 0001b ...0b 0010b ...00b 0011b ...000b 0100b ...0000b 0101b ...00000b 如果违反该要求，则硬件行为是未定义的。 如果 FPB BD 矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值为 000h。

[0106] 表4示出了FPB BD矢量控制2寄存器的一个示例实现方式。在一个示例中，FPB BD矢量控制2寄存器可以具有0Ch的偏移。

[0107] 表4：FPB BD矢量控制2寄存器

[0108]

位的位置	寄存器描述
2：0	保留
15：3	**BD 辅助开始**-由软件写入该字段的值控制通过桥向下游传递的类型 1 配置请求必须转换成类型 0 的 BDF 空间中的偏移。该值表示总线/设备编号（BDF 空间中的地址的位[15:3]。功能编号偏移位[2:0]由硬件固定为 000b 并且无法修改。 当设备控制 2 寄存器中的 ARI 转发使能位被置位时，则软件必须将该字段的位 7:3 写为 00000b。 如果 FPB BD 矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值为 000h。
31：16	保留

[0109] 表5示出了FPB BD矢量访问控制寄存器的一个示例实现方式。在一个示例中，FPB BD矢量访问控制寄存器可以具有10h的偏移。

[0110] 表5：FPB BD矢量访问控制寄存器

[0111]

位的位置	寄存器描述
7：0	**FPB BD 矢量访问偏移**-该字段中的值指示可通过 FPB BD 矢量访问数据寄存器读取或写入的FPB BD矢量的32b部分的偏移。该字段的位根据如此处所示的 FPB BD 矢量大小支持字段中的值映射到偏移： 偏移 位 该字段 000b 2:0 2:0(7:3 未使用) 001b 3:0 3:0(7:4 未使用) 010b 4:0 4:0(7:5 未使用) 011b 5:0 5:0(7:6 未使用)

[0112]

	100b 6:0 6:0(7 未使用) 101b 7:0 7:0 所有其它编码保留 该字段中根据上表未使用的位必须由软件写为 0b，并且允许不要求实现为 RO。 如果 FPB BD 矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值是 00h
31：8	保留

[0113] 表6示出了FPB BD矢量访问数据寄存器的一个示例实现方式。在一个示例中，FPB BD矢量访问数据寄存器可以具有14h的偏移。

[0114] 表6:FPB BD矢量访问数据寄存器

[0115]

位的位置	寄存器描述
31：0	**FPB BD 矢量数据**-来自该寄存器的读取返回在由 FPB BD 矢量访问偏移寄存器中的值所确定的位置处来自 FPB BD 矢量的数据的 DW。对该寄存器的写入取代了在由 FPB BD 矢量访问偏移寄存器中的值所确定的位置处来自 FPB BD 矢量的数据的 DW。如果 FPB BD 矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值是 0000h

[0116] 表7示出了FPB MEM低矢量控制寄存器的一个示例实现方式。在一个示例中，FPB MEM低矢量控制寄存器可以具有18h的偏移。

[0117] 表7:FPB MEM低矢量控制寄存器

[0118]

位的位置	寄存器描述
0	FPB MEM 低矢量使能-当置位时，使能 FPB MEM 低矢量机制。 如果 FPB MEM 低矢量支持位为清除，则允许硬件将该字段实现为 RO，并且在该情况下，该字段中的值是未定义的。

[0119]

	该位的默认值是 0b。
3：1	保留
7：4	**FPB MEM 低矢量粒度**-由软件写入该字段的值控制 FPB MEM 低矢量的粒度，以及 FPB MEM 低矢量开始字段的要求的对齐（下面）。 定义的编码为： 值 粒度 开始对齐约束 000b 1MB <无约束> 001b 2MB ...0b 010b 4MB ...00b 011b 8MB ...000b 100b 16MB ...0000b 所有其它编码保留 基于实现的 FPB EMM 低矢量大小，允许硬件仅针对该字段中的能够编程为非零值的位实现为 RW，在该情况下允许高阶位，但不要求硬连线为 0。 如果 FPB MEM 低矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值为 0000b。
19：8	保留
31：20	**FPB MEM 低矢量开始**-由软件写入该字段的值设置 FPB MEM 低矢量所应用的基址。 软件必须将该字段编程为根据如该字段（上面）的描述中所指示的 FPB MEM 低矢量粒度字段中的值自然对齐的值。如果该要求违反，则硬件行为是未定义的。 如果 FPB MEM 低矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。

[0120]

	该字段的默认值是 0000h。

[0121] 表8示出了FPB MEM低矢量访问控制寄存器的一个示例实现方式。在一个示例中，FPB MEM低矢量访问控制寄存器可以具有1Ch的偏移。

[0122] 表8:FPB MEM低矢量访问控制寄存器

[0123]

位的位置	寄存器描述
6：0	**FPB MEM 低矢量访问偏移**-该字段中的值指示能够通过 FPB MEM 低矢量访问数据寄存器读取或写入的 FPB MEM 低矢量的 32b 部分的偏移。 该字段的位根据如此处所示的 FPB MEM 低矢量粒度字段中的值映射到偏移： 偏移 位 该字段 000b 2:0 2:0(6:3 未使用) 001b 3:0 3:0(6:4 未使用) 010b 4:0 4:0(6:5 未使用) 011b 5:0 5:0(6 未使用) 100b 6:0 6:0 该字段中的根据上面的表未使用的位必须由软件写为 0b，并且允许不要求实现为 RO。 如果 FPB MEM 低矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值为 00h
31：7	保留

[0124] 表9示出了FPB MEM低矢量访问数据寄存器的一个示例实现方式。在一个示例中，FPB MEM低矢量访问数据寄存器可以具有20h的偏移。

[0125] 表9:FPB MEM低矢量访问数据寄存器

[0126]

位的位置	寄存器描述
31：0	**FPB MEM 低矢量数据**-来自该寄存器的读取返回在由 FPB MEM 低矢量访问偏移寄存器中的值所确定的位置处来自 FPB MEM 低矢量的数据的 DW。对该寄存器的写入取代了在由 FPB

[0127]

	MEM 低矢量访问偏移寄存器中的值所确定的位置处来自 FPB MEM 低矢量的数据的 DW。 如果 FPB MEM 低矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值是 0000h

[0128] 表10示出了FPB MEM高矢量控制1寄存器的一个示例实现方式。在一个示例中，FPB MEM高矢量控制1寄存器可以具有24h的偏移。

[0129] 表10:FPB MEM高矢量控制1寄存器

位的位置	寄存器描述
0	**FPB MEM 高矢量使能**-当置位时，使能 FPB MEM 高矢量机制。如果 FPB MEM 高矢量支持位为清除，则允许硬件将该字段实现为 RO，并且在该情况下，该字段中的值是未定义的。 该位的默认值为 0b。
3：1	保留
7：4	**FPB MEM 高矢量粒度**-由软件写入该字段的值控制 FPB MEM 高矢量的粒度，以及 FPB MEM 高矢量开始较低字段的要求的对齐（下面）。 允许软件从下面的表中选择任何允许的粒度，无论 FPB MEM 高矢量大小支持字段中的值如何。 定义的编码为：

[0130]

值	粒度	开始对齐约束
000b	256MB	<无约束>
001b	512MB	...0b
010b	1GB	...00b
011b	2GB	...000b
100b	4GB	...0000b
101b	8GB	...00000b
110b	16GB	...000000b
111b	32GB	...0000000b

[0131]

	基于实现的 FPB EMM 高矢量大小，允许硬件仅针对该字段中的能够编程为非零值的位实现为 RW，在该情况下允许高阶位，但不要求硬连线为 0。 如果 FPB MEM 高矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值为 0000b。
27：8	保留
31：28	**FPB MEM 高矢量开始较低**-由软件写入该字段的值将 FPB MEM 高矢量所应用的基址的较低位置位。软件必须将该字段编程为根据如此处所指示的 FPB MEM 高矢量粒度字段中的值自然对齐的值（即，低阶位为 0）： FPB MEM 高矢量粒度 约束 000b <无约束> 0001b ...0b 0010b ...00b 0011b ...000b 0100b ...0000b 0101b ...00000b 0110b ...000000b 0111b ...0000000b 如果违反该要求，则硬件行为是未定义的。 如果 FPB MEM 高矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值是 00h。

[0132] 表11示出了FPB MEM高矢量控制2寄存器的一个示例实现方式。在一个示例中，FPB MEM高矢量控制2寄存器可以具有28h的偏移。

[0133] 表11：FPB MEM高矢量控制2寄存器

[0134]

位的位置	寄存器描述
31：0	**FPB MEM 高矢量开始较高**-由软件写入该字段的值指示 FPB MEM 高矢量所应用的基址的位 62:32。如果 FPB MEM 高矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值为 00000000h。

[0135] 表12示出了FPB MEM高矢量访问控制寄存器的一个示例实现方式。在一个示例中，FPB MEM高矢量访问控制寄存器可以具有2Ch的偏移。

[0136] 表12:FPB MEM高矢量访问控制寄存器

[0137]

位的位置	寄存器描述
7：0	**FPB MEM 高矢量访问偏移**-该字段中的值指示可以通过 FPB MEM 高矢量访问数据寄存器读取或写入的 FPB BD、MEM 低或 MEM 高矢量的 32b 部分的偏移。 该字段的位根据如此处所示的 FPB MEM 高矢量粒度字段中的值映射到偏移： 偏移 位 该字段 000b 2:0 2:0(7:3 未使用) 001b 3:0 3:0(7:4 未使用) 010b 4:0 4:0(7:5 未使用) 011b 5:0 5:0(7:6 未使用) 100b 6:0 6:0(7 未使用) 101b 7:0 7:0 该字段中的根据上面的表未使用的位必须由软件写为 0b，并且允许不要求实现为 RO。 如果 FPB MEM 高矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值为 00h
13：8	保留

[0138]

15：14	**FPB矢量选择**-写入该字段的值选择要在所指示的FPB矢量访问偏移处访问的矢量，编码为： 00：BD 01：MEM低 10：MEM高 11：保留 该字段的默认值可以为00b
31：16	保留

[0139] 表13示出了FPB MEM高矢量访问数据寄存器的一个示例实现方式。在一个示例中，FPB MEM高矢量访问数据寄存器可以具有30h的偏移。

[0140] 表13:FPB MEM高矢量访问数据寄存器

[0141]

位的位置	寄存器描述
31：0	**FPB MEM 高矢量数据**-来自该寄存器的读取返回在由 FPB MEM 高矢量访问偏移寄存器中的值所确定的位置处来自 FPB MEM 高矢量的数据的 DW。对该寄存器的写入取代了在由 FPB MEM 高矢量访问偏移寄存器中的值所确定的位置处来自 FPB MEM 高矢量的数据的 DW。 如果 FPB MEM 高矢量支持位为清除，则允许硬件将该字段实现为 RO，并且该字段中的值是未定义的。 该字段的默认值是 0000h

[0142] 在替代实现方式中,不是为每个矢量提供单独的矢量访问偏移和矢量数据寄存器,而是可以使用单个矢量访问偏移寄存器,其中添加字段以指示要访问哪个矢量,以及单个矢量矢量数据寄存器可用于对指示的矢量执行读或写操作。在这样的实现方式中,指示符字段可以被实现为两位字段,两位字段被编码使得值00(二进制)可以指示对BD矢量的访问,值01(二进制)可以指示对MEM低矢量的访问,值10(二进制)可以指示对MEM高矢量的访问,而值11(二进制)可以指示保留值。

[0143] 注意,上述装置、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。作为具体说明,下图提供了用于利用如本文所述的本发明的示例性系统。由于更详细地描述了以下系统,因此从上面的讨论中公开、描述和重新考虑了许多不同的互连。并且显而易见的是,上述进步可以应用于那些互连、结构或架构中的任何一种。

[0144] 参考图11,描绘了包括多核处理器的计算系统的框图的实施例。处理器1100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、协处理器、片上系统(SOC)或执行代码的其他设备。在一个实施例中,处理器1100包括至少两个核-核1101和1102,其可包括不对称核或对称核(所示实施例)。然而,处理器1100可以包括可以是对称或不对称的任何数量的处理元件。

[0145] 在一个实施例中，处理元件是指用于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、进程单元、上下文、上下文单元、逻辑处理器、硬件线程、核、和/或能够保持处理器的状态(诸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，处理元件是指任何能够独立地与代码(诸如软件线程、操作系统、应用程序或其它代码)相关联的硬件。物理处理器(或处理器插槽)通常是指集成电路，可能包括诸如核或硬件线程之类的任意数量的其它处理元件。

[0146] 核通常是指位于能够维护独立架构状态的集成电路上的逻辑，其中每个独立维护的架构状态与至少一些专用执行资源相关联。与核不同，硬件线程通常是指位于能够维护独立架构状态的集成电路上的任何逻辑，其中独立维护的架构状态共享对执行资源的访问。能够看出，当某些资源被共享而其它专用于架构状态时，硬件线程和核的命名法之间的线重叠。更通常地，核和硬件线程被操作系统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。

[0147] 如图11所示，物理处理器1100包括两个核-核1101和1102。这里，核1101和1102被认为是对称核，即具有相同配置、功能单元和/或逻辑的核。在另一实施例中，核1101包括乱序处理器核，而核1102包括有序处理器核。然而，核1101和1102可以从任何类型的核中单独选择，例如本机核、软件管理核、适于执行本机指令集架构(ISA)的核、适于执行转换指令集架构(ISA)的核、共同设计的核或其他已知核。在异构核环境(即，非对称核)中，可以利用某种形式的转换(例如二进制转换)来在一个或两个核上调度或执行代码。然而，为了进一步讨论，核1101中示出的功能单元在下面进一步详细描述，因为核1102中的单元在所描绘的实施例中以类似的方式操作。

[0148] 如所描绘的，核1101包括两个硬件线程1101a和1101b，它们也可以称为硬件线程槽1101a和1101b。因此，在一个实施例中，诸如操作系统之类的软件实体可能将处理器1100视为四个单独的处理器，即，能够同时执行四个软件线程的四个逻辑处理器或处理元件。如上所述，第一线程与架构状态寄存器1101a相关联，第二线程与架构状态寄存器1101b相关联，第三线程可以与架构状态寄存器1102a相关联，并且第四线程可以与架构状态寄存器1102b相关联。这里，架构状态寄存器(1001a、1101b、1102a和1102b)中的每个可以被称为处理元件、线程槽或线程单元，如上所述。如所示出的，架构状态寄存器1101a在架构状态寄存器1101b中被复制，因此能够为逻辑处理器1101a和逻辑处理器1101b存储各个架构状态/上下文。在核1101中，还可以为线程1101a和1101b复制其他较小的资源，例如分配器和重命名器块1130中的指令指针和重命名逻辑。可以通过分区来共享一些资源，诸如重新排序/引退单元1135中的重新排序缓冲区、ILTB 1120、加载/存储缓冲区和队列。其他资源(例如，通用内部寄存器、页表基址寄存器、低级数据高速缓存和数据TLB 1115、执行单元1140以及乱序单元1135的部分)是可能完全共享的。

[0149] 处理器1100通常包括其他资源，其可以是完全共享的、通过分区共享的、或由处理元件/对处理元件专用的。在图11中，示出了具有处理器的说明性逻辑单元/资源的纯示例性处理器的实施例。注意，处理器可以包括或省略这些功能单元中的任何一个，以及包括未示出的任何其他已知功能单元、逻辑或固件。如图所示，核1101包括简化的、代表性的乱序(000)处理器核。但是可以在不同的实施例中使用有序处理器。000核包括用于预测要执行/采用的分支的分支目标缓冲器1120和用于存储用于指令的地址转换条目的指令转换缓冲

器(I-TLB)1120。

[0150] 核1101还包括解码模块1125,其耦合到获取单元1120以解码所获取的元件。在一个实施例中,获取逻辑包括分别与线程槽1101a、1101b相关联的各个定序器。通常,核1101与第一ISA相关联,第一ISA定义/指定在处理器1100上能执行的指令。通常作为第一ISA的一部分的机器代码指令包括指令的一部分(称为操作码),其引用/指定要执行的指令或操作。解码逻辑1125包括从其操作码识别这些指令并在管线中传递解码的指令以便进行由第一ISA定义的处理的电路。例如,如下面更详细讨论的,在一个实施例中,解码器1125包括被设计或适于识别特定指令(例如,事务指令)的逻辑。作为解码器1125识别的结果,架构或核1101采取特定的、预定义动作来执行与适当指令相关联的任务。重要的是要注意,可以响应于单个或多个指令来执行本文描述的任何任务、块、操作和方法;其中一些可能是新的或旧的指令。注意,在一个实施例中,解码器1126识别相同的ISA(或其子集)。可替代地,在异构核环境中,解码器1126识别第二ISA(第一ISA的子集或不同的ISA)。

[0151] 在一个示例中,分配器和重命名器块1130包括用于预留资源的分配器,诸如用于存储指令处理结果的寄存器文件。然而,线程1101a和1101b可能能够乱序执行,其中分配器和重命名器块1130还保留其他资源,例如,用于跟踪指令结果的重新排序缓冲器。单元1130还可以包括寄存器重命名器,用于将程序/指令参考寄存器重命名为处理器1100内部的其他寄存器。重新排序/引退单元1135包括组件,例如上面提到的重排序缓冲器、加载缓冲器和存储缓冲器,以支持乱序执行和稍后乱序执行的指令的有序引退。

[0152] 在一个实施例中,调度器和执行单元块1140包括调度器单元,用于调度执行单元上的指令/操作。例如,浮点指令被调度在具有可用浮点执行单元的执行单元的端口上。还包括与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、跳转执行单元、加载执行单元、存储执行单元和其他已知的执行单元。

[0153] 较低级数据高速缓存和数据转换缓冲器(D-TLB)1150耦合到执行单元1140。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,其可能保存在存储器一致性状态中。D-TLB用于存储最近的虚拟/线性到物理地址转换。作为特定示例,处理器可以包括页表结构,用于将物理存储器分成多个虚拟页面。

[0154] 这里,核1101和1102共享对更高级或更远级的高速缓存的访问,例如与片上接口1110相关联的第二级高速缓存。注意,更高级或更远级是指增加或从执行单位进一步远离的高速缓存级。在一个实施例中,较高级高速缓存是最后级数据高速缓存——处理器1100上的存储器层次结构中的最后高速缓存,诸如第二或第三级数据高速缓存。然而,较高级高速缓存不限于此,因为它可以与指令高速缓存相关联或包括指令高速缓存。跟踪高速缓存是一种类型的指令高速缓存,替代地可以耦合在解码器1125之后以存储最近解码的踪迹。这里,指令可能涉及宏指令(即,由解码器识别的一般指令),其可以解码为多个微指令(微操作)。

[0155] 在所描绘的配置中,处理器1100还包括片上接口模块1110。历史上,下面更详细描述的存储器控制器已被包括在处理器1100外部的计算系统中。在这种情况下,片上接口1110用于与处理器1100外部的设备进行通信,所述处理器1100外部的设备例如系统存储器1175、芯片组(通常包括用于连接到存储器1175的存储器控制器集线器和用于连接外围设

备的I/O控制器集线器)、存储器控制器集线器、北桥或其他集成电路。并且在这种情况下，总线1105可以包括任何已知的互连，例如多点总线、点对点互连、串行互连、并行总线，相干(例如高速缓存相干)总线、分层协议架构、差分总线和GTL总线。

[0156] 存储器1175可以专用于处理器1100或者与系统中的其他设备共享。存储器1175类型的常见示例包括DRAM、SRAM、非易失性存储器(NV存储器)和其他已知的存储设备。注意，设备1180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装置、无线收发器、闪存设备、音频控制器、网络控制器或其他已知的设备。

[0157] 然而，最近，随着更多逻辑和设备被集成在诸如SOC之类的单个管芯上，这些设备中的每一个可以合并在处理器1100上。例如，在一个实施例中，存储器控制器集线器与处理器1100在同一封装和/或管芯上。这里，核的一部分(核上部分)1110包括一个或多个控制器，用于与诸如存储器1175或图形设备1180之类的其他设备接合。包括互连和用于与这些设备接口的控制器的该配置通常被称为核上(或非核配置)。作为示例，片上接口1110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1105。然而，在SOC环境中，甚至更多的设备，例如网络接口、协处理器、存储器1175、图形处理器1180和任何其他已知的计算机设备/接口可以集成在单个管芯或集成电路上以提供具有高功能和低功耗的小形状因子。

[0158] 在一个实施例中，处理器1100能够执行编译器、优化和/或转换器代码1177以编译、转换和/或优化应用程序代码1176以支持本文描述的装置和方法或与其接合。编译器通常包括用于将源文本/代码转换成目标文本/代码的程序或程序集。通常，使用编译器编译程序/应用程序代码是在多个阶段和多遍中完成的，以将高级编程语言代码转换为低级机器或汇编语言代码。然而，单遍编译器仍可用于简单编译。编译器可以利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。

[0159] 较大的编译器通常包括多个阶段，但是大多数情况下这些阶段包括在两个一般阶段中：(1)前端，即通常可以进行句法处理、语义处理和一些变换/优化的地方，以及(2)后端，即通常进行分析、转换、优化和代码生成的地方。一些编译器引用了中间部分，它示出了编译器的前端和后端之间描绘的模糊。结果，对编译器的插入、关联、生成或其他操作的引用可以在任何上述阶段或遍以及编译器的任何其他已知阶段或遍中发生。作为一个说明性示例，编译器可能在编译的一个或多个阶段中插入操作、调用、起作用等，例如在编译的前端阶段插入调用/操作，然后在变换阶段将调用/操作变换为更低级别的代码。请注意，在动态编译期间，编译器代码或动态优化代码可以插入此类操作/调用，以及优化代码以便在运行时执行。作为特定说明性示例，可以在运行时期间动态地优化二进制代码(已编译的代码)。这里，程序代码可以包括动态优化代码、二进制代码或其组合。

[0160] 与编译器类似，转换器(例如二进制转换器)静态地或动态地转换代码以优化和/或转换代码。因此，对代码、应用程序代码、程序代码或其他软件环境的执行的引用可以指代：(1)动态地或静态地执行编译器程序、优化代码优化器或转换器以编译程序代码、维护软件结构、执行其他操作、优化代码、或转换代码；(2)执行包括操作/调用的主程序代码，例如已优化/编译的应用程序代码；(3)执行与主程序代码相关联的其他程序代码，例如库，以

维护软件结构、执行其他软件相关操作、或优化代码;或(4)其组合。

[0161] 现在参考图12,示出的是根据本发明的实施例的第二系统1200的框图。如图12所示,多处理器系统1200是点对点互连系统,并且包括经由点对点互连1250耦合的第一处理器1270和第二处理器1280。处理器1270和1280中的每一个可以是处理器的某个版本。在一个实施例中,1252和1254是串行、点对点相干互连结构的一部分,例如英特尔的快速路径互连(QPI)架构。结果,本发明可以在QPI架构内实现。

[0162] 虽然仅用两个处理器1270、1280示出,但应理解,本发明的范围不限于此。在其他实施例中,一个或多个附加处理器可以存在于给定处理器中。

[0163] 示出的处理器1270和1280分别包括集成存储器控制器单元1272和1282。处理器1270还包括作为其总线控制器单元的一部分的点对点(P-P)接口1276和1278;类似地,第二处理器1280包括P-P接口1286和1288。处理器1270、1280可以使用P-P接口电路1278、1288经由点对点(P-P)接口1250交换信息。如图12所示,IMC 1272和1282将处理器耦合到相应的存储器,即存储器1232和存储器1234,其可以是本地附接到相应处理器的主存储器的一部分。

[0164] 处理器1270、1280各自使用点对点接口电路1276、1294、1286、1298经由各个PP接口1252、1254与芯片组1290交换信息。芯片组1290还经由接口电路1292沿着高性能图形互连1239与高性能图形电路1238交换信息。

[0165] 共享高速缓存(未示出)可以包括在处理器中或两个处理器外部;然后通过P-P互连与处理器连接,使得如果处理器被置于低功率模式,则处理器的本地高速缓存信息中的任一个或两者可以存储在共享高速缓存中。

[0166] 芯片组1290可以经由接口1296耦合到第一总线1216。在一个实施例中,第一总线1216可以是外围组件互连(PCI)总线,或者诸如快速PCI总线或另一第三代I/O互连总线之类的总线。但本发明的范围不限于此。

[0167] 如图12所示,各种I/O设备1214耦合到第一总线1216,以及将第一总线1216耦合到第二总线1220的总线桥1218。在一个实施例中,第二总线1220包括低引脚数(LPC)总线。在一个实施例中,各种设备耦合到第二总线1220,包括例如键盘和/或鼠标1222、通信设备1227和存储单元1228,诸如磁盘驱动器或其他大容量存储设备,其通常包括指令/代码和数据1230。此外,音频I/O 1224被示出耦合到第二总线1220。注意,其他架构是可能的,其中所包括的组件和互连架构是变化的。例如,代替图12的点对点架构,系统可以实现多分支总线或其他这样的架构。

[0168] 尽管已经关于有限数量的实施例描述了本发明,但是本领域技术人员将意识到由此产生的许多修改和变化。所附权利要求旨在覆盖落入本发明的真实精神和范围中的所有这些修改和变化。

[0169] 实施例的方面可包括以下示例中的一个或组合:

[0170] 示例1是其上存储有指令的系统、方法、装置或存储介质,所述指令可执行以使机器识别系统中的多个设备并将相应的地址分配给多个设备中的每一个。多个设备中的每个设备在系统中通过多个总线中的至少相应一个总线连接,并且将地址分配给设备包括确定是根据第一寻址方案还是第二总线寻址方案来分配地址,其中所述第一寻址方案将总线/设备/功能(BDF)地址空间中的唯一总线编号分配给在所述第一寻址方案中寻址的每个设备,并且所述第二总线寻址方案分配所述BDF地址空间中的唯一的总线设备编号。

[0171] 示例2可以包括示例1的主题,其中重用特定总线编号以在所述第二寻址方案中对两个或更多个设备寻址。

[0172] 示例3可以包括示例1-2中的任一项的主题,其中分配地址包括在所述BDF地址空间中指定待用于根据所述第二寻址方案对设备进行寻址的总线编号的范围。

[0173] 示例4可以包括示例3的主题,其中所述总线编号的范围与特定交换机相关联,并且在所述总线设备编号中使用所述总线编号的范围中的总线编号以分配给连接到所述交换机的每个设备。

[0174] 示例5可以包括示例4的主题,其中连接到所述交换机的所述设备包括段。

[0175] 示例6可以包括示例1-5中任一个的主题,其中地址包括配置地址。

[0176] 示例7可以包括示例1-6中任一个的主题,其中所述BDF地址空间包括基于外围组件互连(PCI)的地址空间。

[0177] 示例8可以包括示例7的主题,其中每个总线设备编号包括八位总线编号和五位设备编号。

[0178] 示例9是一种装置,包括用于接收特定分组的端口,其中所述端口包括扁平化端口桥(FPB),所述FPB包括主要侧和辅助侧,所述主要侧连接到根据第一寻址方案寻址的第一组设备,并且所述辅助侧连接到根据第二寻址方案寻址的第二组设备。所述FPB还基于特定分组中的地址信息来确定是在所述主要侧还是在辅助侧对所述特定分组路由,对于所述第一组设备中的每个设备,所述第一寻址方案使用总线/设备/功能(BDF)地址中的唯一总线编号,而对于所述第二组设备中的每个设备,所述第二总线寻址方案使用唯一的总线设备编号。

[0179] 示例10可以包括示例9的主题,其中分配给所述第二组设备中的多个设备的相应总线设备编号中的每个均包括特定总线编号和不同设备编号。

[0180] 示例11可以包括示例9-10中任一个的主题,其中所述主要寻址方案包括传统寻址方案。

[0181] 示例12可以包括示例9-11中任一示例的主题,其中所述BDF寻址空间包括快速外围组件互连(PCIe)配置空间。

[0182] 示例13可以包括示例9-12中的任一项的主题,还包括多个端口,其中所述端口包括所述多个端口中的特定一个端口,并且所述多个端口中的至少一个其他端口包括FPB。

[0183] 示例14可以包括示例13的主题,其中多个端口包括至少一个不具有FPB的端口。

[0184] 示例15可以包括示例13的主题,还包括交换机,其中交换机包括多个端口。

[0185] 示例16可以包括示例13的主题,还包括根复合体,其中根复合体包括多个端口。

[0186] 示例17可以包括示例9-16中任一个的主题,还包括BD控制1寄存器。

[0187] 示例18可以包括示例9-17中任一个的主题,还包括BD矢量控制2寄存器。

[0188] 示例19可以包括示例9-18中任一个的主题,还包括BD矢量访问控制寄存器。

[0189] 示例20可以包括示例9-19中的任一项的主题,还包括BD矢量访问数据寄存器。

[0190] 示例21可以包括示例9-20中的任一项的主题,还包括MEM低矢量控制寄存器。

[0191] 示例22可以包括示例9-21中任一个的主题,还包括MEM低矢量访问控制寄存器。

[0192] 示例23可以包括示例9-22中任一个的主题,还包括MEM低矢量访问数据寄存器。

[0193] 示例24可以包括示例9-23中任一个的主题,还包括MEM高矢量控制1寄存器。

[0194] 示例25可以包括示例9-24中任一个的主题,还包括MEM高矢量控制2寄存器。

[0195] 示例26可以包括示例9-25中任一个的主题,还包括MEM高矢量访问控制寄存器。

[0196] 示例27可以包括示例9-26中任一个的主题,还包括MEM高矢量访问数据寄存器。

[0197] 示例28是至少一个其上存储有指令的机器可访问存储介质,所述指令当在机器上执行时,使所述机器用于:配置设备的寄存器以支持总线/设备/功能(BDF)空间中的主要总线寻址方案和替换总线寻址方案,所述替换总线寻址方案在枚举系统的多个不同总线的存储器映射输入/输出(I/O)(MMIO)空间内使用相同的总线编号。

[0198] 示例29可以包括示例28的主题,其中所述指令还能执行以约束要被分配给根端口桥的辅助侧的BD的可允许范围。

[0199] 示例30是一种系统,包括:交换机设备;连接到所述交换机设备的设备的层次结构;连接到所述交换机设备的一组一个或多个其他设备;其中根据第一寻址方案对一个或多个其他设备的集合进行寻址,根据第二寻址方案对设备的层次结构进行寻址,对于第一组设备中的每个设备所述第一寻址方案使用总线/设备/功能(BDF)地址空间中的唯一总线编号,而对于所述第二组设备中的每个设备,所述第二寻址方案使用唯一的总线设备编号。

[0200] 示例31可以包括示例30的主题,其中所述交换机设备包括连接到所述设备层次结构的第一端口,并且所述第一端口包括桥接逻辑,用于确定是利用所述第一寻址方案在桥的主要侧上对特定分组路由还是或利用所述第二寻址方案在桥的辅助侧上对特定分组路由。

[0201] 示例32可以包括示例30-31中任一个的主题,还包括:能力寄存器,其被编码以选择性地启用在所述交换机的特定端口上对所述辅助寻址方案的支持。

[0202] 示例33可以包括示例30-32中任一示例的主题,其中,所述交换机设备包括根复合体。

[0203] 设计可以经历从创建到模拟到制造的各个阶段。表示设计的数据可以以多种方式表示设计。首先,如在模拟中有用,可以使用硬件描述语言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。在使用传统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在用于制造集成电路的掩模不同掩模层上存在或不存在各种特征的数据。在设计的任何表示中,数据可以存储在任何形式的机器可读介质中。存储器或诸如盘的磁或光存储装置可以是机器可读介质,用于存储经由被调制或以其他方式生成以发送这样的信息的光波或电波发送的信息。当发送指示或携带代码或设计的电载波时,在执行电信号的复制、缓冲或重传的程度上,制作新的副本。因此,通信提供商或网络提供商可以将体现本公开的实施例的技术的诸如编码到载波中的信息之类的制品至少临时存储在有形的机器可读介质上。

[0204] 本文所使用的模块是指硬件、软件和/或固件的任意组合。作为示例,模块包括硬件,诸如微控制器,与用于存储适于由微控制器执行的代码的非暂时介质相关联。因此,在一个实施例中对模块的引用是指特别配置成识别和/或执行要保持在非暂时介质上的代码的硬件。此外,在另一实施例中,模块的使用是指包括特别适于由微控制器执行以实施预定操作的代码的非暂时介质。并且,如可推断的,在又一实施例中,术语模块(在该示例中)可以是指微控制器和非暂时介质的组合。通常,图示为单独的模块界限通常变化并且可能重

叠。例如，第一模块和第二模块可以共用硬件、软件、固件或其组合，同时可能保留一些独立的硬件、软件或固件。在一个实施例中，术语逻辑的使用包括硬件，诸如晶体管、寄存器或其它硬件，如可编程逻辑器件。

[0205] 在一个实施例中，短语“用于”或“被配置为”的使用是指布置、放在一起、制造、提供以售卖、进口和/或设计用于执行指定或确定的任务的装置、硬件、逻辑或元件。在该示例中，没有进行操作的装置或其元件仍‘被配置为’执行指定任务，前提是其被设计、耦合和/或互连来执行所述指定任务。纯粹作为说明性的示例，逻辑门可以在操作期间提供0或1。但是‘被配置为’向时钟提供使能信号的逻辑门不包括可提供1或0的每一个可能的逻辑门。相反，该逻辑门是以在操作期间1或0输出用于使能时钟的某种方式耦合的逻辑门。再次注意到，术语‘被配置为’的使用不要求操作，而是侧重于装置、硬件和/或元件的潜在状态，其中在潜在状态下，装置、硬件和/或元件设计成当装置、硬件和/或元件正在操作时执行特定任务。

[0206] 此外，在一个实施例中，短语“能够/能”和/或“能操作”的使用是指以使能按指定方式使用装置、逻辑、硬件和/或元件的这样的方式设计的某种装置、逻辑、硬件和/或元件。如上文所提到的，在一个实施例中，用于、能够或能操作的使用是指装置、逻辑、硬件和/或元件的潜在状态，其中装置、逻辑、硬件和/或元件没有正操作，但是以使得能够以指定方式使用装置的方式进行设计。

[0207] 如本文所使用的，值包括数字、状态、逻辑状态或二进制逻辑状态的任何已知表示。通常，逻辑电平、逻辑值或逻辑上的值还称为1和0，简单地表示二进制逻辑状态。例如，1表示高逻辑电平，而0表示低逻辑电平。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。但是，已经使用了计算机系统中的值的其他表示。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，值包括能够保存在计算机系统中的任何信息表示。

[0208] 此外，状态可由值或值的部分表示。作为示例，诸如逻辑1之类的第一值可以表示默认或初始状态，而诸如逻辑0之类的第二值可以表示非默认值。另外，在一个实施例中，术语复位和置位分别是指默认值或状态和更新值或状态。例如，默认值可能包括高逻辑值，即复位，而更新的值可能包括低逻辑值，即置位。注意，值的任何组合可用于表示任意数量的状态。

[0209] 上文所阐述的方法、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、能够由处理元件执行的指令或代码来实现。非暂时性机器可访问/可读介质包括提供(即，存储和/或发送)诸如计算机或电子系统之类的机器能够读取的形式的信息的任何机制。例如，非暂时性机器可访问介质包括随机存取存储器(RAM)，如静态RAM(SRAM)或动态RAM(DRAM)；ROM；磁或光存储介质；闪存设备；电存储设备；光存储设备；声存储设备；其它形式的用于保存从暂时(传播)信号(例如，载波、红外信号、数字信号)接收到的信息的存储设备；等等，以区别于可从其接收信息的非暂时性介质。

[0210] 用于对逻辑进行编程以执行本公开的实施例的指令可以存储在系统中的存储器内，如DRAM、高速缓存、闪速存储器或其它存储装置。此外，指令可以经由网络或通过其它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)可读形式存

储或发送信息的任何机制,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、闪速存储器或用于经由电、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)通过因特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存储或发送机器(例如,计算机)可读形式的电子指令或信息的任何类型的有形机器可读介质。

[0211] 在本说明书通篇提到“一个实施例”或“实施例”是指结合该实施例描述的特定的特征、结构或特性包括在本公开的至少一个实施例中。因此,在本说明书通篇各处出现的短语“在一个实施例中”或“在实施例中”不一定都指同一实施例。此外,在一个或多个实施例中,特定的特征、结构或特性可以任何适合的方式结合。

[0212] 在前面的说明书中,已经参考具体的示例性实施例进行了详细说明。然而,将显而易见的是,可以对其进行各种修改和改变,而不偏离如随附权利要求中阐述的本公开的更宽泛的精神和范围。因此,说明书和附图应在说明性的意义而非限制性的意义上考虑。此外,前面对实施例和其它示范性语言的使用不一定是指同一实施例或同一示例,但是可以指不同和独特的实施例以及可能同一实施例。

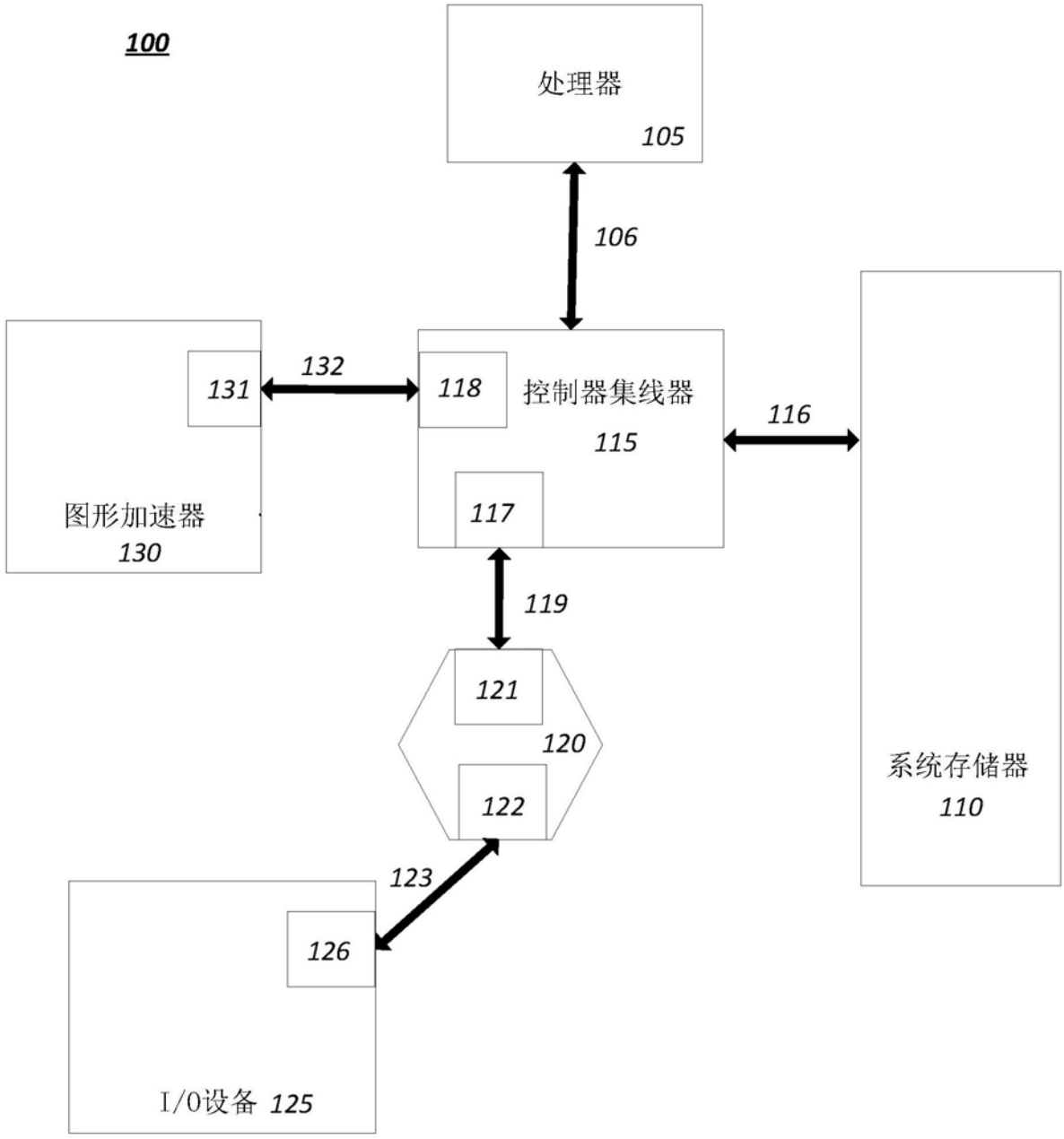

100
处理器
105
106
131
132
118
控制器集线器
115
116
图形加速器
130
117
119
121
120
122
系统存储器
110
123
126
I/O设备 125

图1

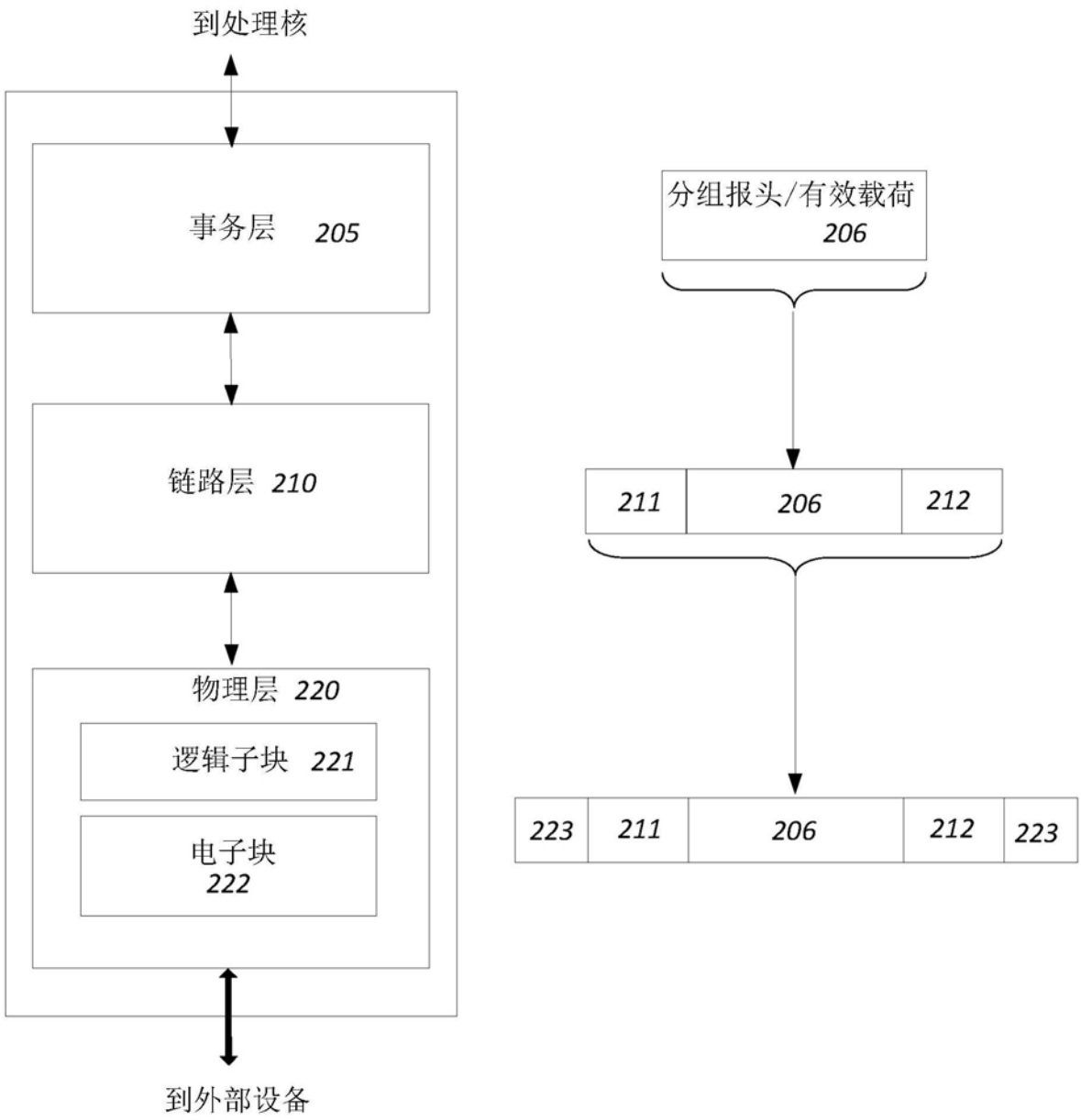

分层协议栈 200
到处理核
事务层 205
链路层 210
物理层 220
逻辑子块 221
电子块 222
到外部设备
分组报头/有效载荷 206
211
206
212
223
211
206
212
223

图2


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

图3

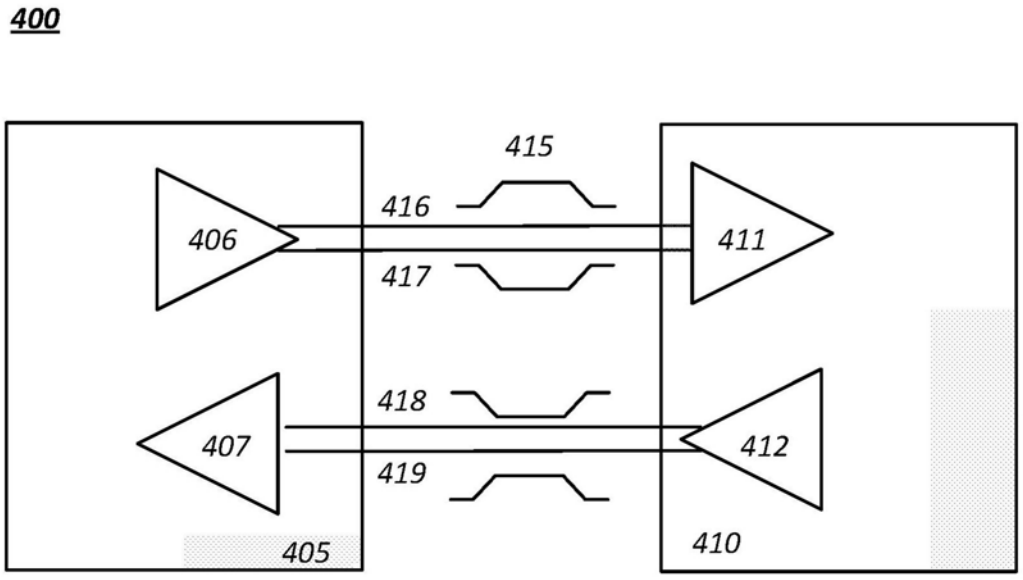

400
415
416
406
411
417
418
407
412
419
405
410

图4

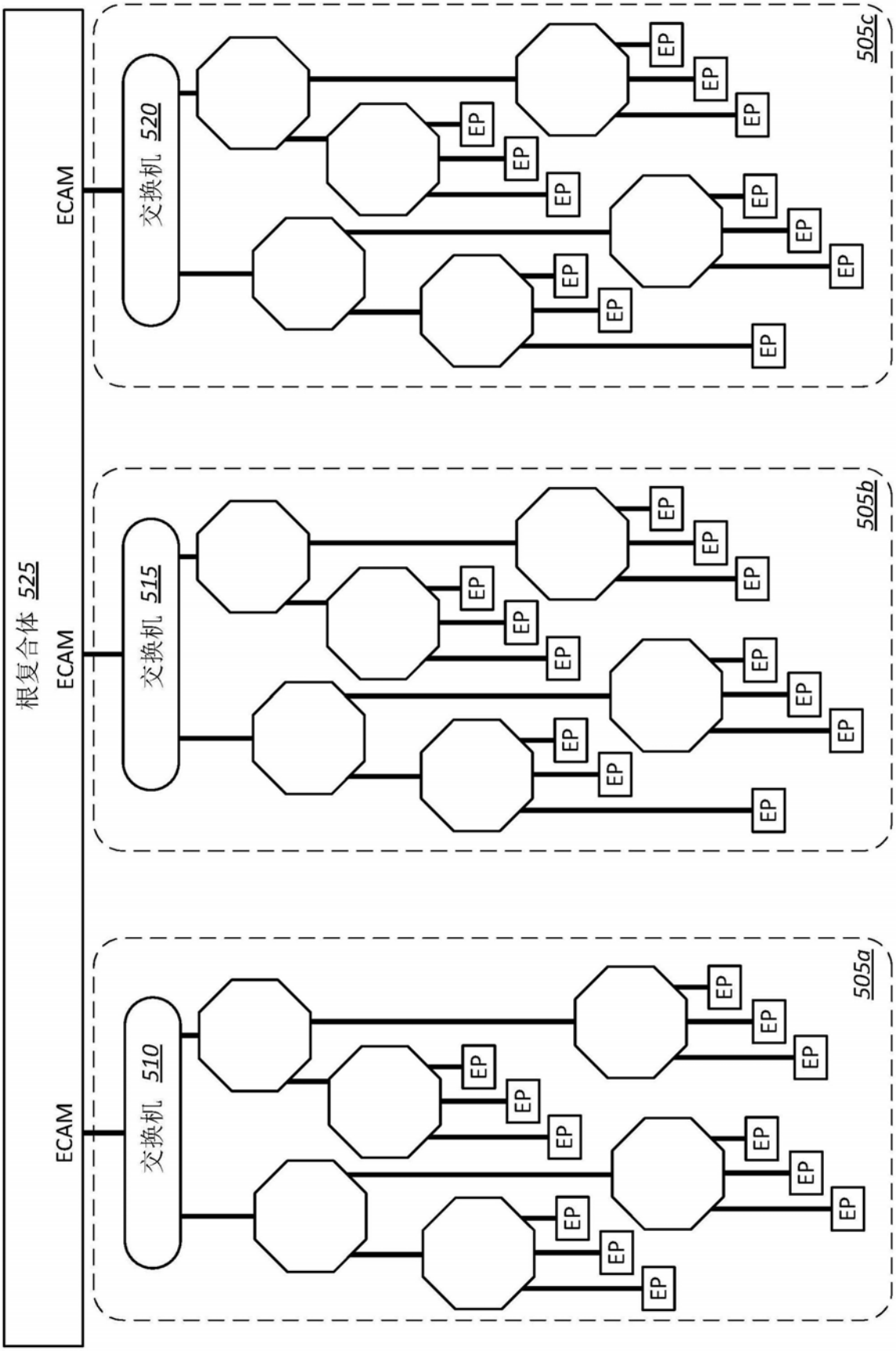

根复合体 525
ECAM
ECAM
ECAM
交换机 510
交换机 515
交换机 520
505a
505b
505c
EP

图5

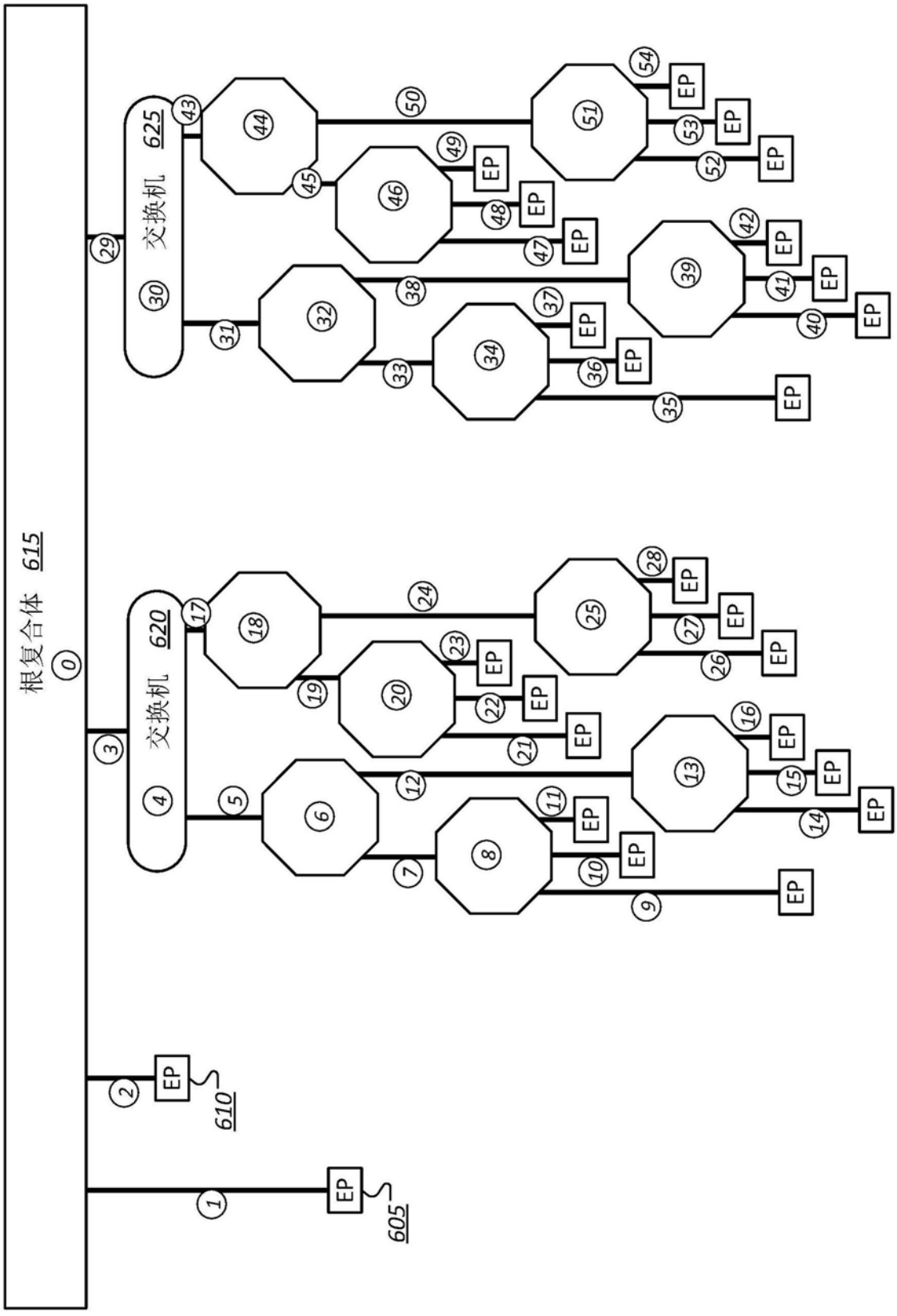


图6

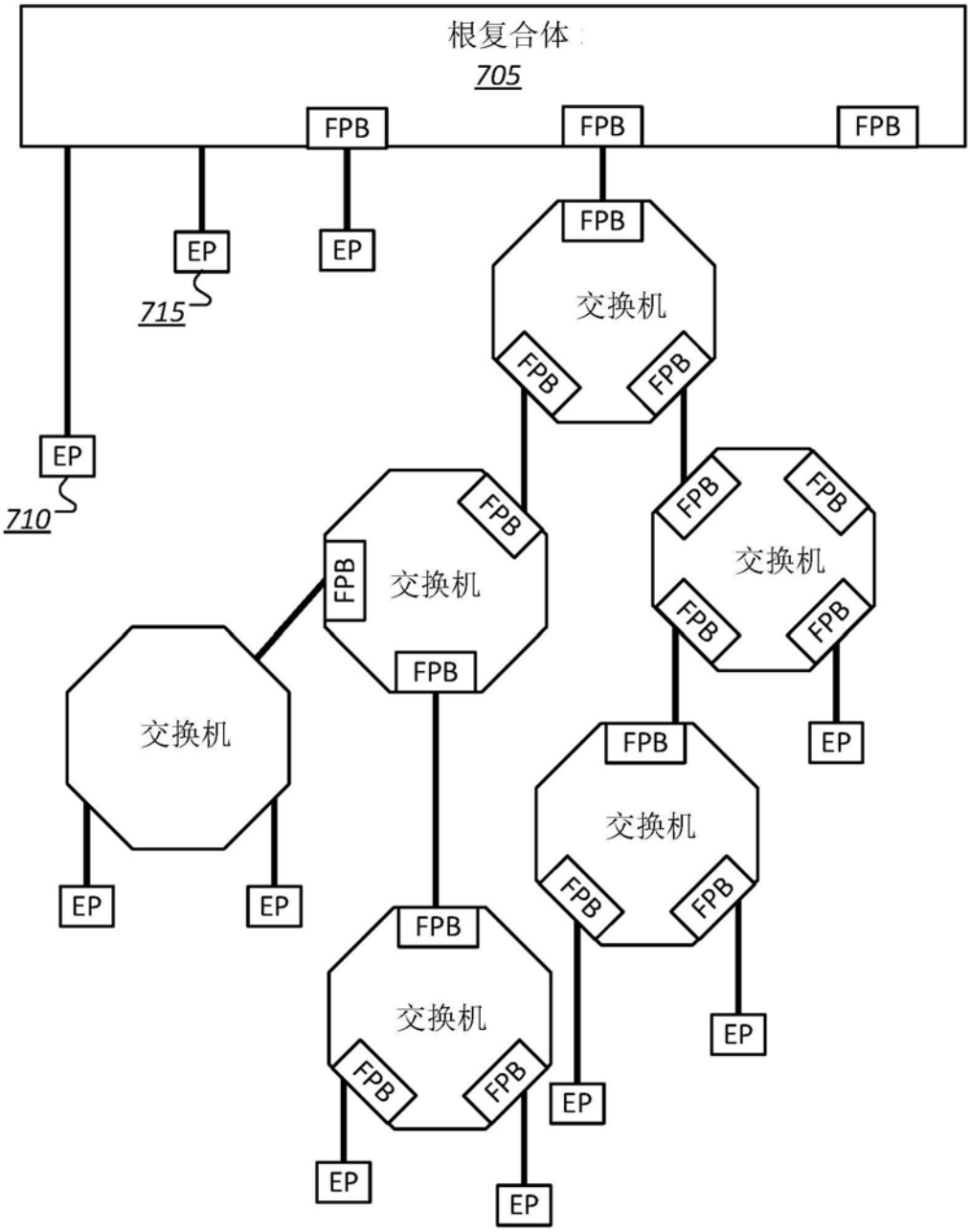

根复合体
705
FPB
FPB
FPB
EP
EP
715
EP
710
交换机
交换机
交换机
交换机
交换机
EP
EP
EP
EP
EP
EP
EP

图7A

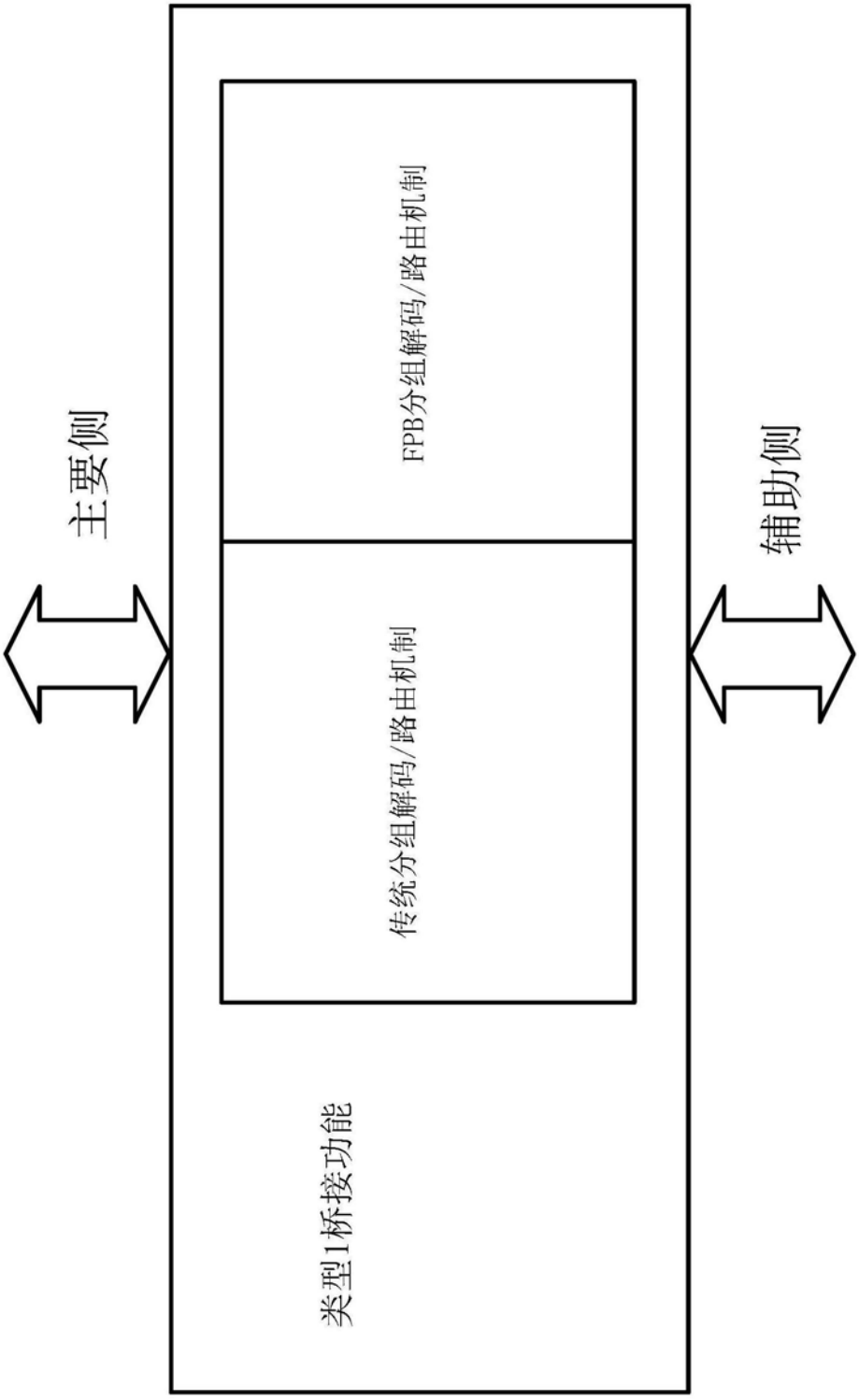


图7B

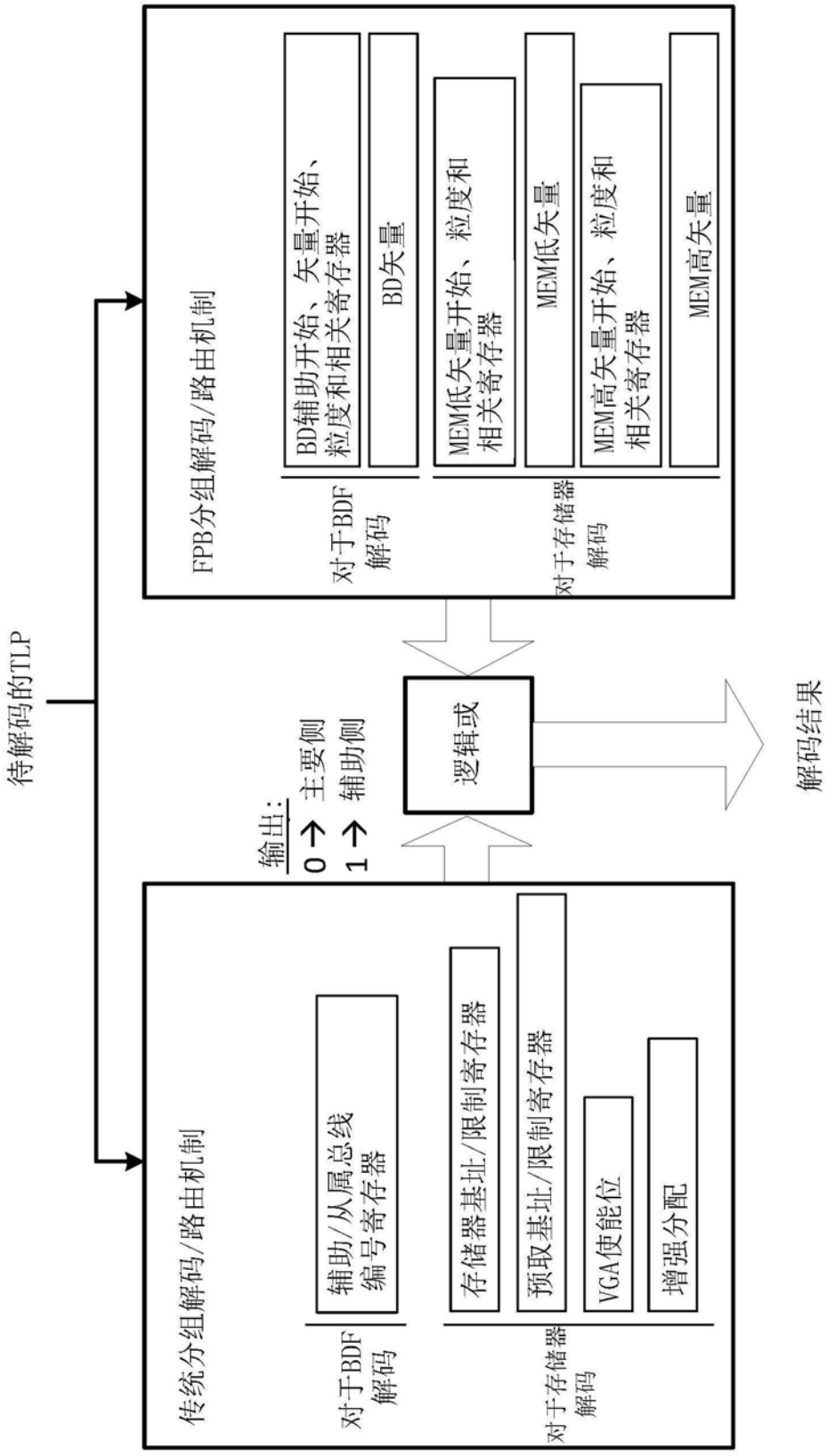

待解码的TLP
传统分组解码/路由机制
对于BDF解码
辅助/从属总线编号寄存器
对于存储器解码
存储器基址/限制寄存器
预取基址/限制寄存器
VGA使能位
增强分配
输出：
0→ 主要侧
1→ 辅助侧
逻辑或
解码结果
FPB分组解码/路由机制
对于BDF解码
BD辅助开始、矢量开始、粒度和相关寄存器
BD矢量
对于存储器解码
MEM低矢量开始、粒度和相关寄存器
MEM低矢量
MEM高矢量开始、粒度和相关寄存器
MEM高矢量

图8


15
8
7
3
2
0
总线编号
设备
Fn
总线/设备
8 BDF粒度
Fn
总线/设备
16 BDF粒度
Fn
总线/设备
32 BDF粒度
Fn
总线/设备
64 BDF粒度
Fn
总线/设备
128 BDF粒度
Fn
总线/设备
256 BDF粒度
Fn

图9

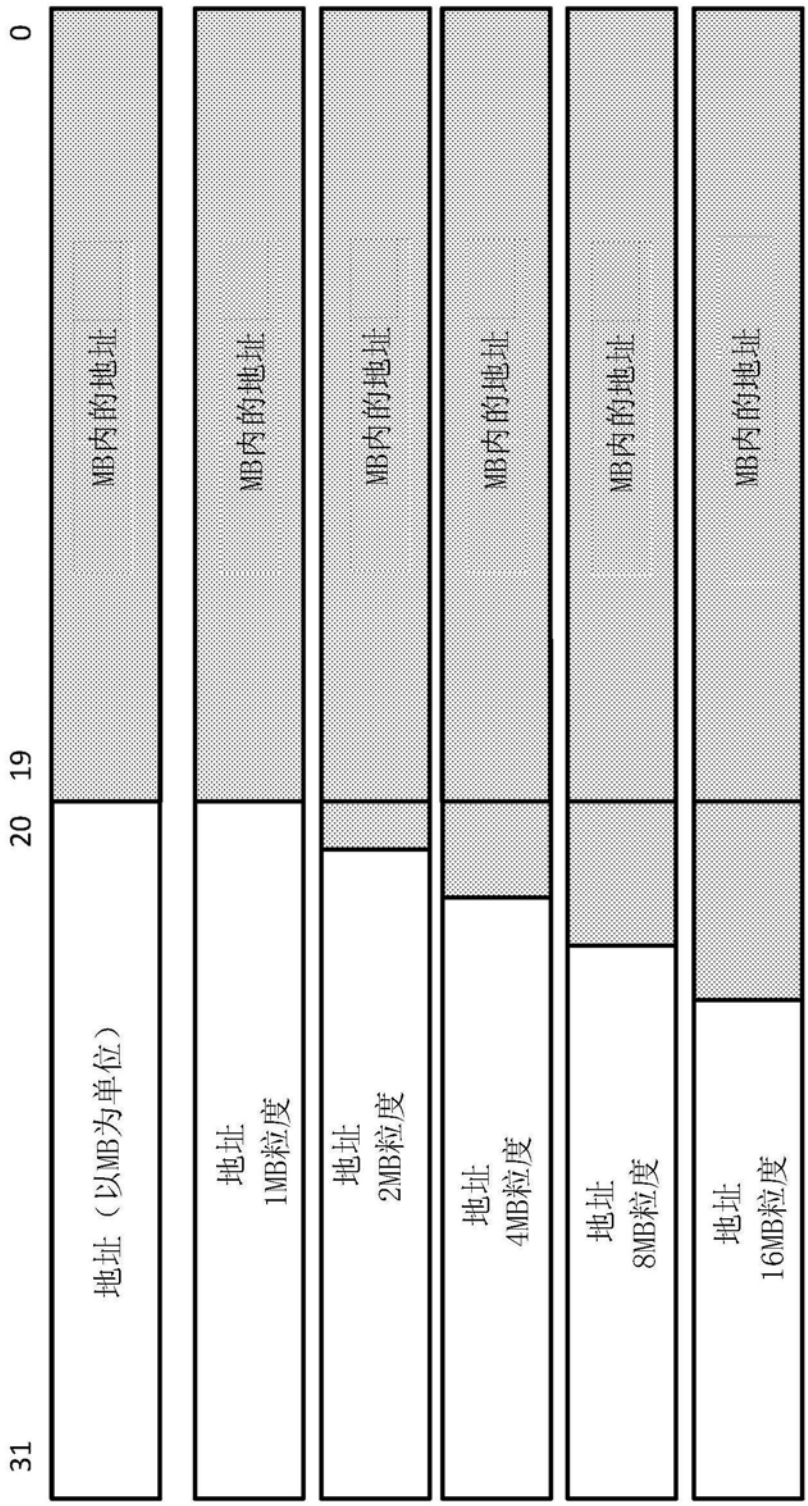

0
19
20
31
MB内的地址
地址（以MB为单位）
地址
1MB粒度
地址
2MB粒度
地址
4MB粒度
地址
8MB粒度
地址
16MB粒度

图10

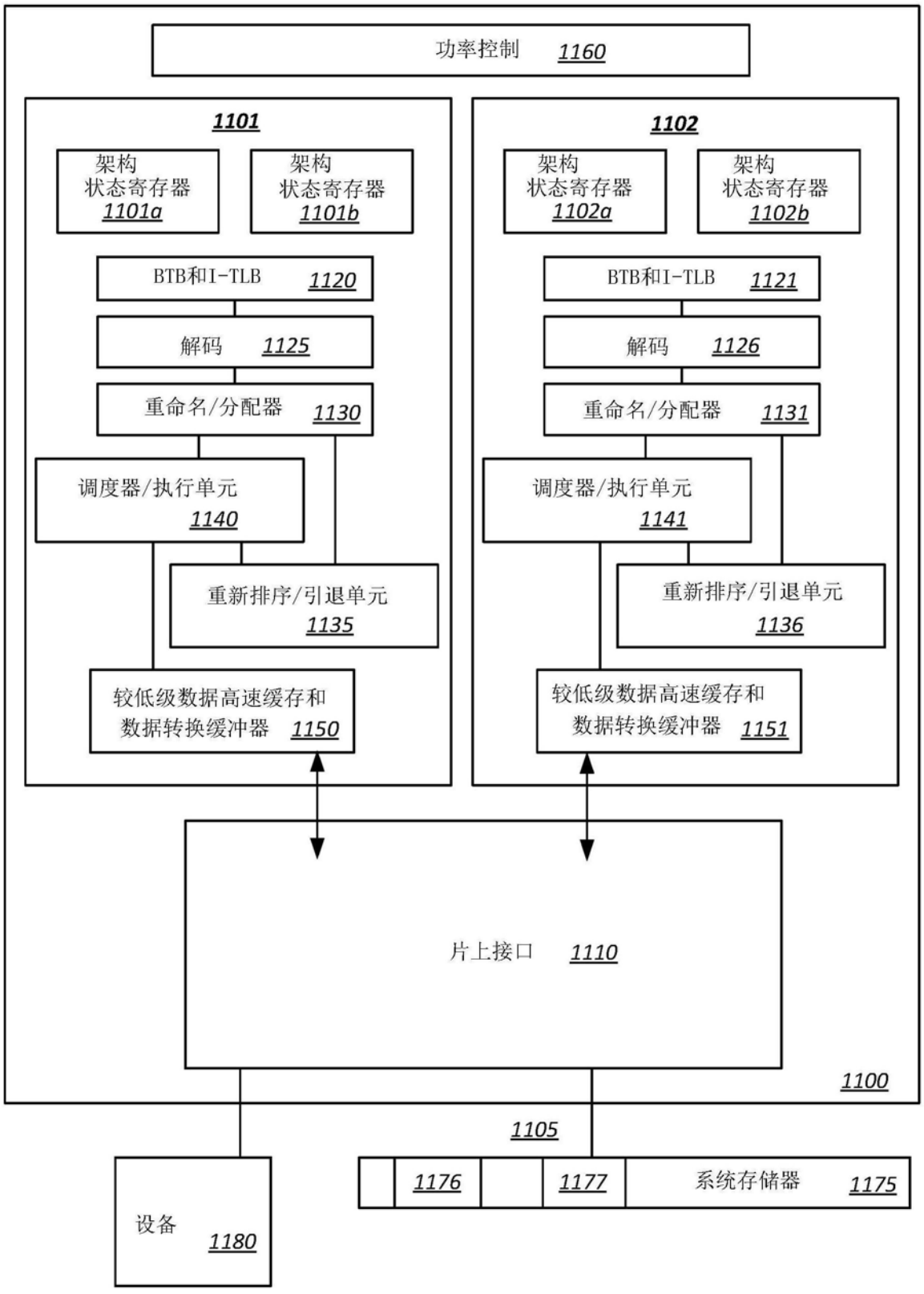

功率控制 1160
1101
架构状态寄存器 1101a
架构状态寄存器 1101b
BTB和I-TLB 1120
解码 1125
重命名/分配器 1130
调度器/执行单元 1140
重新排序/引退单元 1135
较低级数据高速缓存和数据转换缓冲器 1150
1102
架构状态寄存器 1102a
架构状态寄存器 1102b
BTB和I-TLB 1121
解码 1126
重命名/分配器 1131
调度器/执行单元 1141
重新排序/引退单元 1136
较低级数据高速缓存和数据转换缓冲器 1151
片上接口 1110
1100
1105
1176
1177
系统存储器 1175
设备 1180

图11

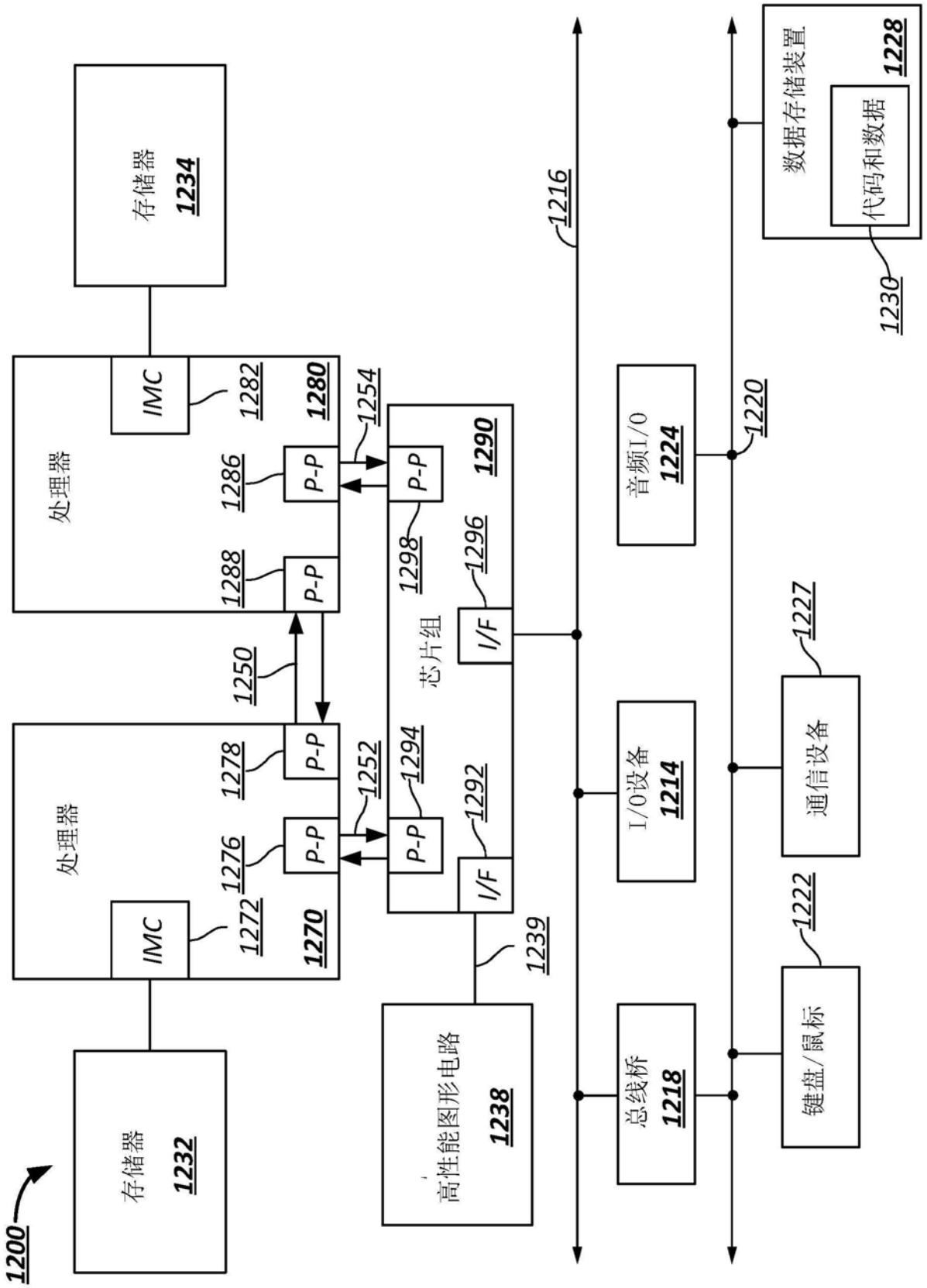

1200
存储器
1232
处理器
IMC
1272
1270
1276
P-P
1278
P-P
1252
1250
处理器
IMC
1282
1280
1286
P-P
1288
P-P
1254
芯片组
1290
P-P
1294
P-P
1298
1292
I/F
1296
I/F
1239
高性能图形电路
1238
存储器
1234
1216
总线桥
1218
I/O设备
1214
音频I/O
1224
1220
通信设备
1227
键盘/鼠标
1222
数据存储装置
代码和数据
1228
1230

图12