



## [12] 发明专利申请公开说明书

[21] 申请号 02816580.2

[43] 公开日 2004 年 11 月 17 日

[11] 公开号 CN 1547704A

[22] 申请日 2002.8.23 [21] 申请号 02816580.2

[30] 优先权

[32] 2001.8.24 [33] US [31] 60/314,708

[86] 国际申请 PCT/US2002/027042 2002.8.23

[87] 国际公布 WO2003/019394 英 2003.3.6

[85] 进入国家阶段日期 2004.2.24

[71] 申请人 英特尔公司

地址 美国加利福尼亚州

[72] 发明人 贾斯明·阿亚诺维奇

戴维·J·哈里曼

伦道夫·L·坎贝尔

乔斯·A·瓦尔加斯

克利福德·霍尔 普拉沙安特·塞西

史蒂夫·帕夫洛夫斯基

[74] 专利代理机构 北京东方亿思专利代理有限公司

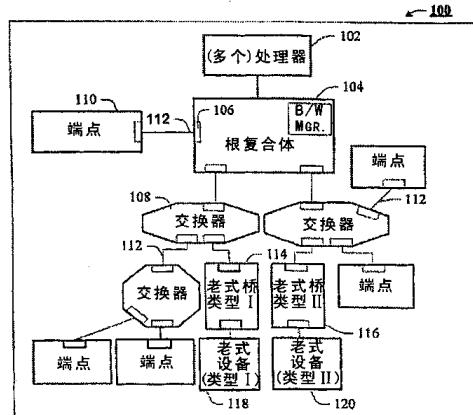
代理人 王 怡

权利要求书 4 页 说明书 51 页 附图 6 页

[54] 发明名称 支持老式中断的通用输入/输出体系结构、协议和方法

[57] 摘要

本发明公开了一种增强型通用输入/输出通信体系结构、协议以及相关的方法。



1. 一种将老式设备集成在增强型通用输入/输出体系结构中的方法，所述方法包括：

5 从位于通用输入/输出接口的老式设备接收指示；

分析所述所接收指示的至少一个子集以识别指示符类型；以及

基于或至少部分基于所述分析的结果，将所述所接收的老式指示转换为适合的一个或多个通用输入/输出消息。

2. 如权利要求 1 所述的方法，还包括：

10 根据与所述通用输入/输出消息相关联的一组规则处理所述通用输入/输出消息。

3. 如权利要求 2 所述的方法，所述处理步骤包括：

识别所述通用输入/输出接口的主机是否是所述通用输入/输出消息的目标；以及

15 如果所述所识别的目标不是所述主机，则将所述通用输入/输出消息通过通用输入/输出链路转发到所述所识别的目标。

4. 如权利要求 3 所述的方法，所述转发步骤包括将所述通用输入/输出消息写入远程通用输入/输出接口的消息空间，所述远程通用输入/输出接口通过通用输入/输出接口链路可通信地与所述通用输入/输出接口相耦合。

20 5. 如权利要求 1 所述的方法，其中所述指示是老式中断。

6. 如权利要求 5 所述的方法，其中所述指示是外围组件互连中断。

7. 如权利要求 1 所述的方法，其中所述指示是电源管理请求。

8. 如权利要求 7 所述的方法，其中所述电源管理请求是热插拔通知。

25 9. 如权利要求 1 所述的方法，其中所述所接收的指示是特殊周期请求。

10. 如权利要求 1 所述的方法，所述分析所述所接收指示的至少一个子集的步骤包括：

将所述所接收的指示与由所述通用输入/输出接口维护的一个或多个已

知老式指示相比较以分析所述所接收的指示的含义。

11. 如权利要求 1 所述的方法，所述转换步骤包括：

识别与通用输入/输出协议相关联的代码，所述通用输入/输出协议由传递所述所接收指示的含义的所述通用输入/输出体系结构使用；以及

5 合成包括所述代码的消息以发送到所述所接收指示的所识别目标。

12. 如权利要求 11 所述的方法，其中所述所接收指示的所述所识别目标是所述通用输入/输出接口的主机。

13. 如权利要求 11 所述的方法，其中所述所接收指示的所述所识别目标是通过一个或多个通用输入/输出链路而被耦合的远程设备。

10 14. 如权利要求 11 所述的方法，其中所述所接收指示被识别为外围组件互连中断，所述合成步骤包括：

生成指明所述所识别中断的断言的消息，用于如所需地通过所述通用输入/输出体系结构向所述中断的目标发送。

15 15. 如权利要求 12 所述的方法，还包括：

在接收到指明终止所述中断的指示之后，生成指明所述所识别中断的反断言的消息，用于如所需地通过所述通用输入/输出体系结构向所述中断的目标发送。

16. 如权利要求 15 所述的方法，其中所述终止所述中断是指明终止所述中断的指示。

20 17. 一种通用输入/输出接口，包括：

物理层接口，通过通用输入/输出链路将所述通用输入/输出接口耦合到远程接口；和

事务层接口，接收从所述远程接口接收的包括老式内容的内容的至少一个子集，其中所述事务层基于或至少部分基于适合由主设备或远程设备进行附加处理的所述老式内容生成通用输入/输出消息。

25 18. 如权利要求 17 所述的通用输入/输出接口，其中所述远程接口包括与老式设备或增强型通用输入/输出设备相关联的老式接口。

19. 如权利要求 17 所述的通用输入/输出接口，所述事务层接口包括：

包括消息空间的数据结构，其中通用输入/输出接口的所述事务层将所

述所生成的通用输入/输出消息写入远程通用输入/输出接口。

20. 如权利要求 19 所述的通用输入/输出接口，其中所述通用输入/输出消息引起所述远程通用输入/输出接口向目标设备转发所述通用输入/输出消息。

5 21. 如权利要求 20 所述的通用输入/输出接口，其中所述远程通用输入/输出接口是所述目标设备的元件。

22. 如权利要求 20 所述的通用输入/输出接口，其中所述通用输入/输出消息引起所述目标设备根据所述老式内容工作。

10 23. 如权利要求 19 所述的通用输入/输出接口，所述数据结构包括：

配置空间，用于维护信息以指明主设备类型；和  
消息空间，用于帮助在通用输入/输出体系结构中的通用输入/输出接

口之间带内传送表示老式内容的通用输入/输出消息。

24. 如权利要求 17 所述的通用输入/输出接口，其中所述老式接口与外

围组件互连兼容设备相关联。

15 25. 如权利要求 17 所述的通用输入/输出接口，其中所述老式内容包括  
中断、电源管理消息和特殊周期请求中的一个或多个。

26. 一种电子设备，包括如权利要求 17 所述的通用输入/输出接口。

27. 一种电子装置，包括：

20 多个如权利要求 26 所述的电子设备，有选择地与通用输入/输出通信  
链路相耦合；和

零个或多个老式设备，每个都有选择地耦合到所述多个电子设备的相  
关联的一个，其中所述相关联的电子设备基于或至少部分基于从相关联的  
老式设备接收的老式内容生成通用输入/输出消息。

25 28. 一种包括内容的存储介质，当所述内容由访问电子设备执行时，  
引起所述电子设备实现通用输入/输出接口，所述通用输入/输出接口包括  
物理层接口，用于通过通用输入/输出链路将所述通用输入/输出接口耦合  
到远程接口，和事务层接口，用于接收从所述远程接口接收的包括老式内  
容的内容的至少一个子集，其中所述事务层基于或至少部分基于适合由主  
设备或远程设备进行附加处理的所述老式内容来生成通用输入/输出消息。

29. 如权利要求 28 所述的存储介质，实现所述事务层的所述内容还包括将所接收的老式内容转换为上行流通用输入/输出接口可理解的通用输入/输出消息的内容。

30. 如权利要求 29 所述的存储介质，其中所述老式内容包括中断、电源管理消息和特殊周期请求的一个或多个。  
5

## 支持老式中断的通用输入/输出体系结构、协议和方法

### 5 优先权

本申请明确地要求了由 Ajanovic 等人于 2001 年 8 月 24 日递交的美国临时申请 No.60/314,708，名为“ A High-speed, Point-to-Point Interconnection and Communication Architecture, Protocol and Related Methods”（“高速、点到点互连和通信体系结构、协议以及相关方法”）的优先权，并且转让给本申请的受让人。

### 技术领域

本发明一般地涉及通用输入/输出（GIO）总线体系结构领域，更具体地说，本发明涉及一种体系结构、协议以及相关的方法，用来在 GIO 总线体系结构中的元件之间支持老式中断。

### 背景技术

计算装置例如计算机系统、服务器、网络交换机和路由器、无线通信设备以及其它电子设备一般由许多电子组件或元件组成。这些元件通常包括处理器、微控制器或其它控制逻辑、存储器系统、（多个）输入和输出接口、外围元件等。为了便于这些元件之间的通信，计算装置长期依赖于通用输入/输出（GIO）总线体系结构，以使得该计算装置的这些根本不同的元件能够互相通信来支持由这样的装置提供的种种应用。

这种传统的 GIO 总线体系结构最普遍的一种形式或许就是外围组件互连总线或 PCI 总线体系结构。PCI 总线标准（1998 年 12 月 18 日发布的外围组件互连（PCI）局域总线规范，修订版 2.2）规定了多接点式（multi-drop）、并行总线体系结构，用于在计算装置中以仲裁的方式来互连芯片、扩充板以及处理器/存储器子系统。为了本发明的目的，PCI 局域总线标准的内容在这里作为参考而被明确地引用。

传统的 PCI 总线实现具有 133 兆字节每秒的吞吐量（即，33 兆赫兹 32 字节），而 PCI 2.2 标准允许每个管脚 64 字节的并行连接，时钟达到 133MHz，从而产生超过 1GBps 的理论吞吐量。在这方面，由这样的传统多接点式 PCI 总线体系结构提供的吞吐量到目前为止已经提供了足够的带宽来适应即使是最先进的计算装置（例如，多处理器服务器应用、网络装置等）的内部通信需要。然而，联系到宽带因特网访问的广泛应用，处理能力的新近进展将处理速度超过了 1GHz 的阈值，诸如 PCI 总线体系结构的传统 GIO 体系结构已经变成这样的计算装置中的瓶颈。

通常与传统 GIO 体系结构联系在一起的另一个限制是，它们通常不能很好的适宜于操作/处理同步（或者说时间相关）数据流。这样的同步数据流的一个例子是多媒体数据流，该多媒体数据流需要同步传输机制来确保接收数据与使用数据同速，并且音频部分与视频部分同步。

传统的 GIO 体系结构异步处理数据，或以带宽允许的随机时间间隔处理数据。这种同步数据的异步处理可能导致音频与视频的不重合，作为结果，某些同步多媒体内容供应商制定了使某些数据优先于其它数据的规则，例如使音频数据优先于视频数据，从而最终用户至少接收相对稳定的音频流（即，不被打断），使得他们可以欣赏或了解正在被流式播放的歌曲、故事等等。

## 20 附图说明

本发明以示例而非限制的方式被说明，附图中类似的标号指示类似的元件，并且其中：

图 1 是电子装置的方框图，该电子装置包括本发明的实施例的一个或多个方面以便于该装置的一个或多个组成元件之间的通信；

25 图 2 是根据本发明一个示例性实施例的示例性通信栈的示图，该通信栈由电子装置的一个或多个元件使用以便于这些元件之间的通信；

图 3 是根据本发明教导的示例性事务层（transaction layer）数据报的示图；

图 4 是根据本发明一个方面的示例性通信链路的示图，该通信链路包

括一个或多个虚拟信道以便于电子设备的一个或多个元件之间的通信；

图 5 是根据本发明一个实施例，用于在 EGIO（增强型通用输入/输出）体系结构中提供同步通信资源的示例性方法的流程图；

图 6 是根据本发明的一个方面，用于在 EGIO 体系结构中实现流控制  
5 的示例性方法的流程图；

图 7 是根据本发明的一个方面，用于在 EGIO 体系结构中实现数据完整性特征的示例性方法的流程图；

图 8 是根据本发明的一个示例性实施例的示例性通信代理的方框图，  
以选择性实现本发明的一个或多个方面；

10 图 9 是本发明的事务层中使用的各种分组头部格式的方框图；

图 10 是根据本发明示例性实施例的示例性存储器体系结构的方框图，  
该存储器体系结构被用于帮助本发明的一个或多个方面；

图 11 是根据本发明一个方面的示例性链路状态机图的状态图；以及

图 12 是含有内容的可访问介质的方框图，所述内容当由电子设备访  
15 问时实现本发明的一个或多个方面。

### 具体实施方式

本发明的多个实施例一般地涉及通用输入/输出（GIO）体系结构、协议和相关方法，以实现支持通用输入/输出接口组件之间的老式中断。在这

20 方面，介绍了创新的增强型通用输入/输出（EGIO）互连体系结构、相关通信协议和有关方法。根据一个示例性实施例，EGIO 体系结构的元件包括根复合体（root complex）（例如，在桥内实现）、以及端点（end point）中的一个或多个，每个元件至少包含 EGIO 特征的一个子集以支持这些元件之间的 EGIO 通信。

25 使用（多条）串行通信信道来执行这些元件的 EGIO 设备之间的通信，所述串行通信信道使用 EGIO 通信协议，所述协议如下面将要详细介绍的那样支持一个或多个创新特征，所述特征包括但不限于虚拟通信信道、基于尾部（tailer）的错误转发（error forwarding）、对老式（legacy）的基于 PCI 的设备及其中断的支持、多种请求响应类型、流控制和/或数据

完整性管理功能。根据本发明的一个方面，通过引入 EGIO 通信协议栈，在每个计算装置的元件中都支持了通信协议，该栈包括物理层、数据链路层和事务层。

在本说明书各处提及的“一个实施例”或“实施例”指的是，所描述的与该实施例有关的具体特征、结构或特性被包括在本发明的至少一个实施例中。因此，在本说明书多个位置出现的短语“在一个实施例中”或“在实施例中”不必都指同一个实施例。此外，所述具体特征、结构或特性可以适当的方式结合在一个或多个实施例中。

根据前述内容和下面的描述，本领域技术人员应当意识到，本发明的一个或多个元件可以容易地以硬件、软件、传播的信号或它们的组合来实现。

## 术语

在深入讨论创新的 EGIO 互连体系结构、通信协议和相关方法的细节之前，引入将在该详细描述中使用的词汇表元素是很有帮助的：

- **通告 (Advertise)**：在 EGIO 流控制的上下文中使用，是指通过使用 EGIO 协议的流控制更新消息来指示接收器发送有关它的流控制信用 (credit) 可用性的信息的动作；
- **完成器 (Completer)**：请求所指向的逻辑设备；
- **完成器 ID**：完成器的总线标识符（例如，号码）、设备标识符和功能标识符中的一个或多个的组合，其唯一标识了请求的完成器；
- **完成 (completion)**：用于终止或部分终止一个序列的分组被称为完成。根据一个示例性实现，完成对应于在前请求，并且在某些情况下含有数据；
- **配置空间**：EGIO 体系结构中的四个地址空间中的一个。具有配置空间地址的分组被用于配置设备；
- **组件**：物理设备（即，在单个封装之中）；
- **数据链路层**：EGIO 体系结构的中间层，位于事务层（上层）和物理层（下层）之间；

- **数据链路层分组 (DLLP)**：数据链路层分组是在数据链路层产生并使用的分组，来支持在数据链路层处执行的链路管理功能；
  - **下行流 (downstream)**：指的是元件的相对位置或离开主桥的信息流；
- 5     • **端点**：具有 00h 类型配置空间头部的 EGIO 设备；
- **流控制**：用于将来自接收器的接收缓冲器信息发送到发送器，以防止接收缓冲器溢出，并且允许发送器装置服从排序规则；
  - **流控制分组 (FCP)**：事务层分组，用于将来自一个组件中的事务层的流控制信息发送到另一个组件中的事务层；
- 10    • **功能**：多功能设备的一个独立部分，在配置空间中由唯一的功能标识符（例如，功能号码）标识；
- **层次 (Hierarchy)**：定义了在 EGIO 体系结构中实现的 I/O 互连拓扑结构。层次由对应于最靠近枚举设备 (enumerating device)（例如，主 CPU）的链路的根复合体来表征；
- 15    • **层次域**：EGIO 层次被根复合体分成多个段，所述根复合体产生不只一个 EGIO 接口，其中这些段被称为层次域；
- **主桥**：将主 CPU 复合体连接到根复合体；主桥可以提供根复合体；
  - **IO 空间**：EGIO 体系结构的四个地址空间中的一个；
  - **管线 (Lane)**：物理链路的一组差分信号对，一对用于发送并且一对用于接收。N 链路由 N 条管线组成；
- 20    • **链路**：两个组件之间的双单工 (dual-simplex) 通信路径；两个端口（一个发送，一个接收）的集合以及它们的（多条）互连管线的集合；
- **逻辑总线**：在配置空间中具有相同总线号码的一系列设备之间的逻辑连接；
  - **逻辑设备**：EGIO 体系结构的元件，其在配置空间中对应于唯一的设备标识符；
  - **存储器空间**：EGIO 体系结构的四个地址空间中的一个；
  - **消息**：具有消息空间类型的分组；

- **消息空间:** EGIO 体系结构的四个地址空间中的一个。如 PCI 中定义的特殊周期作为消息空间的子集而被包括在其中，并且因此提供了（多个）与老式设备的接口；
- **（多个）老式软件模型:** 初始化、发现、配置以及使用老式设备所需的（多个）软件模型（例如，在例如 EGIO 至老式桥中包含的 PCI 软件模型有助于与老式设备的交互）；
- **物理层:** EGIO 体系结构层，其直接面对两个组件之间的通信介质；
- **端口:** 与组件相关联的接口，在该组件和 EGIO 链路之间；
- **接收器:** 通过链路接收分组信息的组件是接收器（有时称为目标）；
- **请求:** 用于开始序列的分组被称为请求。请求包括一些操代码，并且在某些情况下，包括地址和长度、数据或其它信息；
- **请求器 (requester) :** 首先将序列引入到 EGIO 域的逻辑设备；
- **请求器 ID:** 请求器的总线标识符（例如，总线号码）、设备标识符和功能标识符中的一个或多个的组合，其唯一的标识请求器。在大多数情况下，EGIO 桥或交换器（switch）将请求从一个接口转发到另一个接口而不修改请求器 ID。来自除了 EGIO 总线的总线的桥通常应当存储请求器 ID，以在为该请求产生一个完成时使用。
- **根复合体:** 包括主桥和一个或多个根端口的实体；
- **根端口:** 根复合体上的 EGIO 端口，其通过相关联的虚拟 PCI-PCI 桥来映射 EGIO 互连层次的一部分；
- **序列:** 与请求器执行单个逻辑传送相关联的零个或多个完成以及单个请求；
- **序列 ID:** 请求器 ID 和标记的一个和多个的组合，其中所述组合唯一地标识作为公共序列一部分的完成和请求；
- **分裂事务 (split transaction) :** 含有初始事务（分裂请求）的单个逻辑传送，目标（完成器或桥）以分裂响应终止该事务，随后由完成器（或桥）开始一个或多个事务（分裂完成），以将读取数据（如果读取）或完成消息发送回请求器；
- **符号 (symbol) :** 作为 8 比特/10 比特编码的结果而产生的 10 比特数

值；

- **符号时间：**在管线上放置符号所需的时间段；
  - **标记：**由请求器分配到给定序列以区分它和其它序列的号码—序列 ID 的一部分；
- 5     • **事务层分组 (TLP)：**TLP 是在事务层中产生以运送请求或完成的分组；
- **事务层：**EGIO 体系结构的最外层（最上层），其在事务级别进行操作（例如，读取、写入等等）；
- **事务描述符：**分组头部的元素，与地址、长度和类型一起描述事务的属性。

### 示例性电子装置以及 EGIO 体系结构

图 1 提供了根据本发明示例性实施例的电子装置 100 的方框图，该电子装置 100 包括增强型通用输入/输出 (EGIO) 互连体系结构、协议及相关方法。如所示，电子装置 100 被描述为包含多个电子元件，包括 (多个) 处理器 102、根复合体 (例如，包括主桥) 104、交换器 108 以及端点 110 中的一个或多个，每个元件都如所示进行耦合。根据本发明的教导，至少根复合体 104、(多个) 交换器 108 以及端点 110 被赋予了 EGIO 通信接口 106 的一个或多个示例，以有助于本发明的实施例的一个或多个方面。

如所示，元件 102、104、108 和 110 中的每一个都经由 EGIO 接口 106 通过通信链路 112 可进行通信地耦合到至少一个其它元件，其中通信链路 112 支持一条或多条 EGIO 通信信道。根据一个示例性实现，在主电子装置的初始化事件期间或者在外围设备动态连接到电子装置 (例如，热插拔设备) 之后，建立了 EGIO 互连体系结构的操作参数。如上面所介绍的，电子装置 100 被确定为代表多种传统和非传统计算系统、服务器、网络交换器、网络路由器、无线通信用户单元、无线通信电话基础设施元件、个人数字助理、机顶盒或任何电子装置中的任何一个或多个，所述任何电子装置将从通过这里描述的 EGIO 互连体系结构、通信协议或相关方

法的至少一个子集的综合而产生的通信资源获益。

根据图 1 图示的示例性实现，电子装置 100 具有一个或多个处理器 102。如这里所使用的，(多个) 处理器 102 控制电子装置 100 的功能性能力的一个或多个方面。在这个方面，(多个) 处理器 102 可以代表多种 5 控制逻辑的任何一个，控制逻辑包括但不限于微处理器、可编程逻辑器件 (PLD)、可编程逻辑阵列 (PLA)、专用集成电路 (ASIC)、微控制器等等的一个或多个。

如上所述，根复合体 104 提供电子装置 EGPIO 体系结构的一个或多个其它元件 108、110 与处理器 102 和/或处理器/存储器复合体之间的 EGPIO 10 通信接口。如这里所使用的，根复合体 104 指的是最靠近于主控制器、存储器控制器集线器、IO 控制器集线器、上述的任何组合或芯片组/CPU 元件的某种组合 (即，处于计算系统环境) 的 EGPIO 层次的逻辑实体。在这方面，尽管在图 1 中被描述为单个单元，根复合体 104 可以被认为是具有多个物理组件的单个逻辑实体。

15 根据图 1 所图示的示例性实现，根复合体 104 组装有一个或多个 EGPIO 接口 106 以便于与其它外围设备进行通信，所述外围设备例如是 (多个) 交换器 108、(多个) 端点 110 以及 (多个) 老式桥 114 或 116，尽管没有对老式桥 114 或 116 进行具体描述。根据一个示例性实现，每个 EGPIO 接口 106 代表不同的 EGPIO 层次域。在此方面，图 1 所图示的实现表示了具有三 (3) 个层次域的根复合体 104。应当知道，尽管 20 所作的表述包括多个单独 EGPIO 接口 106，但是可以预期其它的实施例，其中单个接口 106 具有多个端口以适应和多个设备进行通信。

根据一个示例性实现，根复合体 104 负责识别 EGPIO 体系结构的每个元件的通信需求 (例如，虚拟信道需求、同步信道需求等等)。根据一个 25 示例性实现，这样的通信需求在主装置 100 的初始化事件期间或它的任何元件的初始化事件期间 (例如，热插拔事件) 被传送到根复合体 104。在另一个实施例中，根复合体 104 询问这些元件以识别通信需求。一旦识别了这些通信参数，根复合体 104 就例如通过协商过程来为体系结构的每个元件建立 EGPIO 通信设备的款项和条件。

在这里公开的 EGIO 体系结构中，交换器有选择地将端点耦合到多个 EGIO 层次和/或域以及它们之间。根据一个示例性实现，EGIO 交换器具有至少一个上行流（upstream）端口（即、朝着根复合体 104 的方向）和至少一个下行流端口。根据一个实现，交换器 108 将最靠近根复合体的一个端口（即，接口的一个端口或接口 106 自身）作为上行流端口，而所有其它的端口是下行流端口。根据一个实现，交换器 108 对于配置软件（例如，老式配置软件）表现为 PCI—PCI 桥，并且使用 PCI 桥机制来对事务进行路由。

在交换器 108 的上下文中，对等事务被定义为这样的事务，其中的接收端口和发送端口都是下行流端口。根据一个实现，交换器 108 支持除了那些与从任何端口到任何其它端口的锁定事务序列相关联的事务层分组（TLP）之外的所有类型的事务层分组。在这方面，所有广播消息一般都会从交换器 108 上的接收端口被路由到它的所有其它端口。不能被路由到端口的事务层分组一般会被交换器 108 确定为不支持的 TLP。当将事务层分组（TLP）从接收端口传送到发送端口时，交换机 108 一般不修改它们，除非需要进行修改以适应发送端口（例如，耦合到老式桥 114、116 的发送端口）的不同协议需求。

应当意识到，交换器 108 代表其它设备工作，并且在这方面，它不会预先知道流量类型和模式。根据下面将要详细讨论的一个实现，本发明的流控制和数据完整性方面以每个链路（per-link）为基础而实现，而不是以端到端（end-to-end）为基础实现。因此，根据这样的实现，交换器 108 参与用于流控制和数据完整性的协议。为了参与流控制，交换器 108 为每个端口维持单独的流控制以提高交换器 108 的性能特性。类似地，交换器 108 通过使用 TLP 检错机制检查进入交换器的每个 TLP 而以每个链路为基础来支持数据完整性过程，这在下面将详细描述。根据一个实现，交换器 108 的下行流端口允许形成新的 EGIO 层次域。

继续参考图 1，端点 110 被定义为具有 00hex（十六进制 00）（00h）类型配置空间头部的任何设备。端点设备 110 代表它自身或是代表截然不同的非 EGIO 设备，可以是 EGIO 语义事务的请求器或完成器。

这样的端点 110 示例包括但不局限于 EGIO 兼容（EGIO compliant）图形设备、EGIO 兼容存储器控制器以及/或者实现了 EGIO 和诸如通用串行总线（USB）、以太网等某些其它接口之间的连接的设备。与下文详细讨论的老式桥 114、116 不同，担当非 EGIO 兼容设备的接口的端点 110 不会  
5 为这些非 EGIO 兼容设备提供完全软件支持。虽然将主处理器复合体 102 连接到 EGIO 体系结构的设备是根复合体 104，但是它可以与位于 EGIO 体系结构中的其它端点具有相同的设备类型，它们只是通过其相对于处理器复合体 102 的位置来加以区分。

根据本发明的教导，端点 110 可以被概括为下列三个类别中的一个或多个：  
10 （1）老式与 EGIO 兼容端点，（2）老式端点，以及（3）EGIO 兼容端点，每个在 EGIO 体系结构中具有不同的操作规则。

如上所述，EGIO 兼容端点 110 与老式端点（例如，118、120）不同，在于 EGIO 端点 110 将具有 00h 类型配置空间头部。这些端点  
15 （110、118 和 120）的每个都作为完成器支持配置请求。这些端点允许产生配置请求，并且可以被分类为老式端点或 EGIO 兼容端点，但是该分类需要遵守另外的规则。

老式端点（例如，118、120）被允许作为完成器来支持 IO 请求并且  
20 被允许产生 IO 请求。如果老式端点（118、120）的软件支持需求要求，则它被允许例如根据传统的 PCI 操作作为完成者产生锁定语义（lock semantics）。老式端点（118、120）一般不发布锁定请求。

EGIO 兼容端点 110 一般不作为完成器来支持 IO 请求并且不产生 IO  
请求。EGIO 端点 110 不作为完成器来支持锁定请求，并且不作为请求器来产生锁定请求。

EGIO 至老式桥 114、116 是专用端点 110，其包括用于老式设备  
25 （118、120）的基本软件支持例如完全软件支持，其中所述桥将所述老式设备连接到 EGIO 体系结构。在这方面，EGIO—老式桥 114、116 一般具有一个上行流端口（也可以具有多个），并具有多个下行流端口（也可以只有一个）。根据老式软件模型（例如，PCI 软件模型）来支持锁定请求。EGIO—老式桥 114、116 的上行流端口应当以每个链路为基础来支持

流控制并且遵守 EGIO 体系结构的流控制和数据完整性规则，这在下文将详细介绍。

如这里所使用的，通信链路 112 被确定为代表多种通信介质中的任何一个，所述多种通信介质包括但不限于铜线、光纤、（多条）无线通信信道、红外通信链路等等。根据一个示例性实现，EGIO 链路 112 是差分串行线路对，一对中的每个都支持发送和接收通信，从而提供了对全双工通信能力的支持。根据一个实现，链路提供具有初始（基本）操作频率为 2.5Ghz 的可变串行时钟频率。每个方向的接口宽度可依 x1、x2、x4、x8、x12、x16、x32 物理管线而变。如上所述以及下面将要详细介绍的，EGIO 链路 112 可以在设备之间支持多条虚拟信道，从而使用一条或多条虚拟信道在这些设备之间提供对同步流量的不间断通信的支持，所述多条虚拟信道例如是一条音频信道和一条视频信道。

### 示例性 EGIO 接口体系结构

根据图 2 所图示的示例性实现，EGIO 接口 106 可以表示为包括了事务层 202、数据链路层 204 和物理层 206 的通信协议栈。如所示，物理链路层接口被描述为包括逻辑子块 208 和物理子块 210，其中每个都将在进行下面详细讨论。

### 20 事务层 202

根据本发明的教导，事务层 202 提供 EGIO 体系结构和设备核心之间的接口。在这方面，事务层 202 的主要职责是为主设备（或代理）中的一个或多个逻辑设备装配和拆解分组（即，事务层分组或 TLP）。

### 25 地址空间、事务类型和用途

事务形成了在发起代理和目标代理之间的信息传送的基础。根据一个示例性实施例，在创新的 EGIO 体系结构中定义了四个地址空间，包括例如配置地址空间、存储器地址空间、输入/输出地址空间以及消息地址空间，每个都具有自己唯一的既定用途（例如见图 7，下面进行了详细的说

明)。

存储器空间 (706) 事务包括读取请求和写入请求中的一个或多个，以将数据发送到存储器映射位置或从该位置取出数据。存储器空间事务可以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 比特的长度）。根据一个示例性实施例，EGIO 体系结构使用锁定协议语义（即，代理可以锁定对所修改的存储器空间的访问）来提供传统的读取、修改和写入序列。更具体地说，根据特定设备规则（桥、交换器、端点、老式桥），允许对下行流锁定的支持。如上所述，支持该锁定语义以帮助老式设备。

10 IO 空间 (704) 事务用于访问 IO 地址空间（例如，16 比特 IO 地址空间）中的输入/输出映射存储器寄存器。诸如英特尔体系结构处理器以及其它处理器的某些处理器 102 通过处理器的指令集而包括 IO 空间定义。因此，IO 空间事务包括读取请求和写入请求以将数据传送至 IO 映射位置或从该位置取出数据。

15 配置空间 (702) 事务用于访问 EGIO 设备的配置空间。配置空间的事务包括读取请求和写入请求。由于如此多的传统处理器一般不含有本地配置空间，所以通过一种机制来映射该空间，所述机制即是与传统 PCI 配置空间访问机制（例如，使用基于 CFC/CFC8 的 PCI 配置机制#1）相兼容的软件。或者，也可以使用存储器别名机制来访问配置空间。

20 消息空间 (708) 事务（或简称为消息）被定义为支持通过（多个）接口 106 而在 EGIO 代理之间进行带内通信。由于传统的处理器不包括对本地消息空间的支持，所以这是通过 EGIO 代理在接口 106 中实现的。根据一个示例性实现，诸如中断和电源管理请求的传统“边带（side-band）”信号作为消息而被实现以减少所需的用来支持这些老式信号的引脚数目。一些处理器以及 PCI 总线包括“特殊周期”的概念，其也被映射到 EGIO 接口 106 中的消息。根据一个实施例，消息通常分为两类：标准消息和厂商定义消息。

根据所图示的示例性实施例，标准消息包括通用消息组和系统管理消息组。通用消息可以是单一目的地消息或广播/组播消息。系统管理消息

组可以包括中断控制消息、电源管理消息、排序控制原语（primitive）和错误信令中的一个或多个，它们的例子将在下文介绍。

根据一个示例性实现，通用消息包括支持锁定事务的消息。根据该示例性实现，引入了 UNLOCK（解锁）消息，其中交换器（例如，108）一般会通过可能参与锁定事务的任何端口来运送 UNLOCK 消息。在没有被锁定的时候接收到 UNLOCK 消息的端点设备（例如，110、118、120）将忽略该消息。否则，将在接收到 UNLOCK 消息之后解锁锁定设备。

根据一个示例性实现，系统管理消息组包括用于排序和/或同步的专用消息。一个这样的消息是 FENCE（防护）消息，用于在由 EGIO 体系结构的接收元件产生的事务上施加严格的排序规则。根据一个实现，只是诸如端点的网络元件的一个精选子集对该 FENCE 消息作出反应。除了前述的内容，例如通过使用下文讨论的尾部（tailer）错误转发，这里还预见了用于指示可校正错误、不可校正错误和致命错误的消息。

根据上文所介绍的本发明的一个方面，系统管理消息组使用带内消息提供中断信令。根据一个实现，引入了 ASSERT\_INTx/DEASSERT\_INTx 消息对，其中断言（assert）中断消息的发布通过根复合体 104 被发送到处理器复合体。根据所图示的示例性实现，ASSERT\_INTx/DEASSERT\_INTx 消息对的使用规则反映了 PCI 规范中的 PCI INTx #信号的消息的使用规则，如上所述。对于来自任何一个设备的 Assert\_INTx 的每次发送，通常都有对应的 Deassert\_INTx 的发送。对于特定‘x’（A、B、C 或 D），一般在发送 Deassert\_INTx 之前只发送一次 Assert\_INTx。交换器一般会将 Assert\_INTx/Deassert\_INTx 消息路由到根复合体 104，其中根复合体一般会跟踪 Assert\_INTx/Deassert\_INTx 消息以产生虚拟中断信号，并且将这些信号映射到系统中断资源。

除了通用和系统管理消息组之外，EGIO 体系结构建立了标准框架结构，其中核心逻辑（例如芯片组）厂商可以定义它们自己的厂商定义消息以迎合它们的平台的特定操作需求。该框架结构是通过公共消息头部而建立的，在所述头部中厂商定义消息的编码被规定为“预留”。

## 事务描述符

5 事务描述符是用于将事务信息从起点运送到服务点并送回的机制。它提供可扩展装置用于提供可以支持新类型的新应用的一般互连解决方案。在这方面，事务描述符支持系统中的事务的标识、缺省事务排序的修改，以及使用虚拟信道 ID 机制关联事务与虚拟信道。参考图 3，示出了事务描述符的示图。

10 参考图 3，根据本发明的教导示出了包括示例性事务描述符的数据报的示图。根据本发明的教导，示出的事务描述符 300 包括全局标识符字段 302、属性字段 304 和虚拟信道标识符字段 306。在所图示的示例性实现中，全局标识符字段 302 被描述为包括本地事务标识符字段 308 和源标识符字段 310。

### • 全局事务标识符 302

15 如这里所使用的，全局事务标识符对所有待处理的请求都是唯一的。  
20 根据图 3 所图示的示例性实现，全局事务标识符 302 包括两个子字段：本地事务标识符字段 308 和源标识符字段 310。根据一个实现，本地事务标识符字段 308 是由每个请求器产生的 8 比特字段，并且对于需要该请求器的完成的所有待处理请求它是唯一的。源标识符唯一地标识 EGIO 层次中的 EGIO 代理。因此，本地事务标识符字段和源 ID 一起提供了在层次域中的事务的全局标识。

25 根据一个实现，本地事务标识符 308 允许来自单个请求源的请求/完成不依顺序（遵守下面详细讨论的排序规则）而被操作。例如，读取请求源可以产生读取 A1 和 A2。处理这些读取请求的目的地代理会首先返回请求 A2 事务 ID 的完成，并且随后返回 A1 的完成。在完成分组头部中，本地事务 ID 信息将标识哪个事务将被完成。这种机制对于使用分布式存储器系统的装置尤为重要，因为它可以更有效的方式来操作读取请求。应当注意，对这种不依顺序读取完成的支持假定了发布读取请求的设备将确保完成的缓冲器空间的预先分配。如上所述，只要 EGIO 交换机 108 不是端点（即，仅仅传送完成请求到适当的端点），它们就不需要预留缓冲器

空间。

单个读取请求可以产生多个完成。属于单个读取请求的完成可以相互不依顺序的返回。这通过在完成分组头部（即，完成头部）中提供对应于部分完成的初始请求的地址偏移来只支持。

5 根据一个示例性实现，源标识符字段 310 包含 16 比特值，其对每个逻辑 EGIO 设备是唯一的。应当注意单个 EGIO 设备可以包括多个逻辑设备。在系统配置期间以对标准 PCI 总线枚举机制透明的方式分配源 ID 值。EGIO 设备使用例如在对那些设备的初始配置访问期间可用的总线号码信息以及用于表示例如设备号码和流号码的内部可用信息，在内部自动 10 地建立源 ID。根据一个实现，该总线号码信息是在 EGIO 配置周期期间使用与 PCI 配置所使用的相类似的机制而产生的。根据一个实现，总线号码由 PCI 初始化机制分配并由每个设备捕获。在热插拔和热交换设备的情况下，这些设备将需要在每个配置周期访问上重新捕获该总线号码信息以能够对热插拔控制器（例如，标准热插拔控制器（SHPC））软件栈透 15 明。

根据 EGIO 体系结构的一个实现，物理组件可以包含一个或多个逻辑设备（或代理）。每个逻辑设备被设计成响应于指定到其特定设备号码的配置周期，即，在逻辑设备中加入了设备号码的概念。根据一个实现，在单个物理组件中允许多达十六个逻辑设备。每个这样的逻辑设备可以包括 20 一个或多个流化（streaming）引擎，例如最多 16 个。因此，单个物理组件可以包括多达 256 个流化引擎。

由不同源标识符标记的事务属于不同的逻辑 EGIO 输入/输出（IO）源，并且从而从排序的方面来看可以相互完全独立地操作这些事务。对于三方、对等事务的情况，如果需要可以使用防护排序控制原语来强制排序。 25

如这里所使用的，事务描述符 300 的全局事务标识符字段 302 遵守下列规则的至少一个子集：

- (a) 每个需要完成的请求用全局事务 ID (GTID) 来标记；
- (b) 由代理发起的所有待处理的需要完成的请求一般应当分配唯一

的 GTID;

- (c) 不需要完成的请求不使用 GTID 的本地事务 ID 字段 308，并且本地事务 ID 字段被认为是预留的；
- (d) 目标不需要以任何方式来修改请求 GTID，而只是为所有与请求相关联的完成在完成分组的头部中回应它，其中发起者使用 GTID 将（多个）完成与原始请求相匹配。

- 属性字段 304

如这里所使用的，属性字段 304 指明了事务的特性和关系。在这方面，属性字段 304 被用于提供允许修改事务的缺省操作的额外信息。这些修改可以应用于在系统中操作事务的不同方面，例如排序、硬件一致性（coherency）管理（例如探听（snoop）属性）和优先级。一种示例性格式以子字段 312-318 来表示属性字段 304。

如所示，属性字段 304 包括优先级子字段 312。优先级子字段可以由发起者修改以为事务分配优先级。在一个示例性实现中，事务或代理的服务特性的等级或质量可以在优先级子字段 312 中实现，从而影响其它系统元件进行的处理。

预留属性字段 314 为将来或厂商定义用途而被预留。通过使用预留属性字段可以实现使用优先级或安全属性的用途模型。

排序属性字段 316 被用于提供用来传达排序类型的可选信息，所述信息可以修改同一排序平面（plane）（其中排序平面包括由具有对应的源 ID 的 IO 设备和主处理器（102）发起的流量）内的缺省排序规则。根据一个示例性实现，排序属性 ‘0’ 表示将应用缺省的排序规则，而排序规则 ‘1’ 表示松散（relaxed）排序，其中在同一方向上写入可以超过写入，并且在同一方向上读取完成可以超过写入。使用松散排序语义的设备主要用于以缺省排序来为读取/写入状态信息移动数据和事务。

探听属性字段 318 被用于提供用来传达高速缓存一致性管理的类型的可选信息，所述信息可以修改同一排序平面内的缺省高速缓存一致性管理规则，其中排序平面包括由具有对应的源 ID 的 IO 设备和主处理器

(102) 发起的流量。根据一个示例性实现，探听属性字段 318 值 ‘0’ 对应于缺省高速缓存一致性管理方案，其中探听事务以增强硬件级别的高速缓存一致性。另一方面，探听属性字段 318 中的值 ‘1’ 中止缺省高速缓存一致性管理方案，并且事务没有被探听。相反，所访问的数据或者是非可高速缓存的（non-cacheable），或者其一致性由软件来管理。

- 虚拟信道 ID 字段 306

如这里所使用的，虚拟信道 ID 字段 306 标识与事务相关联的独立虚拟信道。根据一个实施例，虚拟信道标识符（VCID）是 4 比特字段，其 10 允许以每个事务为基础来标识多达 16 个虚拟信道（VC）。下面的表 I 中提供了 VCID 定义的一个示例：

VCID	VC 名字	用途模型
0000	缺省信道	通用流量
0001	同步信道	本信道用于运送具有下列需求的 IO 流量： (a) 没有探听 IO 流量以照顾到确定性服务定时；以及 (b) 使用 X/T 协定（其中 X=数据总量、T=时间）来控制服务质量
0010-1111	预留	将来使用

表 I：虚拟信道 ID 编码

### 虚拟信道

15 根据本发明的一个方面，EGIO 接口 106 的事务层 202 支持在 EGIO 通信链路 112 的带宽内建立和使用（多条）虚拟信道。如上所述的本发明的虚拟信道（VC）方面被用于基于将要通过信道传输的内容的所需的独立性而在单个物理 EGIO 链路 112 中定义单独的逻辑通信接口。在这方

面，虚拟信道可以基于一个或多个特性来建立，例如带宽需求、服务等级、服务类型（例如系统服务信道）等等。

（多条）虚拟信道和流量（或事务）等级标识符的组合被提供以支持某些等级的应用支持的有区别的服务和服务质量（QoS）。如这里所使用的，  
5 流量（或事务）等级是事务层分组标签，其通过 EGIO 组织结构而未经修改端到端地进行传输。在每个服务点（例如，交换器、根复合体等等）处，服务点使用流量等级标签来应用适当的服务策略。在这方面，单独的 VC 被用于映射流量，所述流量将从不同操作策略和服务优先级获  
10 益。例如，就确保 T 时间段内所传输的数据量 X 而言，需要确定性服务质量的流量可以被映射到同步（或时间协同）虚拟信道。映射到不同虚拟信道的事务相互之间可以没有任何排序需求。即，虚拟信道作为单独逻辑接口来操作，其具有不同的流控制规则和属性。

根据本发明的一个示例性实现，EGIO 兼容元件的每个 EGIO 通信端口（输入或输出）包括端口能力数据结构（未具体描述）。包括（a）由  
15 端口支持的虚拟信道的数目，（b）与每个虚拟信道相关联的流量等级，  
（c）端口 VC 状态寄存器，（d）端口 VC 控制寄存器，以及（e）与这样的虚拟信道相关联的仲裁方案中的一个或多个的关于端口能力的信息保持在端口能力数据结构中。根据一个示例性实现，以每个链路、每个 VC 为  
基础在耦合的元件之间协商通信操作参数和关联的端口能力参数。

20 对于由主处理器 102 发起的流量，虚拟信道可以要求基于缺省排序机制规则的排序控制，或者可以完全不依顺序地操作流量。根据一个示例性实现，VC 包含下列两种类型的流量：通用 IO 流量和同步流量。即，根据该示例性实现，描述了两类虚拟信道：（1）通用 IO 虚拟信道，和（2）同步虚拟信道。

25 如这里所使用的，事务层 202 为组件主动支持的一个或多个虚拟信道的每个保持独立流控制。如这里所使用的，所有的 EGIO 兼容组件一般都会支持缺省通用 IO 类型虚拟信道，例如虚拟信道 0，它的服务等级是“尽最大努力（best effort）”，其中在这一类型的不同虚拟信道之间不需要排序关系。缺省地，VC0 被用于通用 IO 流量，而 VC1 或更高（VC1-

VC7) 被分配用于操作同步流量。在另一个实现中，任何虚拟信道都可以被分配用于操作任何流量类型。参考图 4，示出了包括多条独立管理的虚拟信道的 EGIO 链路的概念示图。

参考图 4，根据本发明的一个方面，示出了包括多条虚拟信道 5 (VC) 的示例性 EGIO 链路 112 的示图。根据图 4 所图示的示例性实现，示出的 EGIO 链路 112 包括在 EGIO 接口 106 之间创建的多条虚拟信道 402、404。根据一个示例性实现，对于虚拟信道 402，示出了来自多个源 406A...N 的流量，这些流量至少由它们的源 ID 来区分。如所示，建立了虚拟信道 402，并且在来自不同源（例如，代理、接口等）的事务之间 10 没有排序需求。

类似地，示出的虚拟信道 404 包括来自多个源多个事务 408A...N 的流量，其中每个事务由至少一个源 ID 指示。根据图示的示例，来自源 ID 0 406A 的事务被严格排序，除非由事务头部的属性字段 304 所修改，而来自源 408N 的事务没有这样的排序规则。

15

### 同步信道

如上所述，建立同步信道以在电子装置 100 的 EGIO 体系结构中的请求器代理和（多个）完成器代理之间传输对时间敏感的内容（例如，多媒体内容流）。根据一个示例性实现，在 EGIO 体系结构之间存在两个不同的同步通信范例，例如，端点到根复合体模型以及对等（或端点到端点）通信模型。  
20

在端点到根复合体模型中，主要的同步流量是对根复合体 104 的存储器读取和写入请求以及来自根复合体 104 的读取完成。在对等模型中，同步流量局限为单播（unicast）、仅压入（push-only）事务（例如，诸如存储器写入的公布事务或消息）。仅压入事务可以在单个主域中或是多个主域中。  
25

为了支持具有保证带宽和确定性服务延迟的同步数据传输，在请求器/完成器对和 EGIO 通信组织结构之间建立了同步“协定”。根据一个实施例，“协定”将执行资源预留和流量调整以防止虚拟信道上的拥塞和过度

预约。

参考图 5，示出了用于在 EGIO 体系结构中建立并管理同步通信信道的示例性方法。根据图 5 所图示的示例性实施例，方法以方框 502 开始，其中识别了 EGIO 组织结构的一个或多个元件（即、根复合体 104、交换器 108、端点 110、链路 112、桥 114 等等）的通信能力。  
5

根据一个示例性实现，EGIO 组织结构的至少一个子集的通信能力对根复合体 104 的带宽管理器公开，该带宽管理器管理 EGIO 体系结构中的同步通信资源的地址分配。在元件的初始化期间发生元件的通信能力的公开，例如在主电子装置 100 启动的时候，或者在 EGIO 兼容设备热插拔到主电子装置的时候。根据一个实施例，公开的信息（例如来自 EGIO 代理 106 中的数据结构）包括端口标识、端口地址分配、（多个）虚拟信道分配、带宽能力等等中的一个或多个。该信息保存在可由带宽管理器访问的数据结构中以用于生成同步协定，这在下文将详细描述。  
10  
15

在电子装置 100 的常规操作过程期间，可能需要或期望在装置 100 中的两个（或多个）代理之间建立同步通信信道。在这种情况下，在方框 504 中，根复合体 104 的带宽管理器从（或代表）请求器/完成器对接收对 EGIO 组织结构中的同步通信资源的请求。如这里所使用的，请求包括诸如带宽和服务延迟需求的期望通信资源的指示。  
20

在方框 506 中，在接收到同步通信资源的请求之后，根复合体 104 的带宽管理器分析 EGIO 体系结构的至少一个适当子集的可用通信资源，以在方框 508 中确定同步通信资源的请求是否合适。根据一个实施例，根复合体 104 的带宽管理器分析包括了请求器和完成器之间的通信路径的与端口 106、（多个）交换器 108、（多条）链路 112 等相关联的信息，来确定是否可以满足合同同步通信请求的带宽和服务延迟需求。在另一个实施例中，请求器/完成器对只是以逐个链路为基础在它们自身和任何介入元件之间建立同步协定（或关于操作参数的协商协约）  
25

如果在方框 508 中根复合体 104 的带宽管理器确定请求的通信资源不可用，则根复合体丢弃同步信道的请求，并且在方框 510 中可以提供所请求的资源不可用的指示。根据某些实施例，可用资源的指示会提供给请求

器/完成器对，随后请求器/完成器对还是根据所指示的可用资源，可以决定重新发布同步通信资源的请求。在另一个实施例中，带宽管理器将通知请求了资源的实体分配了某个带宽（其可能小于所请求的）。在这种情况下，请求实体不需要重新发布请求。

5 根据一个示例性实施例，在确定是否能满足对同步通信资源的请求时，并且在方框 512 中建立同步协定时，根复合体 104 的带宽管理器如下计算请求器/完成器对的带宽需求：

$$BW = (N \cdot Y) / T$$

[1]

10 公式将分配带宽 (BW) 定义为在特定时间段 (T) 内具有特定有效载荷大小 (Y) 的事务的特定数目 (N) 的函数。

15 同步协定中的另一个重要参数是延迟。基于协定，同步事务可以在特定的延迟 (L) 内完成。一旦带宽管理器允许请求器/完成器对进行同步通信，在常规操作条件下，完成器和介入的 EGIO 体系结构元件（例如，交換器、（多条）链路、根复合体等等）向请求器保证带宽和延迟。

因此，在方框 512 中产生的同步协定规定了由（多个）EGIO 接口 106 实现的特定服务纪律，该接口 106 参与了 EGIO 体系结构中的同步通信。以下述方式将服务纪律作用于 EGIO 交換器 108 和完成器（例如，端点 110、根复合体 104 等等），即注入请求的服务受特定服务时间间隔 20 ( $t$ ) 支配。该机制用于提供当请求器发出的同步分组被处理时的控制方法。

因此，在方框 514 中以下述方式管理同步流量，即只有遵照协商的同步协定而被注入到 EGIO 体系结构的分组才允许立即前进，并且开始由 EGIO 体系结构元件处理。通过流控制机制，阻止了试图注入比按照协商 25 协议允许的更多的同步流量的不兼容请求器进行这样的操作，这将在下文详细描述（例如见数据链路层特征集）。

根据一个示例性实现，同步时间段 (T) 被均匀的划分为多个虚拟时隙 ( $t$ ) 的单元。在一个虚拟时隙中最多允许一个同步请求。根据一个实施例，作为 EGIO 接口的数据结构中的头部信息来提供由 EGIO 组件支持

的虚拟时隙的大小（或持续时间）。在另一个实现中，在接收到初始化事件（例如，冷启动、复位等等）时，通过来自 EGIO 组件的广播消息来报告虚拟时隙的大小。在另一个实现中，在接收到专用请求消息时，通过来自 EGIO 组件的专用信息消息来报告虚拟时隙的大小。在另一个实现中，  
5 虚拟时隙的大小可以是固定的，并且同步带宽管理器软件可以以下述方式交错嵌入有效（active）和无效（inactive）时隙（在带宽分配期间），所述方式有效地创建“较宽”的时隙。

根据一个实施例，虚拟时隙（t）的持续时间是 100ns。同步时间段（T）的持续时间取决于所支持的基于时间仲裁方案（例如，基于时间加权轮询（weighted round-robin， WRR）（或加权顺序））的阶段（phase）数目。根据一个实施例，阶段数目由同步虚拟时隙的数目规定，并且由每个元件中保持的端口仲裁表中的条目数目指示。当端口仲裁表大小等于 128 时，在同步时间段中有 128 个虚拟时隙（t），即， $T=12.8 \mu s$ 。  
10

15 根据一个示例性实施例，在 EGIO 配置期间建立同步事务的最大有效载荷大小（Y）。在配置之后，在给定 EGIO 层次域最大有效载荷大小是固定的。固定最大有效载荷大小的值被用于同步带宽预算，而不考虑与请求器/完成器之间的同步事务相关联的数据有效载荷的实际大小。

20 在讨论了同步时间段（T）、虚拟时隙（t）和最大有效载荷（Y）的条件下，时间段中的虚拟时隙的最大数目是：

$$N_{max} = T/t$$

[2]

并且，最大可指定同步带宽是：

$$BW_{max} = Y/t$$

25 [3]

从而，同步带宽可以分配的粒度（granularity）定义如下：

$$BW_{粒度} = Y/T$$

[4]

将同步带宽  $BW_{链路}$  分配给通信链路 112 与按照每个同步时间段（T）分配

$N_{\text{链路}}$  虚拟时隙相类似，其中  $N_{\text{链路}}$  由下式给出：

$$N_{\text{链路}} = BW_{\text{链路}} / BW_{\text{粒度}}$$

[5]

为了保持对链路的受调节的访问，用作同步流量的出口（egress port）的交换器端口建立具有多达  $N_{\max}$  个条目的数据结构，其中  $N_{\max}$  是在给定链路带宽、粒度和延迟需求的条件下容许的同步会话的最大数目。表中的一个条目代表同步时间段（T）中的一个虚拟时隙。当表条目被给定端口号（PN）的值时，意味着该时隙被分配给由端口号指定的入口（ingress port）。因此，当端口仲裁表中的  $N_{\text{链路}}$  条目被给定了 PN 的值时， $N_{\text{链路}}$  虚拟时隙被分配给入口。只有当由出口的同步时间计数器（其每隔  $t$  时间增加 1，并且当到达 T 时重新开始计数）访问的表条目被设定为 PN 时，出口才会容许来自入口的对其他服务的一个同步请求事务。即使在入口 中准备好了待处理的同步请求，直到下一轮仲裁（例如，基于时间、加权轮询（WRR）仲裁）才会处理它。以此方式，基于时间的端口仲裁数据结构用作同步带宽分配和流量调节。

如这里所使用的，上面讨论的事务延迟由通过 EGIO 组织结构的延迟和完成器产生的延迟两者组成。为每个事务定义同步事务延迟，并且以虚拟时隙  $t$  为单位测量同步事务延迟。

对于端点到根复合体通信模型中的请求器，读取延迟定义为往返程延迟，即，从设备向它的事务层递交存储器读取请求分组（在发送方）时到对应的读取完成到达设备的事务层（接收方）时的延时。对于任一个通信模型中的请求器，写入延迟定义为从请求器发送存储器写入请求到其事务层的发送端时到数据写入变得在完成器的存储器子系统中全局可见时的延时。当访问存储器地址的所有代理获得更新数据时，对存储器的写入达到全局可见的情况。

作为同步协定的一部分，提供了同步事务延迟的上边界和下边界。请求器中的同步数据缓冲器大小可以使用最小和最大同步事务延迟来确定。如下文所详细介绍的，最小同步事务延迟比最大同步事务延迟小得多。

对于请求器，可以根据下面的等式（6）来计算最大同步（读取或写

入) 事务延迟 (L) ,

$$L = L_{\text{组织结构}} + L_{\text{完成器}}$$

[6]

其中  $L_{\text{组织结构}}$  是 EGIO 组织结构的最大延迟, 而  $L_{\text{完成器}}$  是完成器的最大延

5 迟。

EGIO 链路 112 或 EGIO 组织结构的事务延迟定义为从事务在发送端公布时到它在接收端可用时的延时。这适用于读取和写入事务两者。在这方面,  $L_{\text{组织结构}}$  取决于拓扑结构、由每条链路 112 引起的延迟以及从请求器到完成器的路径中的仲裁点。

10 继续参考图 5, 过程前进到方框 516, 其中带宽管理器确定同步通信信道的使用是否完成。即, 带宽管理器确定同步通信对话是否已经结束, 并且因而确定为支持同步信道而分配的虚拟信道资源是否可被释放而由 EGIO 组织结构使用。根据一个实施例, 带宽管理器从一个或多个请求器/完成器对接收指示, 即不再需要同步资源的指示。在另一个实施例中, 在 15 某个无效时间段之后带宽管理器推断出同步通信已经结束。

如果在方框 516 中带宽管理器确定同步通信没有结束, 则过程回到方框 514。

或者, 过程前进到方框 518, 其中带宽管理器取消同步协定, 从而释放该带宽以支持余下的虚拟信道。根据一个实施例, 带宽管理器通知 20 EGIO 体系结构的一个或多个其他元件, 同步协议不再有效。

### 事务排序

尽管使所有响应依次序被处理可能更简单, 但是事务层 202 试图通过准许事务的重新排序来提高性能。为了便于这样的重新排序, 事务层 202 25 “标记” 事务。即根据一个实施例; 事务层 202 添加事务描述符到每个分组, 使得它的传输时间可以由 EGIO 体系结构中的元件来优化 (例如, 通过重新排序), 且不会丢失分组最初被处理的相对顺序。这样的事务描述符被用于帮助请求和完成分组通过 EGIO 接口层次而进行路由。

因而, EGIO 互连体系结构和通信协议的创新方面之一是它提供了不

依顺序通信，从而通过减少空闲或等待状态来提高数据吞吐量。在这方面，事务层 202 使用了一组规则来定义 EGIO 事务的排序需求。定义了事务排序需求来确保软件的正确操作，所述软件被设计成支持生产者—消费者排序模型，同时允许基于不同排序模型（例如，图形附着应用的松散排序）的应用的改进的事务操作灵活性。下文描述了两种不同类型的排序需求：单个排序平面模型和多个排序平面模型。

- 基本事务排序—单个“排序平面”模型

假定以下两个组件通过与图 1 相似的 EGIO 体系结构连接起来：存储器控制集线器，提供到主处理器和存储器子系统的接口；以及 IO 控制集线器，提供到 IO 子系统的接口。两个集线器都含有用于操作输入和输出流量的内部队列，并且在这个简单模型中所有 IO 流量都被映射到单个“排序平面”。（注意，事务描述符源 ID 信息为 EGIO 层次中的每个代理都提供了唯一的标识符，还要注意，映射到源 ID 的 IO 流量可以携带不同事务排序属性）。在 IO 发起（IO-initiated）的流量和主发起（host-initiated）的流量之间规定了本系统配置的排序规则。根据上述说法，映射到源 ID 的 IO 流量和主处理器发起的流量代表在单个“排序平面”中传递的流量。

参考表 II，下面提供了该事务排序规则的示例。该表中定义的规则普遍适用于包括存储器、IO 配置和消息的 EGIO 系统中的所有类型的事务。在下面的表 II 中，列代表两个事务的第一个，而行代表第二个。表条目指明了两个事务之间的排序关系。表条目定义如下：

是—一般会允许第二个事务超过第一个事务以避免死锁。（当发生阻塞时，需要第二个事务超过第一个事务。一般应当考虑公平以防 25 止饥饿（starvation））。

Y/N—没有需求。第一个事务可选地超过第二个事务或者被其阻塞。

否—一般不会允许第二个事务超过第一个事务。这需要保持严格的排序。

行 超 过	WR_Req	RD_Req	WR_Req	RD_Comp	WR_Comp
-------	--------	--------	--------	---------	---------

列?	(没有完成请求) (第 2 列)	(第 3 列)	(完 成 请 求) (第 4 列)	(第 5 列)	(第 6 列)
WR_Req (没有完成请求) (第 A 行)	否	是	a. 否 b. 是	Y/N	Y/N
RD_Req (第 B 行)	否	a. 否 b. Y/N	Y/N	Y/N	Y/N
WR_Req (完 成 请 求) (第 C 行)	否	Y/N	a. 否 b. Y/N	Y/N	Y/N
RD_Comp (第 D 行)	否	是	是	a. 否 b. Y/N	Y/N
WR_Comp (第 E 行)	Y/N	是	是	Y/N	Y/N

表 II: 单个排序平面的事务排序和死锁避免

行: 列 ID	表 II 条目的解释
A2	公布的存储器写入请求 (WR_REQ) 一般不应该超过任何其它公布的存储器写入请求
A3	一般应该允许公布的存储器写入请求超过读取请求以避免死锁
A4	a. 一般不应该允许公布的存储器 WR_REQ 超过具有完成请求属性的存储器 WR_REQ. b. 一般应该允许公布的存储器 WR_REQ 超过 IO 和配置请求以避免死锁

A5, A6	不需要公布的存储器 WR_REQ 超过完成。为了允许这一实现的灵活性，同时确保免除死锁的操作，EGIO 通信协议规定代理确保完成的接收
B2, C2	这些请求不能超过公布的存储器 WR_REQ，从而保持支持生产者/消费者用途模型所需的严格写入排序
B3	a.在基本实现（即，没有不依顺序处理）中读取请求不准许互相超过。 b.在另一个实现中，读取请求准许互相超过。事务标识对于提供这个功能很重要。
B4, C3	准许不同类型的请求相互阻塞或超过。
B5, B6, C5, C6	准许这些请求被完成阻塞或超过完成。
D2	读取完成不能超过公布的存储器 WR_Req（以保持严格的写入排序）
D3, D4, E3, E4	一般应该允许完成超过没有公布的请求以避免死锁
D5	a.在基本实现中，不准许读取完成互相超过 b.在另一个实施例中，准许读取完成互相超过。再者，需要严格事务标识。
E6	准许这些完成互相超过。对于使用例如事务 ID 机制来维持事务的轨迹很重要
D6, E5	不同类型的完成可以互相超过。
E2	准许写入完成被公布的存储器 WR_REQ 阻塞或超过公布的存储器 WR_REQ。该写入事务实际上向相反方向运动，因而没有排序关系

表 III: 事务排序解释

- 高级事务排序—“多个平面”事务排序模型

前述部分定义了单个“排序平面”内的排序规则。如上所述，EGIO 互连体系结构和通信协议使用唯一的事务描述符机制来关联事务和额外的信息，以支持更复杂的排序关系。事务描述符中的字段允许创建多个“排

序平面”，从 IO 流量排序来看这些排序平面是互相独立的。

每个“排序平面”都包括对应于具体 IO 设备（由唯一的源 ID 指定）的排队/缓冲逻辑以及传输主处理器发起的流量的排队/缓冲逻辑。“平面”内的排序一般只在这两者之间定义。对独立于其它“排序平面”的 5 每个“排序平面”都实施了在前述部分规定的用来支持生产者/消费者用途模型并且防止死锁的规则。例如，由“平面”N 发起的请求的读取完成可以绕过由“平面”M 发起的请求的读取完成。然而，平面 N 的读取完成和平面 M 的读取完成都不能绕过由主机发起的公布存储器写入。

尽管平面映射机制的使用允许存在多个排序平面，但是排序平面中的一些或全部可以“折叠”到一起以简化实现（即，将多个单独控制的缓冲器/FIFO 结合成单个）。当所有平面折叠在一起时，仅使用事务描述符 10 源 ID 机制来帮助事务的路由，并且它不用于在 IO 流量的独立流之间松散排序。

除了上述的内容，事务描述符机制规定了使用排序属性在单个排序平面内修改缺省排序。从而可以以每个事务为基础而控制排序的修改。 15

### 事务层协议分组格式

如上所述，创新 EGPIO 体系结构使用基于分组的协议以在相互通信的两个设备的事务层之间交换信息。EGIO 体系结构通常支持存储器、IO、 20 配置和消息事务类型。一般使用要求或完成分组运送这些事务，其中只有当要求时，即要求返回数据或请求事务的确认接收时，才使用完成分组。

参考图 9，根据本发明的教导示出了示例性事务层协议的示图。根据图 9 所图示的示例性实现，图示的 TLP 头部 900 包括格式字段、类型字段、扩展类型/扩展长度 (ET/EL) 字段和长度字段。应当知道，某些 TLP 25 在头部之后包括如头部中列出的格式字段确定的数据。没有 TLP 可以含有大于 MAX\_PAYLOAD\_SIZE 设定的极限的数据。根据一个示例性实现，TLP 数据是 4 字节自然对齐的，并且以 4 字节双字 (DW) 增加。

如这里所使用的，根据下面的定义格式 (FMT) 字段规定了 TLP 的格式：

- 000 – 2DW 头部, 无数据
- 001 – 3DW 头部, 无数据
- 010 – 4DW 头部, 无数据
- 101 – 3DW 头部, 有数据
- 5        • 110 – 4DW 头部, 有数据
- 预留所有其它的编码

类型字段用于指示 TLP 中使用的类型编码。根据一个实现, 一般应  
该解码格式[2:0]和类型[3:0]两者来确定 TLP 格式。根据一个实现, 类型  
10 [3:0]字段中的值用于确定扩展类型/扩展长度字段是否被用于扩展类型字  
段或长度字段。ET/EL 字段一般只用于扩展存储器类型读取请求的长度字  
段。

长度字段提供了有效载荷长度的指示, 还是以 DW 增加, 如下所示:

:0000 0000=1DW  
15        :0000 0001=2DW  
:.....  
:1111 1111=256DW

下面提供了示例性 TLP 事务类型的至少一个子集、它们对应的头部  
格式以及描述的总结, 表 IV 中:

TLP 类型	FMT [2:0]	类型 [3:0]	Et [1:0]	描述
初始 FCP	000	0000	00	初始流控制信息
更新 FCP	000	0001	00	更新流控制信息
MRd	001	1001	E19 E18	存储器读取请求
	010			Et/E1 字段用于长度[9:8]
MRdLK	001	1011	00	存储器读取请求一锁定
	010			
MWR	101	0001	00	存储器写入请求一公布
	110			

<b>IORd</b>	001	1010	00	IO 读取请求
<b>IOWr</b>	101	1010	00	IO 写入请求
<b>CfgRd0</b>	001	1010	01	配置读取类型 0
<b>CfgWr0</b>	101	1010	01	配置写入类型 0
<b>CfgRd1</b>	001	1010	11	配置读取类型 1
<b>CfgWr1</b>	101	1010	11	配置写入类型 1
<b>Msg</b>	010	011s2	Sls0	消息请求一子字段 s[2:0] 规定一组消息。根据一个实现，该消息字段被解码以确定包括是否需要完成的特殊周期
<b>MsgD</b>	110	001s2	Sls0	带有数据的消息请求一子字段 s[2:0] 规定一组消息。根据一个实现，该消息字段被解码以确定包括是否需要完成的特殊周期
<b>MsgCR</b>	010	111s2	Sls0	需要完成的消息请求一子字段 s[2:0] 规定一组消息。根据一个实现，该消息字段被解码以确定特殊周期
<b>MsgDCR</b>	110	111s2	Sls0	需要完成且带有数据的消息请求一子字段 s[2:0] 规定一组消息。根据一个实现，决定特殊周期字段以确定特殊周期
<b>CPL</b>	001	0100	00	不带有数据的完成一用于具有除成功完成之外

				的完成状态的存储器读取完成、IO 和配置写入完成依旧某些消息完成
CplD	101	0100	00	带有数据的完成—用于存储器、IO 和配置读取完成，以及某些消息完成
CplDLK	101	001	01	锁定存储器读取的完成—否则与 CplD 类似

表 IV: TLP 类型总结

附录 A 中提供了有关请求和完成的其它细节，其中的说明在这里作为参考而被明确引入。

5

### 流控制

与传统流控制方案普遍关联的限制之一是它们对可能发生的问题有反应 (reactive)，而不是在预先 (proactively) 降低发生这些问题首先发生的机会。例如在传统的 PCI 系统中，发送者将向接收者发送信息直到它接收到停止/中止发送的消息。其中停止/中止发送直到下一个通知。这些请求随后可以跟随有重新发送起始于发送的给定点处的分组的请求。而且，目前这样的流控制机制是基于硬件的，它们不适合上述动态建立、独立管理的虚拟信道应用。本领域技术人员将理解，这一反应 (reactive) 方法导致周期浪费，并且在这方面可能效率较低。

为了解决这个限制，EGIO 接口 106 的事务层 202 包括流控制机制，其预先降低发生溢出情况的机会，同时还规定以发起者和 (多个) 完成者之间建立的虚拟信道的每个链路为基础来遵守排序规则。

根据本发明的一个方面，引入了流控制“信用”的概念，其中接收者共享下列信息：(a) 缓冲器 (信用) 大小，和 (b) 对于发送者和接收者之间建立的每条虚拟信道 (即以每条虚拟信道为基础) 的发送者当前可用

缓冲器空间。这使得发送者的事务层 202 能够保持可用缓冲器空间的估值（例如，可用信用的计数），并且如果确定发送将在接收缓冲器内产生溢出情况则能够预先节流通过任何虚拟信道进行的发送，其中所述可用缓冲器空间分配给通过被识别的虚拟信道进行的发送。

5 根据本发明的一个方面，如上所述，事务层 202 有选择地调用流控制来防止与虚拟信道相关联的接收缓冲器的溢出并且能够遵循排序规则。根据一个实现，由发送者使用处理层 202 的流控制机制以通过 EGIO 链路 112 来跟踪代理（接收者）中的可用队列/缓冲器空间。在这方面，与传统的流控制机制不同，发送者而非接收者负责确定何时接收者暂时不能通过 10 虚拟信道接收更多内容。如这里所使用的，流控制没有暗示请求已经到达它的最终完成器。

在 EGIO 体系结构中，流控制与数据完整性机制相互独立，其中所述数据完整性机制用于实现发送者和接收者之间的可靠信息交换。即，流控制能够保证从发送者到接收者的事务层分组（TLP）信息流完好，这是由于数据完整性机制（下文讨论）保证通过重新传输改正错误的和丢失的 TLP。如这里所使用的，事务层的流控制机制包括 EGIO 链路 112 的虚拟信道。在这方面，将在由接收者通告的流控制信用（FCC）中反映由接收者支持的每个虚拟信道。

根据一个示例性实现，由事务层 202 和数据链路层 204 合作来执行流 20 控制。即，使用数据链路层分组（DLLP）在 EGIO 链路 112 的两端之间（例如，以每个 VC 为基础）传输流控制信息，以由事务层 202 的流控制机制使用。为了方便描述流控制机制，区分出下列分组信息类型或流控制信用类型：

- (a) 公布请求头部 (PH)
- 25 (b) 公布请求数据 (PD)
- (c) 非公布请求头部 (NPH)
- (d) 非公布请求数据 (NPD)
- (e) 读取、写入和消息完成头部 (CPLH)
- (f) 读取和消息完成数据 (CPLD)

如上所述，预先流控制的 EGIO 实现中的测量单元是流控制信用 (FCC)。根据仅仅一个实现，对于数据，流控制信用是十六 (16) 字节。对于头部，流控制信用的单元是一个头部。如上所述，每个虚拟信道 5 都保持了独立流控制。因此，事务层 202 中的流控制机制为分组信息的每个前述类型（如上所述的 (a) - (f)）以每个 VC 为基础来维持并跟踪信用的单独的指示符。根据所图示的示例性实现，分组的发送根据下述内容来消耗流控制信用：

- 存储器/IO/配置读取请求：1NPH 单元
- 存储器写入请求：1PH+nPD 单元（其中 n 与数据有效载荷的大小相关联，例如由流控制单元大小（例如，16 字节）划分的数据的长度）
- IO/配置写入请求：1NPH+1NPD
- 消息请求：取决于消息，至少 1PH 和/或 1NPH 单元
- 带有数据的完成：1CPLH+nCPLD 单元（其中 n 与由诸如 16 字节的流控制数据单元大小划分的数据大小有关）
- 没有数据的完成：1CPLH

对于所跟踪的每种类型的信息，有三个概念寄存器来监测消耗的信用（发送者内）、信用极限（发送者内）和分配的信用（接收者内），每个 20 概念寄存器有八 (8) 比特宽。信用消耗寄存器含有自从初始化以来所消耗的流控制单元的例如模 256 的总量的计数。已经引入了流控制机制的体系元件，参考图 6，示出了初始化和操作的示例性方法。

图 6 是根据本发明的仅仅一个示例性实施例的 EGIO 体系结构的流控制机制的示例性操作方法的流程图。根据图 6 所图示的示例性实现，方法 25 从方框 602 开始，其中当硬件初始化或复位时，初始化这里所描述的与至少一个初始虚拟信道相关联的流控制机制。根据一个示例性实现，当初始化 EGIO 元件的 EGIO 接口 106 的数据链路层 204 时，初始化与 VC0（例如，用于大量 (bulk) 通信的缺省虚拟信道）相关联的流控制机制。

在方框 604 中，事务层 202 的流控制机制更新一个或多个流控制寄存

器的参数。即，在初始化时信用消耗寄存器被设定为全零（0），并且当事务层承诺发送信息到数据链路层时增加。增加的大小与承诺发送的信息消耗的信用数量有关。根据一个实现，当达到或超过最大计数（例如，全 1）时，计数器翻转为零。根据一个实现，使用无符号 8 比特模算术来维持计数器。

在发送者中保持的信用极限寄存器含有可能消耗的流控制单元的最大数值的极限。在接口初始化（例如，启动、复位等）后，信用极限寄存器设定为全零，并且随后在接收消息后被更新以反映在流控制更新消息（上文进行了描述）中指示的值。

在接收者中保持的信用分配寄存器保持了自从初始化以来授与发送者的信用总数的计数。根据接收者的缓冲器大小和分配策略来初始设定该计数。该值可以包括在流控制更新消息中。

在方框 606 中，EGIO 接口 106 确定是否需要额外的虚拟信道，即除缺省 VC0 之外。如果是这样，随着建立这些额外 VC，则在方框 608 中事务层初始化与这些 VC 相关联的流控制机制，进而更新（多个）流控制寄存器。

如上所述，当初始化与虚拟信道相关联的流控制机制时，值随着接收者事务层从它的接收缓冲器移除已处理的信息而增加。增加的大小与产生可用空间的大小有关。根据一个实施例，接收者一般会将分配的信用最初设定为等于或大于下列值的值：

- PH: 1 流控制单元 (FCU)；
- PD: FCU 等于设备最大有效载荷大小的极大可能设定
- NPH: 1 FCU
- NPD: FCU 等于设备最大有效载荷大小的极大可能设定
- 交换设备—CPLH: 1 FCU
- 交换设备—CPLD: FCU 等于设备最大有效载荷大小的极大可能设定和设备将产生的极大读取请求中的较小的一个。
- 根和端点设备—CPLH 或 CPLD: 255 FCU (全 1)，发送者认为该值无穷大，因而其从不会阻塞。

根据这样的实现，接收者一般不会为任何消息类型而将信用分配寄存器值设定为大于 127 FCU。

- 根据另一个实现，与上述使用计数器方法保持信用分配寄存器不同，  
5 接收者（或发送者）可以基于下述等式动态计算可用的信用：

$$C_A = (\text{最近接收的发送的信用单元数值}) + (\text{可用的接收缓冲器空间})$$

[7]

- 如上所述，发送者将为发送者将使用的每个虚拟信道实现概念寄存器（消耗的信用，信用极限）。类似地，接收者为接收者支持的每个虚拟信  
10 道实现概念寄存器。一旦为适合的 VC 建立了（多个）流控制寄存器，随着过程前进到方框 610，EGIO 接口 106 就准备好参与 EGIO 通信。

- 在方框 610 中，发送者中的 EGIO 接口 106 接收数据报用于沿着 VC 发送。在方框 612 中，在发送所接收的数据报之前，用来发送数据报通过 EGIO 链路的 EGIO 元件的事务层 202 中的流控制机制证实该发送不会导致  
15 接收者处的溢出情况。根据一个示例性实现，事务层 202 的流控制机制基于或至少部分基于使用可用寄存器以及发送数据报将消耗的信用数量来作出这个确认。

- 为了预先制止如果这样做将引起接收缓冲器溢出的信息的发送，如果消耗的信用的计数加上与将要发送的数据相关的信用单元的数目，小于或  
20 等于信用单元值，则允许发送者发送一类信息，即：

$$\text{Cred\_Req} = (\text{Cred_Consumed} + \langle \text{Info_cred} \rangle) \bmod 2^{[\text{字段大小}]}$$

[8]

其中字段大小对于 PH、NPH、CLPH 等于八（8），对于 PD、NPD 和 CPLD 等于十二（12）。

- 25 当发送者接收指示非无穷信用（即，<255 FCU>）的完成的流控制信息时，发送者将根据可用信用来节流完成。当考虑信用使用及返回时，来自不同事务的信息不混合在一个信用中。类似地，当考虑信用使用及返回时，来自一个事务的头部和数据信息也不混合在一个使用中。因此，当某个分组由于缺乏流控制信用而被阻塞传输时，发送者在确定应当准许哪个类型

的分组绕过“停滞”分组时将遵循排序规则（上文）。

如果在方框 612 中流控制机制确定接收者没有合适的缓冲器空间来接收数据报，则流控制机制暂时中止沿相关虚拟信道的发送，直到发送者的（多个）流控制寄存器进行了更新以准许该发送，如方框 614 所示。根据 5 一个示例性实施例，通过流控制更新消息来接收更新，下文将对此进行详细描述。

如果在方框 612 中，流控制机制推断出数据报的发送不会导致接收者处的溢出情况，则 EGIO 接口 106 开始发送数据报，如方框 616 所示。如上所述，数据报的发送涉及事务层 202、数据链路层 204 和/或物理层 206 10 处的处理步骤（例如，添加头部、数据完整性信息等等）。

根据一个实施例，响应于通过虚拟信道的数据报的接收，接收者中的流控制机制将发布流控制更新。该更新可以是确认分组中的头部形式等等。在这样的实施例中，事务的流控制信用的返回不认为是意味着事务已经完成或事务已经实现系统可见（visibility）。使用存储器写入请求语义 15 的消息信号中断（MSI）象任何其它存储器写入一样被处理。如果随后的 FC 更新消息（来自接收者）指示了比最初指示的值更低的信用极限值，则发送者应当承认新的较低极限，并且提供一个消息错误。

根据这里所描述的流控制机制，如果接收者接收到比分配的信用更多的信息（超过分配的信用），则接收者将向违规的发送者指示接收者溢出 20 错误，并且对引起溢出的分组发起数据链路级别的重试请求。

在方框 618 中，在接收到流控制更新信息之后，与发送者中特定虚拟信道有关的流控制机制进而更新（多个）流控制寄存器以助于随后的流控制。

上面已经介绍了体系结构元件和示例性操作细节，示出了用来传输流 25 控制信息的示例性协议。根据一个示例性实施例，使用流控制分组在数据链路层 204 上传输流控制信息。

- 流控制分组（FCP）

根据一个实现，使用流控制分组（FCP）在设备之间传输保持上述寄

存器所需的流控制信息。参考图 9，示出了示例性流控制分组。根据一个实施例，流控制分组 900 包括用于具体虚拟信道的关于六个信用寄存器的状态的输送信息和 2-DW 头部格式，其中六个信用寄存器由接收事务层的流控制逻辑为每个 VC 保持。

5 根据本发明的教导的一个实施例，如图 9 所示有两种类型的 FCP：初始 FCP 和更新 FCP。如上所述，在初始化事务层时，发布初始 FCP 902。在初始化事务层之后，更新 FCP 904 被用于更新寄存器中的信息。

10 在常规操作期间接收到初始 FCP 902 引起本地流控制机制的复位以及初始 FCP 902 的发送。初始 FCP 902 的内容包括为 PH、PD、NPH、NPD、CPHL、CPHD 和信道 ID（例如，与应用 FC 信息相关联的虚拟信道）中的每个所通告的信用的至少一个子集。

更新 FCP 904 的格式与初始 FCP 902 的格式类似。应当知道，尽管 FC 头部不包括其它事务层分组头部格式普遍具有的长度字段，但是分组的大小是明确的，因为没有与该分组相关的额外 DW 数据。

15

### 错误转发

与传统的错误转发机制不同，EGIO 体系结构依靠附加到被识别为由于如下描述的多个原因而且具有的缺陷的（多个）数据报上的尾部信息。

20 根据一个示例性实现，事务层 202 使用了多种公知错误检测技术中的任何一种，例如循环冗余校验（CRC）错误控制等等。

根据一个实现，为了有助于错误转发特征，EGIO 体系结构使用了“尾部”，其附加到携带已知坏数据的 TLP 上。可能使用尾部错误转发的情况示例包括：

示例 #1：来自主存储器的读取遇到无法纠正的 ECC 错误

25

示例 #2：向主存储器的 PCI 写入的奇偶错误

示例 #3：内部数据缓冲器或高速缓存中的数据完整性错误

根据一个示例性实现，错误转发仅用于读取完成数据或写入数据。即，对于与数据报相关的管理开销中发生错误的情形，例如头部中的错误（例如请求阶段、地址/命令等等），一般不使用错误转发。如这里所使用

的，具有头部错误的请求/完成通常不能被转发，这是由于不能确定地识别真实目的地，并且因此该错误转发可能引起直接或间接影响，例如数据损坏、系统故障等等。根据一个实施例，错误转发用于传播错误通过系统以及系统诊断。错误转发不使用数据链路层重试，因此只有在 EGIO 链路 5 112 上出现如 TLP 错误检测机制（例如，循环冗余校验（CRC）等等）所确定的发送错误时，才重试以尾部结束的 TLP。因此尾部可能最终引起请求的发起者重新发布它（在上述的事务层）或者采取某个其它的动作。

如这里所使用的，所有 EGIO 接收者（例如，位于 EGIO 接口 106 中）都能够处理以尾部结束的 TLP。在发送者中对加入尾部的支持是可选的（因而与老式设备兼容）。交换器 108 对尾部和 TLP 的其余部分一起进行路由。具有对等（peer）路由支持的根复合体 104 一般会一起路由尾部和 TLP 的其余部分，但不是必需如此。错误转发一般适用于写入请求（公布的或非公布的）或读取完成中的数据。发送者知道的含有坏数据的 TLP 应当以尾部结束。

15 根据一个示例性实现，尾部由 2 DW 组成，其中字节[7:5]是全零（例如，000），并且比特[4:1]是全一（例如，1111），而预留所有其它比特。EGIO 接收者会认为以尾部结束的 TLP 中的所有数据都是损坏的。如果应用错误转发，则接收者将指定 TLP 的所有数据标记为坏（“中毒”）。在事务层中，分析器（parser）一般会分析到整个 TLP 的末端并 20 马上校验随后的数据，以了解数据是否结束。

## 数据链路层 204

如上所述，图 2 的数据链路层 204 充当事务层 202 和物理层 206 之间的中间级（stage）。数据链路层 204 的主要责任是提供用于通过 EGIO 链路 25 112 在两个组件之间交换事务层分组（TLP）的可靠机制。数据链路层 204 的发送方接收由事务层 202 装配的 TLP、应用分组序列标识符（例如，标识号码）、计算并应用错误检测代码（例如，CRC 代码）并且向物理层 206 递交修改的 TLP，用于通过挑选的一条或多条在 EGIO 链路 112 的带宽中建立的虚拟信道而进行传输。

接收数据链路层 204 负责校验所接收 TLP 的完整性（例如，使用 CRC 机制等等），并且负责向事务层 204 递交完整性校验是肯定的那些 TLP，以用于在转发到设备核心之前进行分解。由数据链路层 204 提供的服务通常包括数据交换、错误检测与重试、初始化与电源管理服务，以及

5 数据链路层内部通信服务。基于前述分类提供的每种服务列举如下：

#### 数据交换服务

- 从发送事务层接受用于发送的 TLP
  - i. 接受通过链路从物理层接收的 TLP，并且将它们传输到接收事  
务层

#### 10 错误检测&重试

- TLP 序列号码与 CRC 生成
- 已发送 TLP 存储器，用于数据链路层重试
- 数据完整性校验
- 确认以及重试 DLLP

#### 15 - 记录机制和错误报告的错误指示

- i. 链路 Ack 超时定时器

#### 初始化与电源管理服务

- 跟踪链路状态并能够传输有效/复位/断开连接状态到事务层

#### 数据链路层内部通信服务

#### 20 - 用于包括错误检测以及重试的链路管理功能

- 在两个直接相连的组件的数据链路层之间进行传输
- 没有暴露给事务层

如在 EGPIO 接口 106 中所使用的，数据链路层 204 对于事务层 202 表现为具有不同延迟的信息导管（conduit）。馈送到发送数据链路层的所有信息在较晚的时间处将出现在接收数据链路层的输出端。延迟将取决于许多因素，包括管道延迟、链路 112 的宽度和操作频率、通过介质的通信信号的发送、以及由数据链路层重试引起的延时。由于这些延时，发送数据链路层可以向发送事务层 202 施加反压力（backpressure），并且接收数据链路层将有效信息的存在和缺失传输到接收事务层 202。

根据一个实现，数据链路层 204 跟踪 EGIO 链路 112 的状态。在这方面，DLL 204 与事务 202 和物理层 206 传输链路状态，并且通过物理层 206 执行链路管理。根据一个实现，数据链路层含有链路控制与管理状态机来执行这样的管理任务，参考图 11 图示了所述状态机的一个示例。根据图 11 的示例性实现，链路控制与管理状态机的状态 1100 定义如下：

### 示例性 DLL 链路状态

- LinkDown（链路停用）（LD）—物理层报告链路是不可操作的，或者没有连接端口
  - LinkInit（链路初始化）（LI）—物理层报告链路是可操作的并且正在初始化
  - LinkActive（链路有效）（LA）—常规操作模式
  - LinkActDefer（链路动作延期）（LAD）—常规操作中断，物理层试图恢复
- 每个状态的对应管理规则：
- LinkDown（LD）
    - 跟随在组件复位之后的初始状态
    - 在进入 LD 后
      - 将所有数据链路层状态信息复位成缺省值
    - 在 LD 中时
      - 不和事务层或物理层交换 TLP 信息
      - 不和物理层交换 DLLP 信息
      - 不产生或接受 DLLP
    - 退出转入 LI，如果：
      - 来自事务层的指示是链路没有被 SW 禁用
  - LinkInit（LI）
    - 在 LI 中时
      - 不和事务层或物理层交换 TLP 信息
      - 不和物理层交换 DLLP 信息

- 不产生或接受 DLLP
- 退出转入 LA, 如果:
- 来自物理层的指示是链路训练 (training) 成功
- 退出转入 LD, 如果:
- 来自物理层的指示是链路训练 (training) 失败
- 5 • LinkActive (LA)
- 在 LinkActive 中时:
- 和事务层与物理层交换 TLP 信息
  - 和物理层交换 DLLP 信息
  - 产生并接受 DLLP。
- 10 退出进入 LinkActDefer, 如果:
- 来自数据链路层重试管理机制的指示是需要链路的重新训练,  
或者如果物理层报告重新训练正在进行中。
- LinkActDefer (LAD)
- 15 在 LinkActDefer 中时
- 不和事务层或物理层交换 TLP 信息
  - 不和物理层交换 DLLP 信息
  - 不产生或接受 DLLP
- 退出进入 LinkActive, 如果:
- 来自物理层的指示是重新训练成功
- 20 退出进入 LinkDown, 如果:
- 来自物理层的指示是重新训练失败
- 25 数据完整性管理
- 如这里所使用的, 数据链路层分组 (DLLP) 被用于支持 EGIO 链路数据完整性机制。在这方面, 根据一个实现, EGIO 体系结构规定了下列 DLLP 来支持链路完整性管理:
- Ack DLLP: TLP 序列号码确认一用于指示成功接收了某些数量的 TLP

- Nak DLLP: TLP 序列号码否定确认—用于指示数据链路层重试
- Ack 超时 DLLP: 指示最近发送的序列号码—用于检测某些形式的 TLP 丢失

如上所述，事务层 202 向数据链路层 204 提供 TLP 边界信息，使得  
5 DLL 204 能够将序列号码和循环冗余校验（CRC）错误检测应用于 TLP。  
根据一个示例性实现，接收数据链路层通过校验序列号码、CRC 代码和来自接收物理层的任何错误指示来验证接收的 TLP。如果 TLP 中有错误，则使用数据链路层重试来恢复。尽管这里的描述使用了 CRC，本领域技术人员应当理解也可以使用其它形式的错误检测，例如数据报内容的哈希散列  
10 （hash）等等。

#### CRC、序列号码以及重试管理（发送者）

在概念“计数器”和“标志”方面，以下描述了用于确定 TLP、CRC 和序列号码以支持数据链路层重试的机制：

##### 15 CRC 与序列号码规则（发送者）

- 使用下列 8 比特计数器：
  - TRANS\_SEQ—存储应用于正在准备发送的 TLP 的序列号码
    - . 在 LinkDown 状态下设定为全 ‘0’
    - . 在每个 TLP 发送后，增加 1
    - . 当全 ‘1’ 时，增加引起翻转使得全 ‘0’
      - . Nak DLLP 的接收引起值被重新设定为 Nak DLLP 中指示的序列号码
  - ACKD\_SEQ—存储在最近接收的链路到链路确认 DLLP 中确认的序列号码。
    - . 在 LinkDown 状态下设定为全 ‘1’
- 每个 TLP 被分配 8 比特序列号码
  - 计数器 TRANS\_SEQ 存储这个号码
  - 如果 TRANS\_SEQ 等于 (ACKD\_SEQ - 1) 模 256，则发送者一般不会发送另一 TLP，直到 Ack DLLP 更新 ACKD\_SEQ，使得条件

(TRANS\_SEQ==ACKD\_SEQ-1) 模 256 不再正确。

- TRANS\_SEQ 应用于 TLP, 通过:

- 为 TLP 预先准备(prepend)单个字节值
- 为 TLP 预先准备单个预留字节

5 • 使用下述算法为 TLP 计算 32b CRC, 并将其附加在 TLP 末端

- 使用的多项式是 0x04C11DB7
  - 与以太网使用的相同的 CRC-32

- 计算过程是:

1) CRC-32 计算的初始值是通过为序列号码预先准备 24 个 ‘0’ 而形成的 DW

10 2) 以从包括头部的字节 0 的 DW 到 TLP 的最后 DW 的顺序, 使用来自事务层的 TLP 的每个 DW 而继续 CRC 计算

3) 取来自计算的比特序列的补码, 结果是 TLP CRC

4) CRC DW 附加在 TLP 的末端

15 • 已发送 TLP 的拷贝一般会存储在数据链路层重试缓冲器中

- 当从其它设备接收到 Ack DLLP 时:

- ACKD\_SEQ 装入在 DLLP 中指定的值

- 重试缓冲器清除序列号码在下述范围内的 TLP:

. 从 ACKD\_SEQ 的先前值+1

20 . 到 ACKD\_SEQ 的新值

- 当从链路上的其它组件接收到 Nak DLLP 时:

- 如果正在向物理层传输 TLP, 则继续该传输直到该 TLP 的传输完成

- 不从事务层获得另外的 TLP, 直到完成了下述步骤

- 重试缓冲器清除序列号码在下述范围内的 TLP:

. 从 ACKD\_SEQ 的先前值+1

. 到在 Nak DLLP 的 Nak 序列号码字段中指定的值

- 重试缓冲器中所有剩余的 TLP 都重新提交到物理层, 用于以原始顺序重新发送

. 注意: 这将包括序列号码在下述范围内的所有 TLP:

- o 在 Nak DLLP 的 Nak 序列号码字段中指定的值+1
  - o TRANS\_SEQ 的值-1
    - . 如果在重试缓冲器中没有剩余的 TLP，则 Nak DLLP 错误
  - o 根据错误跟踪和记录部分，一般会报告错误的 Nak DLLP
- 5 o 发送者不需要其它的动作

#### CRC 与序列号码（接收者）

类似地，在概念“计数器”和“标志”方面，以下描述了用于确定 TLP、CRC 和序列号码以支持数据链路层重试的机制：

- 10 . 使用下列 8 比特计数器：
  - o NEXT\_RCV\_SEQ—为下一个 TLP 存储期望的序列号码
    - . 在 LinkDown 状态下设定为全‘0’
    - . 对于接受的每个 TLP，增加 1，或者当通过接受 TLP 而清除 DLLR\_IN\_PROGRESS 标志（下文描述）时
  - . 每次接收到链路层 DLLP 并且 DLLR\_IN\_PROGRESS 标志被清除时，装入值 (Trans. Seq. Num+1)
- 15 o 如果 NEXT\_RCV\_SEQ 的值与已接收的 TLP 或 Ack 超时 DLLP 指定的值不同，则指示在发送者和接收者之间的序列号码同步丢失；在这种情况下：
  - . 如果设定了 DLLR\_IN\_PROGRESS 标志，则
    - o 复位 DLLR\_IN\_PROGRESS 标志
    - o 发送“发送坏 DLLR DLLP”错误到错误记录/跟踪
    - o 注意：这指示错误地发送了 DLLR DLLP (Nak)
  - . 如果没有设定 DLLR\_IN\_PROGRESS 标志，则
    - o 设定 DLLR\_IN\_PROGRESS 标志并且发起 Nak DLLP
    - o 注意：这指示 TLP 丢失

- 使用下述 3 比特计数器:
    - DLLRR\_COUNT一对在特定时间段内发布的 DLLR DLLP 的次数进行计数
      - . 在 LinkDown 状态下设定为 b'100
      - . 对于发布的每个 Nak DLLP, 增加 1
  - 当计数达到 b'100 时:
    - 链路控制状态机从 LinkActive 移动到 LinkActDefer
    - DLLRR\_COUNT 随后被复位为 b'000
  - 如果 DLLRR\_COUNT 不等于 b'000, 每 256 个符号时间减 1
    - 即, 在 b'000 饱和
  - 使用下述标志:
    - DLLR\_IN\_PROGESS
- 15     • 下面描述设定/清除条件
- 当设定了 DLLR\_IN\_PROGESS 时, 丢弃所有已接收的 TLP (直到接收到由 DLLR\_DLLP 指示的 TLP)
  - 当 DLLR\_IN\_PROGESS 是清空的时, 如下所述校验已接收的 TLP
  - 对于将要接受的 TLP, 下述条件一般应当为真:
- 20
  - 已接收的 TLP 序列号码等于 NEXT\_RCV\_SEQ
  - 物理层没有指示在 TLP 接收过程中的任何错误
  - TLP CRC 校验不指示错误
- 当接受了 TLP 时:
    - TLP 的事务层部分被转发到接收事务层
    - 如果设定, 则清空 DLLR\_IN\_PROGESS 标志
    - 增加 NEXT\_RCV\_SEQ
  - 当没有接受 TLP 时:
    - 设定 DLLR\_IN\_PROGESS 标志

o 发送 Nak DLLP

- Ack/Nak 序列号码字段一般会包含值 (NEXT\_RCV\_SEQ-1)

- Nak 类型 (NT) 字段一般会指示 Nak 的原因

5        o b'00—由物理层识别的接收错误

      o b'01—TLP CRC 校验失败

      o b'10—序列号码不正确

      o b'11—由物理层识别的成帧错误

10      • 接收者一般不会允许从接收 TLP 的 CRC 到发送 Nak 的时间超过 1023 个符号时间，如从组件的端口所测量的那样。

      o 注意：没有增加 NEXT\_RCV\_SEQ

- 如果接收数据链路层没有接收到在其后的 512 符号时间内跟随着 Nak DLLP 的期望 TLP，则重复 Nak DLLP。

15      o 如果经过四次尝试后仍然没有接收到期望的 TLP，则接收者将：

      . 进入 LinkActDefer 状态，并启动由物理层进行的链路重新训练

      . 将主要错误的发生指示给错误跟踪与记录

- 当下列条件为真时，一般会发送数据链路层确认 DLLP：

20      o 数据链路控制与管理状态机处于 LinkActive 状态

      o 已经接受了 TLP，但还没有通过发送确认 DLLP 进行确认

      o 从最后的确认 DLLP 起已经经过了超过 512 个符号时间

- 可以比所需要的更频繁地发送数据链路层确认 DLLP

- 数据链路层确认 DLLP 在 Ack 序列 Num 字段内规定值 (NEXT\_RCV\_SEQ-1)

### Ack 超时机制

考虑 TLP 在链路 112 上被损坏使得接收者不能检测到 TLP 的存在的情况。当发送随后的 TLP 时将检测到丢失的 TLP，因为 TLP 序列号码与

接收者处的期望序列号码不匹配。然而发送数据链路层 204 通常不能限定下一 TLP 从发送传输层到在发送数据链路层 204 上面出现的时间。Ack 超时机制允许发送者限定接收者所需的检测丢失 TLP 的时间。

#### Ack 超时机制规则

- 5 . 如果发送重试缓冲器含有没有接收到 Ack DLLP 的 TLP,  
并且如果在超过 1024 个符号时间的时间段内没有发送 TLP 或链路  
DLLP, 则一般会发送 Ack 超时 DLLP。
  - . 在发送 Ack 超时 DLLP 之后, 数据链路层一般不会传送任何 TLP 到  
物理层用于发送, 直到从链路的另一方的组件接收到确认 DLLP。
- 10 o 如果在超过 1023 个符号时间的时间段内没有接收到确认 DLLP,  
则再次发送 Ack 超时 DLLP, 在第四次连续发送 Ack 超时 DLLP  
之后的 1024 个符号时间内仍没有接收到确认 DLLP, 进入  
LinkActDefer 状态并且启动由物理层进行的链路保持  
. 将主要错误的发生指示给错误跟踪与记录。

15

上文已经介绍了数据链路层 204 的数据完整性机制的体系结构上的元件以及协议元件, 参考图 7, 其中根据一个示例性实施例示出了数据完整性机制的示例性实现。

图 7 是根据本发明的一个示例性实施例的用于在 EGPIO 体系结构中监视数据完整性的示例性方法的流程图。根据图 7 所图示的示例性实现, 方法以方框 702 开始, 其中在 EGPIO 元件的 EGPIO 接口 106 处通过虚拟信道接收数据报。如上所述, 数据报在提升进入数据链路层 204 之前通过物理链路层 206 接收。根据某些实施例, 物理层 206 确定所接收的数据报是否符合分组成帧 (framing) 需求等。在某些实施例中, 丢弃未能满足这样的成帧需求的数据报, 而不会提升或由数据链路层 204 的数据完整性机制分析。如果证实了成帧, 物理层从数据报剥去成帧边界以显露数据链路层分组, 其被提升到数据链路层。

在方框 704 中, 在从物理层 206 接收到数据报之后, 在数据链路层 204 中证实数据链路层分组的完整性。如上所述, 数据链路层 204 的数据

完整性机制使用序列号码、CRC信息等等中的一个或多个来证实包括TLLP和其他事物的DLLP中的信息正确。

如果在方框704中数据链路层204识别了所接收DLLP的完整性中的缺陷，则数据链路层204调用上面提到的错误处理机制实例。

5 如果在方框704中，数据链路层204证实了所接收DLLP的完整性，则在方框708中所接收DLLP的至少一个子集被提升到事务层202。根据一个示例性实现，剥去针对数据链路层的信息（例如，头部、注脚（footer）等）以显露TLLP，其被传送到事务层用于进一步的处理。

## 10 物理层206

继续参考图2，示出了物理层206。如这里所使用的，物理层206使事务202和数据链路204与用于链路数据相互交换的信令技术相隔离。根据图2所图示的示例性实现，物理层划分为逻辑208和物理210功能子块。

15 如这里所使用的，逻辑子块208负责物理层206的“数字”功能。在这方面，逻辑子块204具有两个主要划分：发送部分，准备输出信息用于由物理子块210进行发送；以及接收者部分，用于在将所接收信息传送到链路层204之前识别并准备该信息。逻辑子块208和物理子块210通过状态与控制寄存器接口协调端口状态。由逻辑子块208指导物理层206的控制与管理功能。  
20

根据一个示例性实现，EGIO体系结构使用8b/10b发送代码。使用该方案，8比特字符被视为3比特和5比特，所述3比特和5比特各自映射到4比特代码组和6比特代码组。这些代码组被连接以形成10比特符号。EGIO体系结构使用的8b/10b编码方案提供了专用符号，其与用来表示字符的数据符号完全不同。这些专用符号用于下面的多种链路管理机制。专用符号还用于成帧DLLP和TLP，使用完全不同的专用符号允许快速便捷地区分这两类分组。

物理子块210包括发送者和接收者。逻辑子块208向发送者供应符号，发送者串行化这些符号并将其发送到链路112。链路112向接收者供

应串行化符号。接收者将所接收信号转换为比特流，比特流被解串行化，并且连同从输入串行流中恢复的符号时钟一起被供应道逻辑子块 208。应当理解，如这里所使用的，EGIO 链路 112 可以代表多种通信介质中的任何一种，包括：电通信链路、光通信链路、RF 通信链路、红外线通信链路、无线通信链路等等。在这方面，包括物理层 206 的物理子块 210 的（多个）发送者和/或（多个）接收者中的每一个都适合于一种或多种上述通信链路。

### 示例性通信代理

图 8 示出了含有与本发明相关联的特征的至少一个子集的示例性通信代理的方框图。根据本发明的一个示例性实现，根据图 8 所图示的示例性实现，所描述的通信代理 800 包括控制逻辑 802、EGIO 通信引擎 804、数据结构的存储器空间 806、以及可选的一个或多个应用 808。

如这里所使用的，控制逻辑 802 向 EGIO 通信引擎 804 的一个或多个元件中的每个提供处理资源以选择性地实现本发明的一个或多个方面。在这个方面，控制逻辑 802 被规定为代表微处理器、微控制器、有限状态机、可变逻辑器件、现场可编程门阵列、或当执行时使控制逻辑实现上述之一功能的内容中的一个或多个。

所描述的 EGIO 通信引擎 804 包括事务层接口 202、数据链路层接口 204、以及包括逻辑子块 208 和物理子块 210 以连接通信代理 800 和 EGIO 链路 112 的物理层接口 206 中的一个或多个。如这里所使用的，EGIO 通信引擎 804 的元件执行与上述的功能相同或类似的功能。

根据图 8 所图示的示例性实现，所描述的通信代理 800 包括数据结构 806。如下文参考图 10 将要详细介绍的，数据结构 806 可以包括存储器空间、IO 空间、配置空间和消息空间；所述空间由通信引擎 804 使用以便于 EGIO 体系结构的元件之间进行通信。

如这里所使用的，应用 808 被规定为代表由通信引擎 800 选择性调用的多种应用的任何一种，以实现 EGIO 通信协议和相关的管理功能。根据一个示例性实现，带宽管理器、流控制机制、数据完整性机制和对老式中

断的支持被实现为通信代理 800 中的可执行内容，所述可执行内容可由 EGIO 通信引擎 804 的一个或多个合适元件有选择地调用。

### 示例性（多种）数据结构

5 参考图 10，描述了（多个）EGIO 接口 106 使用的一种或多种数据结构的示图。根据本发明的一个实现，更具体地说，参考图 10 所图示的示例性实现，定义了四（4）个地址空间以在 EGIO 体系结构中使用：配置空间 1010、IO 空间 1020、存储器空间 1030 以及消息空间 1040。如所示，  
10 配置空间 1010 包括头部字段 1012，其包括定义了 EGIO 类别的信息，其中主设备（例如，端点、交换器、根复合体等等）属于该类别。这些地址空间的每个都执行他们如上所述的各自功能。

### 其它实施例

15 图 12 是根据本发明另一个实施例的其上存有多个指令的存储器介质的方框图，其中所述指令包括实现 EGIO 互连体系结构和通信协议的一个或多个方面的指令。

大体而言，图 12 图示了其上（中）存储有内容 1202 的机器可访问介质/设备 1200，所述内容包括以下内容的至少一个子集，即当访问机器执行所述内容时，所述内容实现了本发明的创新 EGIO 接口 106。如这里所使用的，机器可访问介质 1200 被规定为代表本领域技术人员公知的多种介质的任何一个，例如易失性存储器设备、非易失性存储器设备、磁存储介质、光存储介质、传播信号等等。类似地，可执行指令被规定为表达本领域公知的多种软件语言的任何一种，例如 C++、Visual Basic、超文本标记语言（HTML）、Java、可扩充标记语言（XML）等等。此外，应当  
20 认识到介质 1200 不需要与任何主系统共处在一起。即，介质 1200 可以位于远程服务器中，所述服务器可通信地耦合到执行系统并可由执行系统访问。因此，图 12 的软件实现应当认为是示例性的，因为另一个存储介质  
25 与软件实施例被认为位于本发明的精神和范围之内。

尽管以详细的描述以及对结构特征和/或方法步骤的专用语言的抽象描

述了本发明，但是应当理解，在所附权利要求中定义的本发明不必限制为所描述的具体特征或步骤。相反，所述具体特征和步骤只是作为实现所主张的发明的示例性形式而被公开。然而很明显，可以对其进行各种修改和变化而不会背离本发明较宽的精神和范围。因此本说明书和附图被认为是示例性的而不是限制性的。说明书和摘要没有被规定为是穷尽性的或是要将本发明限制为所公开的确定形式。

所附权利要求中使用的术语不应被解释成将本发明限制为说明书中公开的具体实施例。相反，所附权利要求完全确定了本发明的范围，其中根据已有权利要求解释原则来解释所述权利要求。

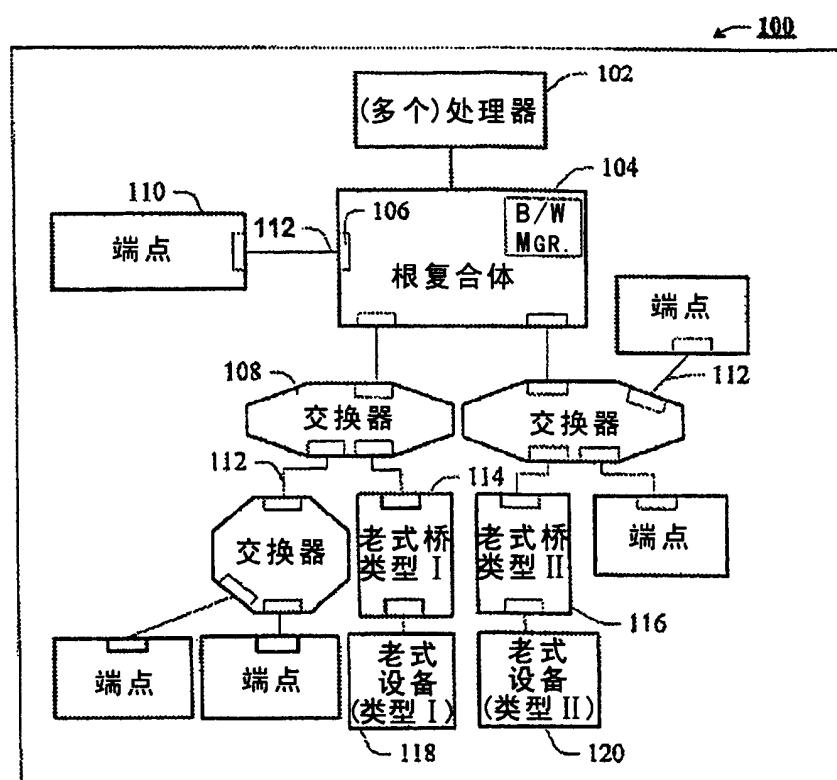


图1

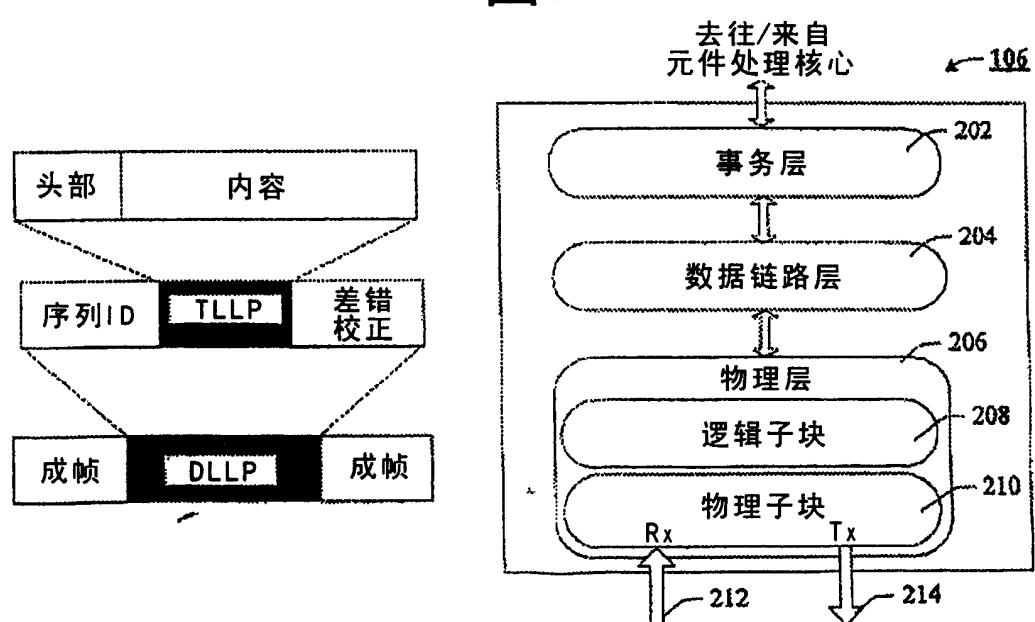


图2

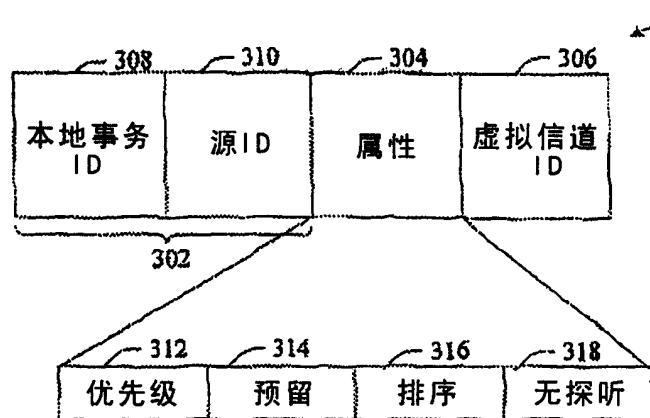


图3

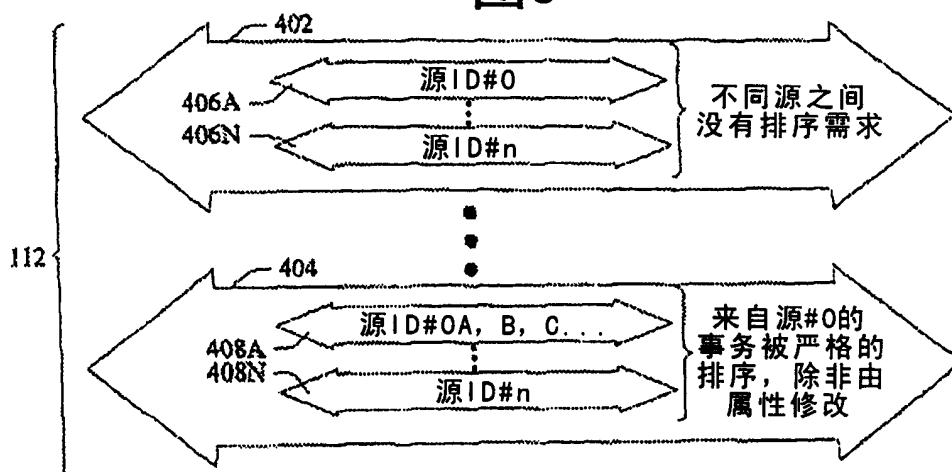


图4

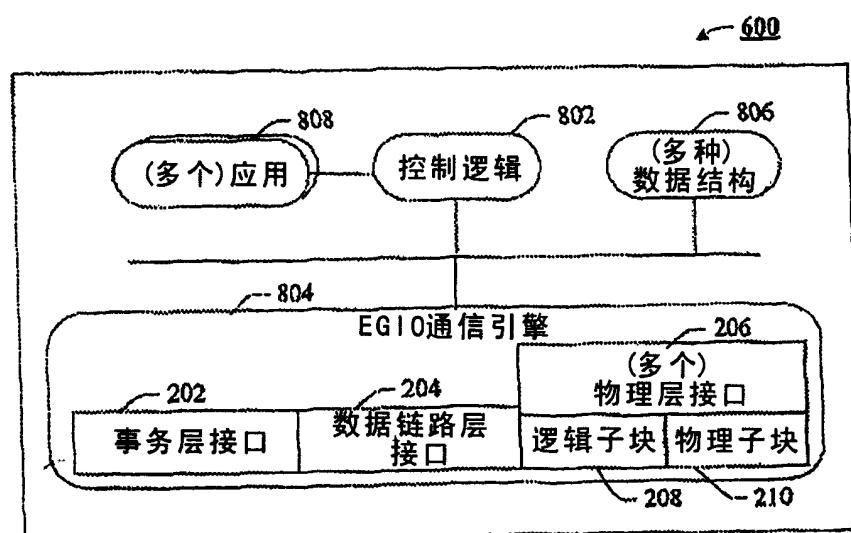


图8

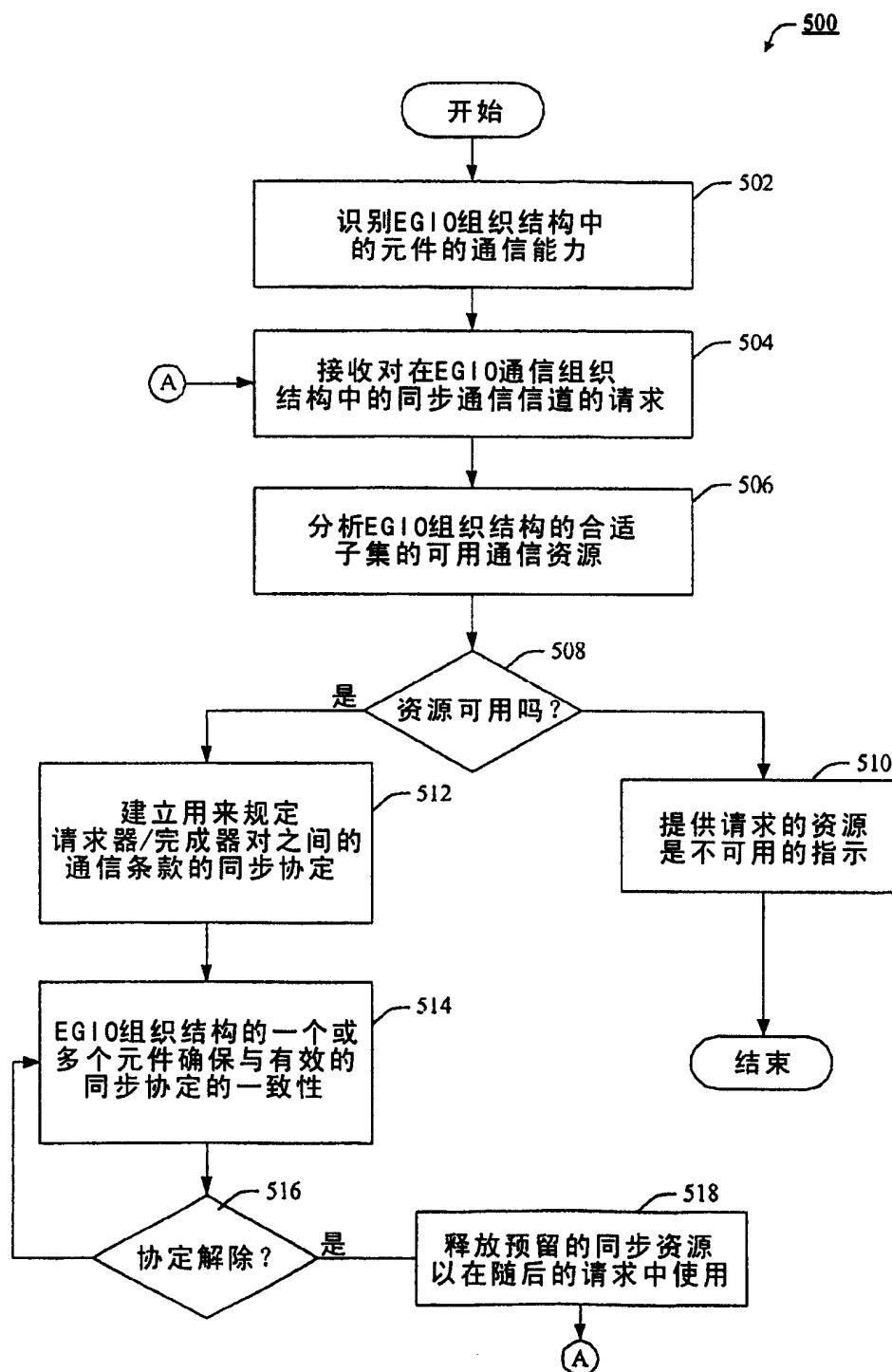


图5

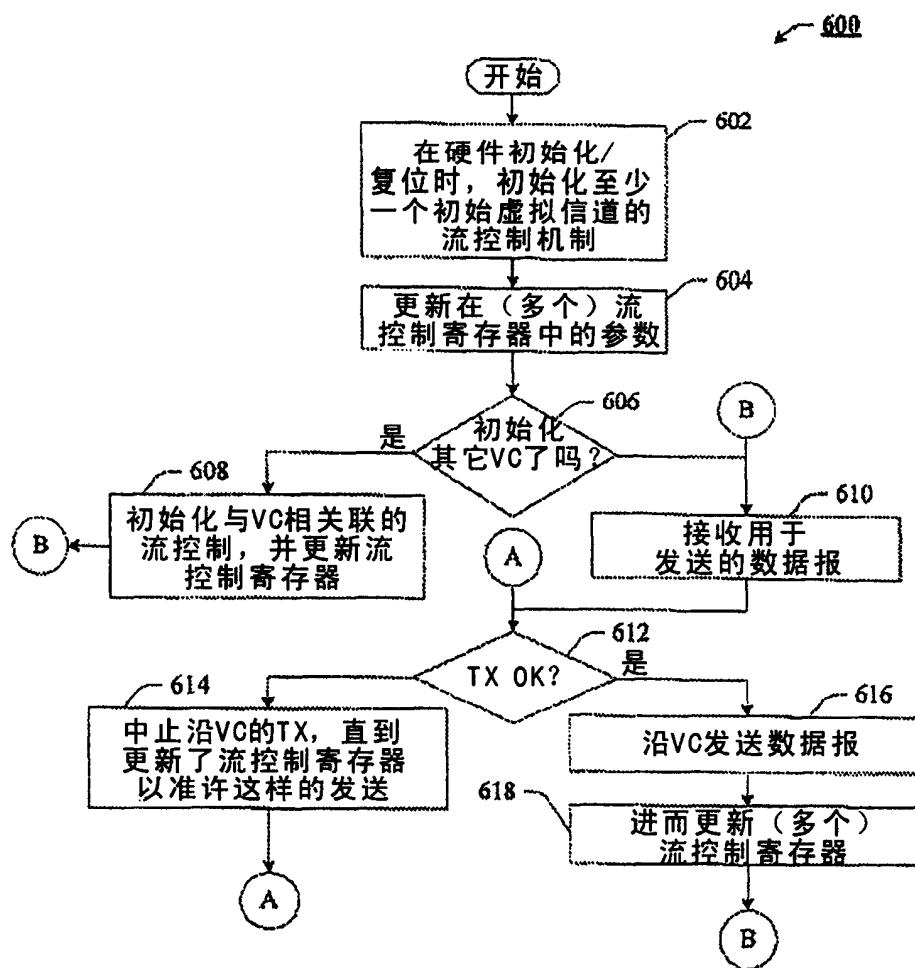


图6

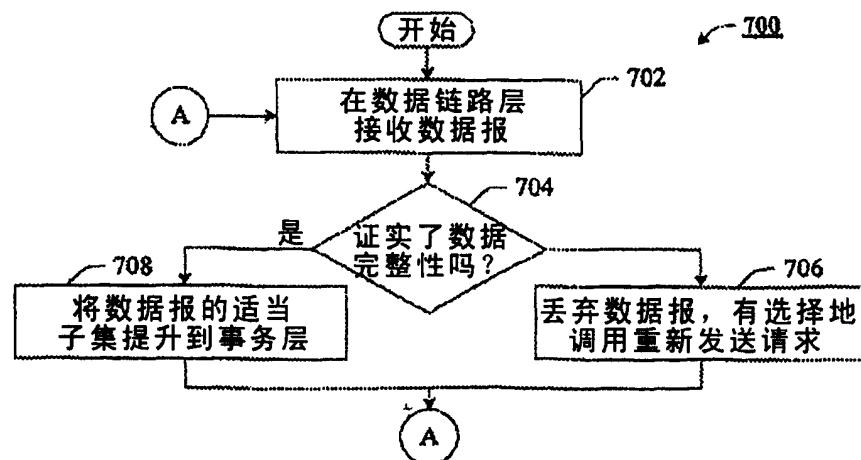


图7

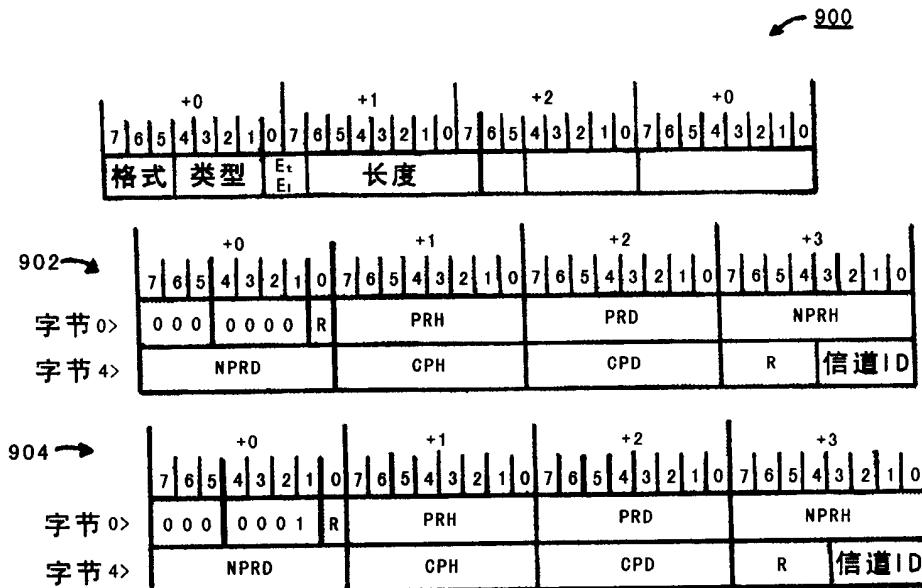


图9

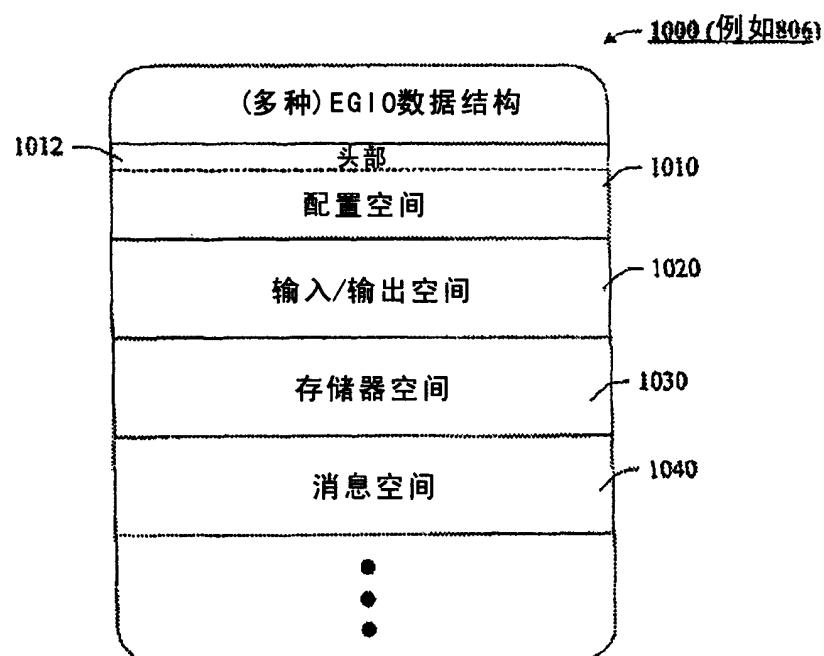


图10

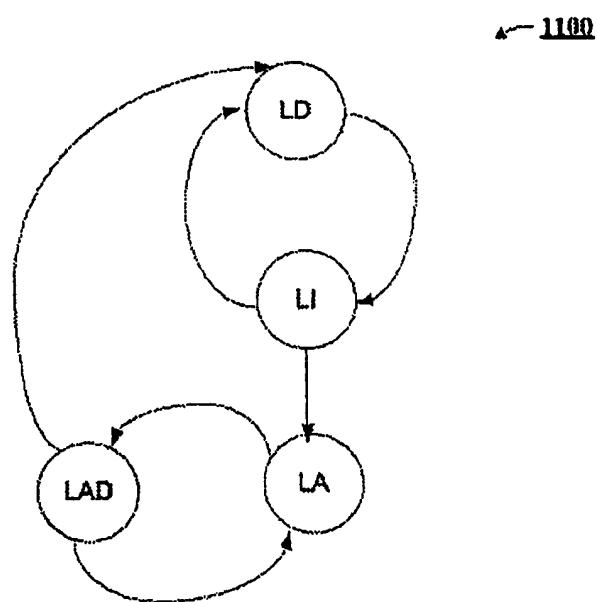


图 11

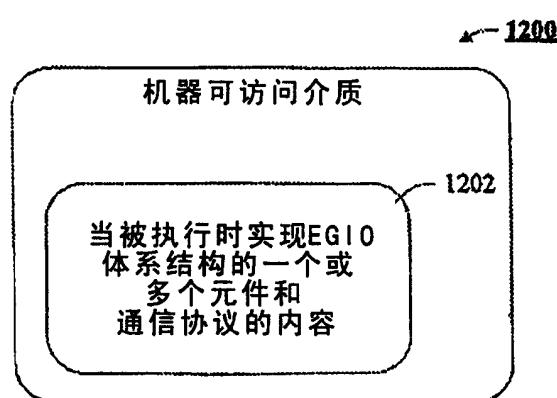


图 12