

[19] 中华人民共和国国家知识产权局

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



[12] 发明专利申请公布说明书

[21] 申请号 200610099597.5

[43] 公开日 2007 年 2 月 7 日

[11] 公开号 CN 1908924A

[22] 申请日 2002.11.7

[21] 申请号 200610099597.5

分案原申请号 02822453.1

[30] 优先权

[32] 2001.11.12 [33] US [31] 10/010,994

[71] 申请人 英特尔公司

地址 美国加利福尼亚州

[72] 发明人 兰迪·奥斯本

[74] 专利代理机构 北京东方亿思知识产权代理有限公司
代理人 王 怡

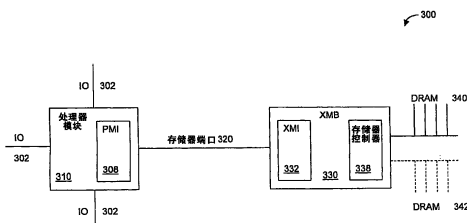
权利要求书 2 页 说明书 19 页 附图 19 页

[54] 发明名称

在存储器互连中进行读发起优化的方法和装置

[57] 摘要

本发明公开了一种方法和装置，用于通过在存储器互连中进行读发起优化来优化存储器读操作。在一个实施例中，可以在接收读请求迁徙期间，向存储器分派来自读请求迁徙的提前读请求。



1. 一种方法，包括：

接收写分组内的读请求迁徙；以及

5 在接收读请求迁徙期间，向存储器分派来自所述读请求迁徙的提前读请求。

2. 如权利要求 1 所述的方法，其中，向所述存储器分派所述提前读请求发生在接收所述读请求迁徙的前一半之后。

3. 如权利要求 2 所述的方法，其中，所述迁徙的前一半具有标识符，
10 用于将所述提前读请求向所述存储器进行提前分派。

4. 如权利要求 3 所述的方法，其中，所述用于将所述读请求进行提前分派的标识符是从以下组中选出的，该组由允许读提前分派和不允许读提前分派所组成。

5. 一种装置，包括：

15 具有输入、输出和控制的存储器系统；和

具有存储器输入、存储器输出和存储器控制的存储器端口，其中所述存储器系统输入被耦合以接收所述存储器端口存储器输出，所述存储器端口存储器输入被耦合以接收所述存储器系统输出，并且所述存储器系统控制被耦合到所述存储器端口存储器控制；

20 其中所述存储器端口具有存储器端口协议，该协议通过在接收读请求迁徙期间向存储器分派来自所述读请求迁徙的读请求，从而支持所述提前读请求，并且其中所述读请求迁徙位于写分组内。

6. 如权利要求 5 所述的装置，其中，所述提前读请求位于读请求迁徙的前一半内。

25 7. 一种系统，包括：

能够发出来自读请求迁徙的提前读请求的处理器，其中所述读请求迁徙位于写分组内；和

耦合到所述处理器的存储器设备，该存储器设备能够在所述读请求迁徙正被接收时接收所述提前读请求，并执行对存储器的提前读。

8. 如权利要求 7 所述的系统, 其中, 所述处理器提前读请求是在读请求迁徙分组的前一半内发出的。

9. 如权利要求 8 所述的系统, 其中, 所述读请求迁徙分组的前一半具有标识符, 用于将所述读请求向所述存储器设备进行提前分派, 所述标识符是从由允许读提前分派和不允许读提前分派所组成的组中选出的。

10. 一种装置, 包括:

用于发出来自读请求迁徙的提前读请求的装置, 其中所述读请求迁徙位于写分组内; 和

用于使存储器设备在所述读请求迁徙正被所述存储器设备接收时接收所述提前读请求并执行对存储器的提前读的装置。

11. 如权利要求 10 所述的装置, 其中, 用于发出提前读请求的装置是用于在读请求迁徙的前一半中设置标识符的装置。

12. 如权利要求 10 所述的装置, 其中, 用于执行对所述存储器的提前读的装置是由一个标识符控制的, 所述标识符是从由允许读提前分派和不允许读提前分派所组成的组中选出的。

13. 一种方法, 包括:

接收写分组内的读请求迁徙的前一半; 以及

在接收所述读请求迁徙的后一半之前, 向存储器分派来自所述读请求迁徙的读请求。

14. 如权利要求 13 所述的方法, 其中, 向所述存储器分派所述读请求发生在接收所述读请求迁徙的后一半之前。

15. 如权利要求 13 所述的方法, 其中, 向所述存储器分派所述读请求发生在对整个读请求迁徙进行错误校验之前。

在存储器互连中进行读发起优化的方法和装置

- 5 本申请是申请号为 02822453.1，申请日为 2002 年 11 月 7 日，题为“在存储器互连中进行读发起优化的方法和装置”的 PCT 中国国家阶段专利申请的分案申请。

技术领域

- 10 本发明涉及存储器（memory）连接。更具体而言，本发明涉及通过在存储器互连中进行读发起优化来优化存储器读操作的方法和装置。

背景技术

- 15 随着微处理器变得更快，对更快的存储器互连的需求也在增长。微处理器性能已经大大提高。系统性能总的来说未能跟上微处理器性能的提高，这是由于多方面的原因。一个原因是诸如硬盘驱动器之类的大容量存储设备的机械特性。

- 20 影响系统性能的另一个方面是微处理器外部的存储器。此存储器可能由外部高速缓存和外部主存两者组成，所述主存一般速度较慢，但大小较大。对外部存储器的访问可能具有延迟和实现的复杂性，这些影响了性能。降低延迟和降低实现复杂性的能力是有益的。对存储器互连进行优化的协议也是有益的。

发明内容

- 25 本发明的一个技术方案提供了一种方法，包括接收写分组内的读请求迁徙，以及在接收读请求迁徙期间，向存储器分派来自所述读请求迁徙的提前读请求。

 本发明的另一技术方案提供了一种装置，该装置包括具有输入、输出和控制的存储器系统，以及具有存储器输入、存储器输出和存储器控制的

存储器端口，其中所述存储器系统输入被耦合以接收所述存储器端口存储器输出，所述存储器端口存储器输入被耦合以接收所述存储器系统输出，并且所述存储器系统控制被耦合到所述存储器端口存储器控制；其中所述存储器端口具有存储器端口协议，该协议通过在接收读请求迁徙期间向存储器分派来自所述读请求迁徙的读请求，从而支持所述提前读请求，并且其中所述读请求迁徙位于写分组内。

本发明的又一技术方案提供了一种系统，该系统包括：能够发出来自读请求迁徙的提前读请求的处理器，其中所述读请求迁徙位于写分组内；还包括耦合到所述处理器的存储器设备，该存储器设备能够在所述读请求迁徙正被接收时接收所述提前读请求，并执行对存储器的提前读。

本发明的再一技术方案提供了一种装置，包括：用于发出来自读请求迁徙的提前读请求的装置，其中所述读请求迁徙位于写分组内；还包括用于使存储器设备在所述读请求迁徙正被所述存储器设备接收时接收所述提前读请求并执行对存储器的提前读的装置。

本发明的另外一个技术方案提供了一种方法，包括接收写分组内的读请求迁徙的前一半，以及在接收所述读请求迁徙的后一半之前，向存储器分派来自所述读请求迁徙的读请求。

附图说明

在附图中，示例性而非限制性地图示了本发明，其中相同的标号指示相似的元件，其中：

图 1 根据一个实施例，示出了可在其中实施本发明的网络环境；

图 2 根据一个实施例，示出了计算机系统的框图；

图 3 根据一个实施例，示出了存储器端口接口；

图 4A 和 4B 示出了读请求分组抢先于写请求分组的两个实施例；

图 5 根据一个实施例，示出了向存储器分派提前读请求；

图 6A 和 6B 示出了发送读返回头部（header）的两个实施例；

图 7 根据一个实施例，示出了存储器端口接口；

图 8 根据一个实施例，示出了存储器端口协议；

图 9 根据一个实施例，示出了需要 4 次传输的迁徙（flit）；

图 10 根据另一实施例，示出了迁徙；

图 11 根据一个实施例，示出了迁徙中的链路层与有效载荷的时间错开；

5 图 12 根据另一实施例，示出了存储器端口协议；

图 13 根据一个实施例，示出了存储器命令格式；

图 14 根据一个实施例，示出了设备命令格式；

图 15 根据一个实施例，示出了配置命令格式；

图 16 根据一个实施例，示出了传出链路层格式；

10 图 17 根据一个实施例，示出了公共传入链路层格式位；

图 18 根据一个实施例，示出了读返回头部格式；

图 19 根据一个实施例，示出了写确认格式；以及

图 20 根据一个实施例，示出了状态格式。

15 具体实施方式

本发明描述了通过在存储器互连中进行读发起优化来对存储器读操作进行优化的方法和装置。

在以下描述中，为说明目的，阐述了许多具体细节，以提供对本发明的透彻理解。然而对本领域技术人员来说，很显然没有这些具体细节也可以实施本发明。在某些情况下，公知的结构和设备以框图的形式示出而未详细示出，以免模糊了本发明。

图 1 示出了网络环境 100，所描述的技术可以应用在该网络环境中。如图所示，若干个处理器 104-1 到 104-P 以及存储器 106-1 到 106-P 通过网络 102 彼此相连，所述网络 102 例如是计算机总线。注意，网络 102 或者也可以是本地网络或包括本地网络。此处所描述的方法和装置本质上可以应用于任何类型的通信装置或设备，不管是本地的还是远程的。

图 2 以框图形式示出了计算机系统 200，在一个实施例中，可在所述计算机系统 200 中实施本发明。总线系统 202 将以下部件互连起来：中央处理单元（CPU）204、只读存储器（ROM）206、随机访问存储器

(RAM) 208、存储设备 (storage) 210、显示器 220、音频设备 222、键盘 224、指针设备 226、各种输入/输出 (I/O) 设备 228 和通信设备 230。所述总线系统 202 例如可以是下列总线中的一种或多种：系统总线、外围组件互连 (PCI)、高级图形端口 (AGP)、小型计算机系统接口 (SCSI)、电气与电子工程师协会 (IEEE) 标准号 1394 (火线)、通用串行总线 (USB)，等等。CPU 204 可以是单个、多个或者甚至是分布式的计算资源。此外，CPU 204 所例示的子系统可以具有到例如存储器之类的其它子系统的单独总线。一个这样的例子是用于图形的，例如用于 AGP。另一个例子是到 CPU 204 外部的存储器的存储器端口接口，所述存储器例如是 ROM 206、RAM 208，等等。

如下面所详细描述的那样，应该理解的是，能够加速操作和/或降低延迟和/或复杂性的存储器端口接口和/或协议是有益的。例如，在对存储器的读请求中，如下面所详细描述的那样，可以通过首先仅发送发起存储器读所需的信息，而在接收整个读请求之前开始实际的对存储器的读。例如，如果首先发送地址，则存储器子系统就可以开始读存储器内容。读请求可能只想要正被访问的字中的一个字节，然而，这个“仅需字节”的信息可能是在地址之后被发送，并且及时到达使得存储器子系统随后仅发送所请求的字节。这样，操作就可以并行和/或先于其它操作地进行。类似地，如果存储器子系统知道它已访问存储器并将在固定时间之后得到结果，则如下面所详细描述的那样，可以在实际数据之前发送一个读返回头部。这可以使接收数据的设备知道数据将要到来，从而做好准备。例如，读返回头部可以具有标识出将要到来的数据与哪个读请求相关联的信息，从而允许设备在所述数据到来之前就确定它要去往何处。通过在实际数据之前发送这个读返回头部信息，设备就有时间在数据实际到来之前确定其目的地。

图 3 以框图形式示出了存储器端口接口 300 的一个实施例。在此实施例中，存储器端口 320 是处理器（在处理器模块 310 内）及其本地存储器（340 和 342）之间的快速高引脚效率互连。存储器端口接口的协议层（有时称为存储器端口协议）独立于任何具体的存储器技术，从而将处理

器与存储器技术和未来的存储器发展问题隔离开来。所述协议层具有作为本发明主题的多种具体特征，用于降低存储器请求的延迟并且降低实现的复杂性等等。

通过例如 IO 302、存储器端口 320 等，处理器模块 310 可以具有到各种其它设备的接口。可以认为存储器端口 320 是从处理器模块 310 到 XMB（外部存储器桥）330 的专用接口，如图 3 所示。XMB 330 可以包含存储器控制器 338，并接口到本地存储器例如 DRAM 340 和 342 这两组。因此，XMB 330 封装了存储器相关的信息，例如存储器类型（技术和速度）、存储器结构（DRAM 大小、每个模块的 DRAM 数，以及通道数），以及存储器控制（定时、刷新和功率管理）。

存储器端口 320 可以由物理互连和协议这两者组成。所述物理互连可以由若干并行互连组成，或者可以被认为是若干并行互连。例如，可能有一个用于数据的主链路和另一个用于控制的链路层（有时称为边带（sideband））。所述协议可以由在物理互连上传送命令、数据、响应等组成。例如，存储器端口命令可以由以下部分组成：存储器读和写；设备读和写；以及配置读和写。此外，链路层可以具有诸如读返回头部、写确认、状态等等的传入（inbound）和传出（outbound）信息。

存储器端口流量可以由存储器、设备及配置读和写命令组成。为了使 XMB 330 的复杂性最低，对于一个实施例，处理器存储器接口（PMI）308 发起所有数据传输。为了进一步降低复杂性，对于一个实施例，存储器端口 320 不支持 XMB 330 所发起的数据传输。在一个实施例中，通过使存储器端口 320 不支持在 XMB 330 中或通过 XMB 330 的 I/O 或图形设备，降低了复杂性。可以通过使存储器端口 320 不支持任何一致性（coherency）流量，来实现复杂性的进一步降低。

在处理器模块 310 中，存储器端口 320 包括封装在 PMI 308 中的功能。PMI 308 可以包括诸如写缓冲器、流控制、错误处理，以及控制和状态寄存器等等。在 XMB 330 处，存储器端口 320 可以包括封装在外部存储器接口（XMI）332 中的这些功能。XMI 332 还可以包括写和读缓冲器、错误日志、控制和状态寄存器等等。

在一个实施例中，存储器端口 320 可以利用存储器流量只是存储器读和写这一知识，从而可以对存储器请求进行优化，使性能最优，同时使复杂性最低。本发明描述了存储器端口 320 中的两种这样的优化，它们涉及降低发起对存储器的读请求时的延迟；还描述了第三种优化，其涉及在存储器端口 320 中使用边带读返回头部。

图 4A 示出了读分组可以如何抢先于写分组。读分组可以包含读命令和要读取的地址或地址范围。类似地，写分组可以包含写命令、要写入的地址或地址范围，以及要写入的数据（写数据）。因为所传输的信息量和通信信道的宽度，读和/或写分组可能在多于一次传输中发送。因为写分组具有以要写入的数据形式出现的额外信息，可以看出，写分组可能比读分组需要更多次传输。

在 402 接收输入。在 404，确定在 402 所接收到的输入是不是读分组的一部分。如果它是读分组的一部分，则在 406 进行检查，看该读分组是否完整。如果所述读分组完整，则在 408 进行读，然后在 402 接收另一输入。如果在 406 处确定所述读分组不完整，则在 402 接收另一输入。

如果在 404 确定接收到的输入 402 不是读分组的一部分，则在 410，确定在 402 接收到的输入是不是写分组的一部分。如果接收到的输入 402 不是写分组的一部分，则在 416 执行某种其它操作，然后在 402 接收另一输入。如果在 410 确定接收到的输入 402 是写分组的一部分，则在 412 确定该写分组是否完整。如果所述写分组完整，则在 414 进行写，然后在 402 接收另一输入。如果在 410，确定在 402 接收到的输入不是写请求分组的一部分，则某种其它操作 416 将会发生，然后回到 402 以接收输入。

从而，读分组就可以抢先于写分组。对于一个实施例，由读命令和要读取的一个或多个地址所组成的读分组可以是单一传输或单一迁徙。写分组由单一传输组成，该单一传输具有写命令、要写入的一个或多个地址，以及可能有多数的其它带有相关写数据的传输。为描述方便起见，包含读命令和要读取的一个或多个地址的读分组被称为读请求。从而，读请求和读分组传递相同的信息。为描述方便起见，写分组中包含写命令和要写入的一个或多个地址的部分被称为写请求。写分组的数据部分被称为写数

据。从而，写分组包含写请求和写数据。

于是，例如假设在 402 接收到写请求。转到 404，不是读分组的一部分所以转到 410。在 410 是写分组的一部分，因此前进到 412。在 412，确定仅是写请求而还未接收到写数据，因此写分组不完整，因此随后回到
5 402。现在如果在 402 接收到读请求，则转到 404，在此确定所述读请求是读分组的一部分，因此前进到 406。在 406，对所述读分组是否完整进行判断。在此实施例中，我们已经将读请求定义为包含完整的读分组，因此在 408 进行读，然后回到 402。然后，在此示例中，（通过接收到写请求）所开始的写操作还未被完成，因此读请求已经抢先于写。

10 跟随图 4A 的流程图，应该理解到，读请求在任何地方都可能抢先于写请求，直到接收到最后的写数据为止。从而，例如如果写分组包含写请求和 4 次写数据的传输，则如果在写请求、第一写数据、第二写数据或第三写数据之后接收到读请求，读请求都会抢先于写请求。

在另一实施例中，读请求不包含完整的读分组。在这样一个实施例中，在 406 对完整读分组的检查可能不完整，在该种情况下，将会前进到
15 402 以接收另一输入。如果并且只要随后在 406 接收到完整的读分组，就会前进到 408 并进行读，然后返回到在 402 接收输入。

允许在写请求或写数据之后接收到的读请求抢先于较早接收到的写请求或写数据，这导致了在写之前执行读。这使得存储器与必须首先响应写
20 请求然后响应读请求时相比，更快地响应读请求。

应该理解到，发送读和/或写请求的代理可以防止排序的危险。也就是说，发送代理可能想要对如何发出读和写请求进行限制。例如，发送代理可以为了降低复杂性而决定一个读请求不应抢先于另一读请求。

图 4B 示出了本发明另一实施例，其中读分组可以抢先于写请求。在此示例中，我们仅考虑读和写分组。此外，图 4B 示出了这样一个实施
25 例，其中读请求可以抢先于写分组，但不能像图 4A 中可能发生的那样，另一写分组抢先于该写分组。在图 4B 中，在 452 接收输入。在 454 进行检查，看其是不是读分组。如果其是读分组，则在 456 进行读，然后我们返回到在 452 接收输入。如果在 452 接收到的输入不是读分组，则在 458

进行检查，看其是不是写分组。如果其不是写分组，则我们返回到在 452 接收输入。另一方面，如果在 458 所述分组是写分组，则我们进入标为 460-470 的操作“内部循环”。

这个内部循环考虑到了读分组优先，正如我们可以从以下操作中看出 5 的那样：在 460 接收输入，462 检查读分组，如果是则进行读 464，然后返回到接收输入 460。然而，回到外部循环（452-458）的唯一途径是当写分组完整 468 然后进行写 470 的时候。在从 458 进入内部循环时，在 466 进行检查以确定有没有写数据。如果没有写数据，则我们前进到在 460 接收输入。如果有写数据，则在 468 进行检查，看所述写分组是否完整。如 10 果所述写分组不完整，则我们前进到在 460 接收输入。然而，如果所述写分组完整，则在 470 进行写，然后我们前进到在 452 从外部循环接收输入。从而，图 4B 示出了这样一个实施例，其中仅允许一个写分组被（读分组中的）一个读请求或多个读请求（即，在 470 的写之前遍历循环 460、462、464 一次或多次）所抢先。

15 在以上讨论中，我们详细叙述了例如读请求如何包含读命令和要读取的一个或多个地址。此外，也可以传递其它信息，例如确保请求完整性的纠错位、控制信息等等。为方便讨论本发明，用术语“迁徙”来指示与一个实体或操作相关联的信息传输单位，一个迁徙可以由一个或多个传输组成。分组可以由一个或多个迁徙组成。因此，例如，一个读请求分组可以 20 由指示与该读请求相关联的所有信息的单一迁徙所组成。这个信息取决于具体实现方式而可能需要多于一次传输才能完全发送。例如，如果读请求迁徙包含 32 位的信息，而计算机总线能够实现 16 位传输，则需要两次传输来实现对所述读请求迁徙的传输。下面所描述的是迁徙和传输的更多细节和示例。

25 图 5 所示出的 500 是向存储器分派（dispatch）提前读请求。这里，在 502 接收输入。在 504 进行检查，看在 502 接收到的输入是不是读请求迁徙的一部分。如果是，则在 506 进行检查，看对读的提前分派是否可能。如果提前分派可能，则在 508 进行提前读分派，然后回到 502 以接收输入。如果提前分派不可能，则在 510 进行检查，看读迁徙是否完整。如果

完整，则在 512 进行读，然后回到 502 以接收输入。如果读迁徙不完整，则回到 502 以接收输入。如果在 504 确定在 502 接收到的输入不是读请求迁徙的一部分，则某种其它操作 514 将会发生，然后回到 502 以接收输入。

- 5 在接收读请求迁徙期间，可能向存储器提前分派读请求。从而，不必在向存储器分派提前读请求之前接收到整个读请求迁徙。不必等待接收到整个读请求迁徙，这就允许存储器比在发出的请求之前必须等待接收到整个读请求迁徙时更快地响应提前读请求。从而，如果一个迁徙的第一部分包含提前读请求，则可以在接收迁徙第二部分的同时访问存储器，而所述
- 10 第二部分可能具有对存储器内容的修饰符信息。这个修饰符信息可能影响最终发送何种存储器数据。

图 6A 所示出的 600 发送读返回头部 606。这里，在 602 接收输入。在 604 进行检查，看其是不是读请求。如果不是读请求，则可能进行其它操作 614，然后回到 602 以接收输入。如果在 604，确定在 602 接收到的输入是读请求，则发生两件事情。第一，在 606 发送读返回头部，然后回到 602 等待输入。第二，在 608 发起读，在 610 接收读结果，在 612 发送所述读结果，然后回到 602 以接收输入。从而，在 608 发起的存储器读和在 606 发送的读返回头部可以基本上同时发生，和/或读返回头部可以恰好先于读结果的发送。从而，不必在发送读返回头部 606 之前接收到整个读结果。不必等待整个读结果，这就允许存储器和/或系统比在发送读返回头部之前必须等待接收到整个读请求时更早地发送读返回头部 606。

15

20

图 6B 示出了发送读返回头部的另一实施例 656。这里，在 652 接收输入。在 654 进行检查，看其是不是读请求。如果不是读请求，则可能进行其它操作 664，然后回到 652 以接收输入。如果在 654 确定在 652 接收到的输入是读请求，则在 608 发起读。在这个点上，发生两件事情。第一，在 656 发送读返回头部，然后回到 652 等待输入。第二，在 660 接收读结果，在 662 发送读结果，然后返回 652 以接收输入。从而，例如，在 658 发起存储器读导致了在 656 发送读返回头部。这个头部是与接收读结果 660 和发送读结果 662 并行地发送的，因此读返回头部可能先于对读结果

25

的发送。从而，不必在发送读返回头部 656 之前接收到整个读结果。不必等待整个读结果，这就允许存储器和/或系统比在发送读返回头部之前必须等待接收到整个读请求时更早地发送读返回头部 656。

在（606 或 656 处所发送的）读返回头部的一个实施例中，读返回头部具有一个编码在其中的标识符，该标识符使该读返回头部与导致其被发送的读请求相关。这样，例如，PMI 308 可以通过读返回头部标识符来识别出它与哪个读请求（以及所产生的即将到来的数据）相关联。

应该理解的是，在 606 或 656 发送的读返回头部的实际传输路径尽管也可以在主数据路径中，但优选地是在边带中，并且优选地有一个已知的从数据开始的固定偏移量。也就是说，使用与数据路径并行的边带路径（链路层），其中所述边带路径在相关联的读数据之前运送读返回头部，这可以允许接收设备在接收数据之前就确定数据要去往何处。从而，如果读返回头部在数据之前发送，则可以实现延迟的降低。

图 7 示出了存储器端口接口 700 的一个实施例。在此实施例中，存储器端口是一个全双工互连（存储器端口链路 720 和 740）。在传出方向（离开处理器模块 710 和 PMI 708），存储器端口链路 720 由两个信息流组成。一个流是宽度为 w 的数据流，而第二个流是宽度为 x 的链路层流。如上，为讨论本发明，将相关联的数据和链路层信息的方便的编组称为迁徙。在传入方向（进入处理器模块 710 和 PMI 708），存储器端口链路 740 由两个信息流组成。一个流是宽度为 y 的数据流，而第二个流是宽度为 z 的链路层流。

例如，图 8 示出了存储器端口协议的一个实施例 800。在此实施例中，迁徙 802 由指示“空闲”数据 802a 和链路层（LL）信息 802b 所组成。以类似的方式，迁徙 804 到 816（通称 8XX）由各自的 8XXa 和 8XXb 部分所组成。此实施例示出的是一个迁徙有 80 位的实施例，其中数据总线宽度 w 是 18 位宽，而链路层总线宽度 x 是 2 位宽。所示出的实施例对于一个迁徙需要 4 次传输，有效载荷中有 72 位，而链路层中有 8 位。

例如，图 9 示出了需要 4 次传输的迁徙 900。这里，迁徙 910 具有 72 位的数据和/或命令（d/c）信息，以及 8 位的链路层（LL）信息。这里，

传输是每次 20 位地完成的, 910a-0 和 910b-0, 然后 910a-1 和 910b-1, 等等。尽管图 9 所示出的 d/c 和 LL 传输出现在同一时间帧中, 但并不要求如此。也就是说, 一个迁徙内与特定传输相关联的 LL 数据可以在时间上与各自的 d/c 传输错开。此外, 所述时间错开可以跨越迁徙的边界。

- 5 链路层 (LL) 例如可以具有控制位和循环冗余校验 (CRC) 位。所述控制位例如可以表示头部、尾部 (tail) 或空闲。

回来参照图 8, 应该注意的是, 写分组 818 通常仅包含写请求和相关联的数据。然而在本发明中, 如图 8 所示, 不需要这种约束。将写分组 818 图示为在其中具有写请求迁徙 804、关联数据迁徙 806、808、812、
10 814 和 816, 以及读请求分组迁徙 810。

图 10 示出了具有 96 位的迁徙的实施例 1000, 其中数据总线宽度 y 是 18 位宽, 而链路层总线宽度 z 是 4 位宽。此实施例对一个迁徙需要 4 次传输, 并且有 64 个数据位、8 个校验位, 链路层中有 16 位。每个迁徙 1002 到 1016 都具有有效载荷分组 10xxa 和相关联的链路层 (LL) 分组 10xxb。
15 图 10 可以表示对处理器模块的输入。链路层 (LL) 例如可以具有链路层控制位和 CRC 位。所述控制位例如可以是读头部、写头部、数据、带尾部的数据、空闲、写 ack (确认)、流控制, 等等。

还应该理解的是, 尽管 LL 控制与有效载荷相关联, 但它不必同时发送。例如, 如果处理器模块具有读数据很快将到达这样的预先信息, 它就可以提高性能。从而, 例如, LL 中的提前读头部可以提供这种指示。例如, 在图 10 中, 与迁徙数据 1006a 相关联的 LL 可以位于时间上较早的位置, 例如在位置 1004b。从而, 如图 11 所示, 诸如 1106 之类的迁徙可以包含位置 1106a 处的有效载荷和位置 1106b 处的 LL 控制。从而, 图 11 示出了迁徙 1106 中的链路层与有效载荷的时间错开。
20

25 (进入 PMI 308 的) 传入边带链路层 (LL) 还可以包含关于其它传输和/或事务的信息。例如, LL 可以包含用于确认传出写命令的写确认、用于传出流控制反馈的停止位、状态指示 (空闲或数据有效载荷, 等等) 等等。而且, 与一个迁徙相关联的 LL 可以传递多于一条信息, 例如, LL 编码可以指示空闲的有效载荷或者非空闲的有效载荷, 此外还可以指示是不

是读返回头部、写确认、停止指示、空闲控制等等。此外，应该理解的是，如上所述，这个信息可以在时间上移位。

将信息在时间上移位使得可以降低接口的复杂性。例如，将读返回头部置于读返回数据的频带内，这可能给 XMI 332 强加了额外的功能，因为

5 XMI 332 必须将头部和数据信息复用起来，并且 XMI 332 必须对进入的读返回数据进行缓冲，以将存储器读带宽与稍高些的传入存储器端口带宽进行速率匹配，所述稍高些的传入存储器端口带宽是补偿内嵌头部的带宽开销所必需的。这种额外功能就可能意味着额外的复杂性以及很可能产生的额外的延迟。相反，通过将读返回头部置于时间上移位了的边带中，就可

10 以使得 XMI 332 数据路径能工作在与存储器总线相同的频率上，从而允许 XMI 332 工作时在读返回数据路径上没有缓冲或复用。从而，使用与数据路径并行的边带路径（链路层），其中所述边带路径可能不相邻地在相关联的读数据之前运送读返回头部，这可以允许操纵和控制逻辑在接收数据之前工作。如果读返回头部可以在数据之前被发送，则可以实现延迟的降

15 低。头部可以先于读返回数据 0 个或更多个迁徙（读返回偏移量可以在初始化时确定）。提前读返回头部允许处理器将头部译码和数据操纵逻辑移出关键路径，并将进入的读返回数据直接驱动到处理器内的合适目的地。

读返回头部提前于数据而发送的时间（偏移量）可以固定在某个时间点和/或是动态的。例如，可以在系统初始化时确定固定延迟。或者，系统

20 可以在操作时确定此时间，并使其固定。另一实施例可以将所述偏移时间置于编码在读返回头部的数据当中。从而，本领域技术人员将会认识到，有多种可能的实施例。

在时间上将信息移位也使得可以降低 PMI 308 接口的复杂性。也就是说，如果 PMI 308 通过边带而在主数据路径上的数据之前发送信息，则其

25 就可以降低复杂性和/或延迟。本领域技术人员将会理解到，可以在传出方向和传入方向上实现降低延迟和/或复杂性。

图 12 示出了存储器端口协议的另一实施例 1200，其中数据和命令带宽是 36 位，而链路层是 4 位。空闲迁徙 1202、1204、1216、1220 和 1234 具有其各自的“a”部分（1202a、1204a、1216a、1220a 和 1234a）中的空

闲有效载荷，并具有其各自的“b”部分（1202b、1204b、1216b、1220b和 1234b）中的链路层（LL）控制码。普通的写分组 1240 仅具有写关联数据，例如写请求 1206 迁徙和写数据迁徙（1208-1214）。每个迁徙都具有有可能指示额外信息的 LL 控制。例如，数据 0、数据 1 和数据 2（分别是 5 1208a、1210a 和 1212a）可能在链路层（LL）控制中具有指示数据是写数据的信息（分别是 1208b、1210b 和 1212b）。LL 1214b 可以指示出 1214a 是写数据以及该写数据的尾部。LL 1206b 可以指示出关于写请求 1206a 命令的额外信息。

迁徙 1218 示出了未抢先于写分组的读分组。读分组迁徙 1218 具有相 10 关联的读请求 1218a，其在相关联的链路层 1218b 中具有读请求标识和可能的额外信息。或者，所述读请求标识和/或额外信息可以位于 1218a 和/或 1218b 中。

被读请求所优先的写分组示出于 1228。这里，具有写请求迁徙 1222 和写数据迁徙 1224、1226、1230 和 1232 的写请求分组在数据 1 迁徙 1226 15 和数据 2 迁徙 1230 之间具有读请求迁徙 1228。具有读请求 1228a 和相关联的链路层 1228b 的这个读请求迁徙 1228 已经抢先于写请求迁徙 1222。

如上所述，对于普通的写分组 1240，写请求迁徙 1222 以及相关联的写数据迁徙 1224、1226、1230 和 1232 具有“a”有效载荷和相关联的“b”链路层控制。

20 在下面的讨论中详述的是命令、字段位定义、链路层控制等的可能的实施例。应该理解的是，这些具体示例不应被认为是限制先前对本发明的讨论。这些示例仅用于说明性目的。

再次参照图 3，PMI 308 可以向存储器端口 320 发出存储器命令和配置命令。在此示例中，XMI 332 不通过存储器端口 320 发出任何命令。此 25 外，未示出部分（partial）存储器写。

有三种一般类型的命令：存储器、设备和配置。

存储器命令，例如读、写和取消，一般以作为纯粹的存储器而起作用的主存储器（例如 DRAM）为目标，所述主存储器连接到 XMB 330 和/或包含在 XMB 330 中。

设备命令，例如读和写，一般以不作为纯粹的存储器起作用的存储器位置为目标。例如，对映射到在其中写（以及有时）读有副作用的存储器空间中的闪存存储器和/或设备寄存器进行写入。在这种情况下，每个写命令可能想要在目标上只执行一次。而且，这种命令可能想要按与源发出命令时相同的顺序在目标上执行。

设备命令类似于存储器命令，并可能具有指示数据传输大小的大小字段或位。可能的大小例如可以包括 1 至 8 字节、16、32 或 64 字节，和/或一整个缓存线（cacheline）的数据。不考虑数据传输大小，命令有效载荷的大小可能是一整个缓存线，根据需要用 1 和/或 0 来填充，从数据传输出去到缓存线中。

设备命令还可以具有掩码位，用于支持部分读和/或写。例如对于读命令，此掩码可以指示读哪些字节位置，以及读返回中的哪些字节有效。对于写命令，字节掩码可以指示有效载荷中的哪些字节有效。

配置命令，例如读和写，可以被认为是一种特殊类别的设备命令，其目标可以是设备和/或寄存器，例如 PCI 配置空间中的位置。此外，由于这些命令的特性，PMI 308 可以选择每次只有一个发出的配置命令。

所有命令类型可以共享一个公共的基础格式。例如，命令迁徙可以包含在迁徙有效载荷中编码为 64 个命令位和 8 个校验位的 72 位命令。为了使存储器读请求的延迟最短，读命令格式可以在迁徙的前一半中具有所有的关键信息，从而允许在 XMI 332 接收到整个迁徙之前向存储器分派读请求。

命令中的提前指示符，例如一个位，可以指示迁徙的前一半是否包含足以发起存储器读请求的信息，或者 XMI 332 是否必须积累整个迁徙中的信息。这个位可以仅为存储器读命令而设置；否则，XMI 332 可能不会正确地配置命令、设备命令或存储器写命令发起存储器读请求。然而，不是所有存储器读命令都可以将所述提前位设置为有效。例如，任何在迁徙的后一半具有特定调度信息（例如流标识或优先级）的存储器读命令都可使提前位无效，以迫使 XMI 332 在分派存储器读请求之前考虑此信息。

多个命令目的地位可以指示命令类型：存储器命令、设备命令或配置

命令。

在一个实施例中，多个事务（标签）位与读/写位一起构成了每个命令的独有 ID。独有 ID 不会被重新用于随后的命令，直到 PMI 308 已经接收到对先前任何具有相关联事务 ID 的该类型命令的响应，或者 PMI 308 已经确定先前任何具有该事务 ID 的该读/写（rd/wr）类型的命令都已经超时为止。

取消位是一个修饰符，其指示了一个存储器读命令是否要取消先前的存储器读命令。

在存储器命令的情况下，多个偏移量位标识了从存储器读和写的地址位所指定的存储器位置起的偏移量。在设备命令的情况下，与地址位相耦合的这个偏移量可以指定目标位置（例如指向存储器映射的设备寄存器）。

如上所述，可定义流 ID/优先级来对存储器读命令所属于的流、以及存储器读命令优先级进行编码，以使得 XMI 332 能对不同的存储器请求应用不同的调度策略。例如，这个字段可以指示存储器命令是否属于等时流（isochronous stream）并因此可能需要特定的服务时间以符合等时数据传递期限时间。

掩码位可以是指示在设备和配置命令的部分读和写中哪些字节有效的位向量字节掩码。

大小位可以指示设备和配置命令的数据传输大小。这种命令可能的数据传输大小是：1 至 8 字节（与掩码位相结合）、16 字节、32 字节、64 字节等等，以及整个缓存线的大小。

利用分布于迁徙后一半的校验位，对 72 位命令使用纠错码（ECC），从而可以保护每个命令的完整性。然而，在接收到了整个命令迁徙并且完成了 ECC 错误校验之前，XMI 332 可能希望将提前位看作提示（hint），并将任何对于该命令而提前分派的存储器读请求看作是推测性的（speculative）。因此，XMI 332 可能想要在这种推测性提前存储器读请求被分派之前就确保具有足够的资源来处理之。例如，XMI 332 必须确保对提前读请求的分派不会导致存储器控制器读请求队列的溢出。

对于写命令字段，在此实施例中为了简化编码和译码，除了偏移量和代码字段之外，这些字段可以与读命令的相同。

图 13 示出了存储器命令格式 1300 的一个实施例。在此示例中，我们假设了一个其中每个迁徙有四次传输并且每次传输 18 位的物理层。也就是说，命令具有 72 位，在 4 次传输中传递。进一步假设每个传输（传输 0、1、2、3）对应于跨越互连的一个位单元时间（bit cell time）。例如，在第一次传输期间（传输 0），可以传输较低地址位和读/写命令。在传输 1 期间，可以传输较高地址位和提前读指示符。在传输 2 和 3 期间，可以传输指示以下内容的位：命令目的地、从地址起的偏移量、事务 ID、校验、掩码、流 ID、大小、取消命令、优先级等等。

图 14 示出了设备命令格式的一个实施例 1400，其中我们假设了一个其中每个迁徙有四次传输并且每次传输 18 位的物理层。这里，在传输 0 期间，可以传输较低地址位和读/写命令。在传输 1 期间，可以传输较高地址位和提前读指示符。在传输 2 和 3 期间，可以传输指示以下内容的信息：命令目的地、从地址起的偏移量、事务 ID、校验位、掩码位、传输大小等等。

图 15 示出了配置命令格式 1500 的一个实施例，其中，我们假设了一个其中每个迁徙有四次传输并且每次传输 18 位的物理层。在传输 0 期间，可以传输较低配置地址位和读/写命令。在传输 1 期间，可以发送较高配置地址位和提前读指示符。在传输 2 和 3 期间，可以传输指示以下内容的信息：命令目的地、从地址起的偏移量、事务 ID、校验位、掩码位等等。

如上所述，链路层（LL）既具有传出格式又具有传入格式。图 16 示出了传出链路层格式 1600 的一个实施例。对于基础传出链路层格式，例如可以每个迁徙具有 8 位。这个基础格式可以被可选地扩展，例如再扩展 8 位成为每个迁徙总共 16 位。图 16 示出了每个迁徙 16 位的格式。LL 信号 0 用于在传输 0:3（四次传输：0、1、2 和 3）期间所发送的通信信息（信息）位和多个校验位。（多个）信息位可以指示迁徙是不是非空闲的，即该迁徙是包含命令或数据，还是空闲。LL 信号 1 用于传送头部、尾

部和校验位。尾部位指示分组的末端，而头部位指示迁徙是否包含命令。

头部和尾部位可以对其它状态进行编码，所述其它状态例如是：迁徙有效载荷是数据并且不是分组中最后的数据有效载荷；迁徙有效载荷是数据有效载荷并且是分组最后的迁徙；迁徙有效载荷是写命令；以及迁徙有效载荷是读命令并且是分组最后的迁徙。所述多个校验位包括在链路层格式的 16 个位上计算出来的 CRC。多个扩展模式位在 LL 信号 2 和 3 上发送。

扩展模式位还可以位于链路层中，并可被用于具有锁步（lockstep）存储器端口的数据迁徙。为了支持具有锁步存储器端口的 XMI（332）中的错误校验，扩展模式位可以对与发送给另一存储器端口的半个缓存线的错误相关的信息进行编码。

传入链路层可以具有各种格式。例如，在一个实施例中，可以有四种不同的传入链路层格式：读返回头部、写确认、状态和配置。在下面描述的实施例中，这四种链路层格式可以一起共享多个位。图 17 示出了公共传入链路层格式位 1700 的一个实施例。

信息位指示迁徙有效载荷是不是非空闲的，即包含数据还是空闲。多个校验位包括在链路层格式的 16 个位上计算出来的 CRC。

类型位可以指示四种不同的传入链路层格式中的一种，例如：读返回头部；写确认；状态；以及配置。

图 18 示出了读返回头部格式 1800 的一个实施例。这里，多个标签位可以按例如小端顺序（little endian order）将事务 ID 编码。控制位可以指示读返回具有数据，并且第一个数据有效载荷迁徙开始于从当前迁徙的开始处起给定偏移量个迁徙的地方。如果所述偏移量为 0，则第一个数据有效载荷在当前迁徙中开始。控制位还可以指示读返回头部是 nack（否定性确认的缩写），在这种情况下没有相关联的数据。读返回 nack 通知 PMI 308，告知其 XMI 332 因为接收到读取消命令而取消了原来的存储器读命令，并且从而 PMI 308 不应期望该读命令的读返回数据。如果 XMI 332 取消了读命令，则它仅发送一个读返回 nack。从而，如果 XMI 332 丢弃了读取消命令而未取消读命令，它就不发送读返回 nack。

图 19 示出了写确认格式 1900 的一个实施例。在此示例中，除了缺少

控制位之外，此格式与上面讨论的读返回头部的相同。

图 20 示出了状态格式的一个实施例 2000。存储器端口 320 可以将状态格式用于三种功能：将传出流控制信息返回给处理器；传递异步信号；以及指示链路层对于一个迁徙是空闲的。

5 如果停止位有效，则处理器不应向 XMI 332 发送任何写命令，直到该处理器随后接收到具有链路状态格式并且停止位无效的迁徙为止。当 XMI 332 希望对进入的写命令进行流控制，例如当 XMI 332 写缓冲器超出了给定阈值时，就置位（assert）停止位。如果停止位无效，则处理器可以向 XMI 332 发送写命令。

10 多个信号位可以包括信号代码。XMI 332 通过适当设置信号代码，就可以与读返回数据并行而异步地发信号以通知处理器的某些动作。

从以上实施例的示例中，本领域技术人员将会认识到，可以类似地对配置格式进行编码，以向传入链路层提供传递配置信息的功能。

15 足够详细地描述了这些实施例，以使得本领域技术人员能够实施本发明，应该理解到，可以利用其它实施例，并且可以进行逻辑、机械、电和其它方面的改变，而不会脱离本发明的范围。应该理解到，上面所描述的体系结构和功能可以具有其它实施例。任何格式、命令等都可以具有不同的位分配和不同的位定义。例如，读返回头部格式可以具有不同的位分配和不同的位定义，并且顺序可以是大规模的而不是小端的。

20 此外，尽管为了清楚起见，在本说明书中对于传入和传出方向而在实施例中描述了本发明的某些方面，但是应该理解到，它们也可以同时应用于传入和传出两者。例如，可以与迁徙中特定传输相关联的链路层数据可以与各自的传输在时间上错开，而这个传输可以是传入和/或传出方向上的。例如，PMI 308 可以在信息之前（一个偏移量）发出一个头部，该头部通过链路层而发送到 XMI 332。

25 机器可读介质被理解为包括用于以机器（例如计算机）可读的形式存储或传输信息的任何机构。例如，机器可读介质包括只读存储器（ROM）；随机访问存储器（RAM）；磁盘存储介质；光存储介质；闪存存储器设备；电、光、声或其它形式的传播信号（例如载波、红外信

号、数字信号等)；等等。

本领域技术人员应该理解到，使用计算机的实现可能需要计算机在作为装置而执行方法或任务之前，在上电（power up）或复位时适当地自我配置。这通常被称为初始化，并且可以包括但不限于设置芯片时序、确定所附接的设备、配置诸如存储器之类的子系统、检查访问速度、检查互连，等等。应该理解的是，在为完整功能操作作准备时，计算机系统可能进行许多检查、设置许多参数，等等。例如，在初始化时，计算机可能确定所附接的存储器并改变设置和参数，以确定最优接口速度、时序设置等等。此外，计算机在操作期间也可能进行检查，以确保合适的操作，并在必要时改变设置，等等。

从而，描述了通过存储器互连中的读发起优化而优化存储器读操作的方法和装置。

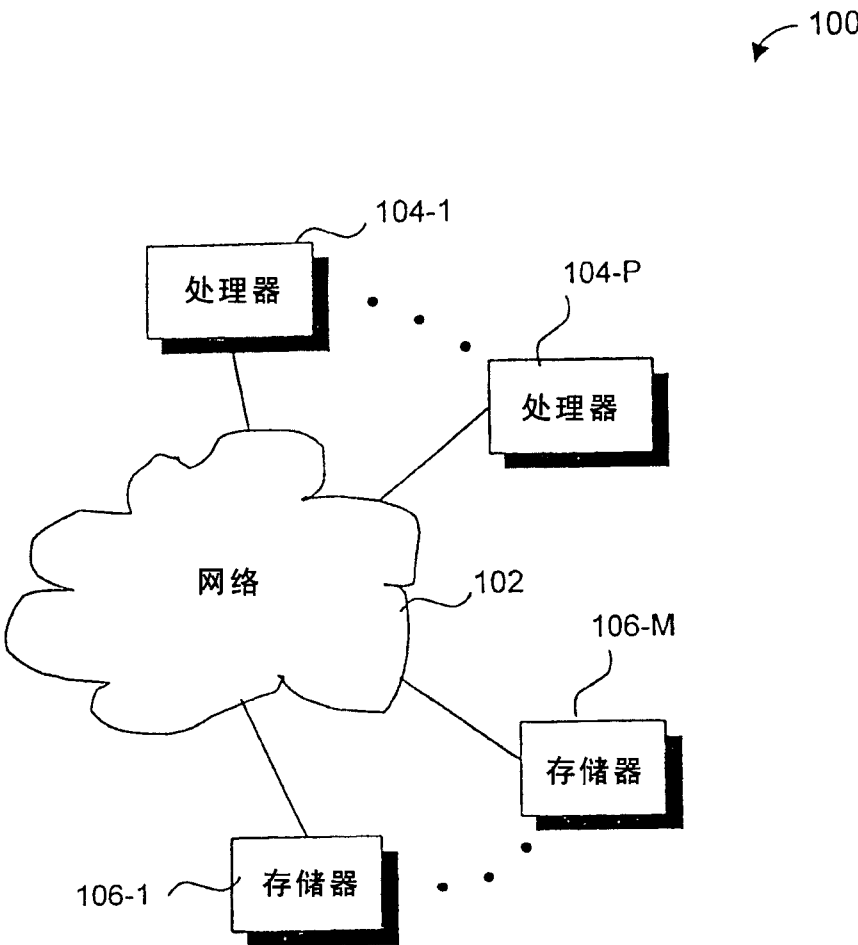


图1

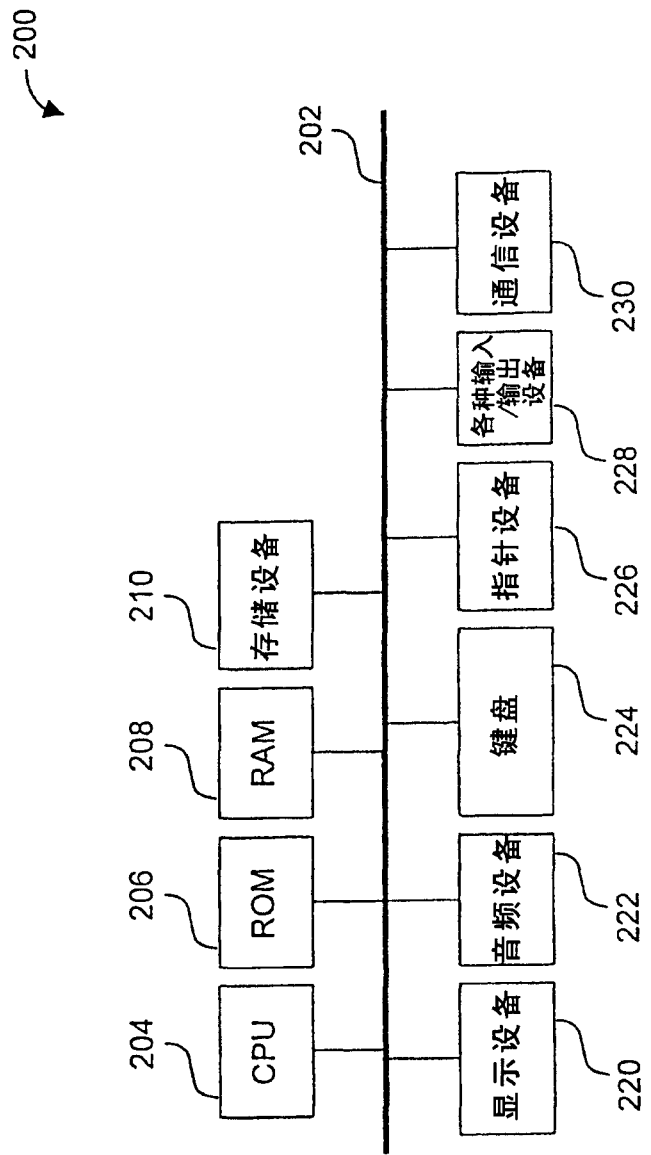


图2

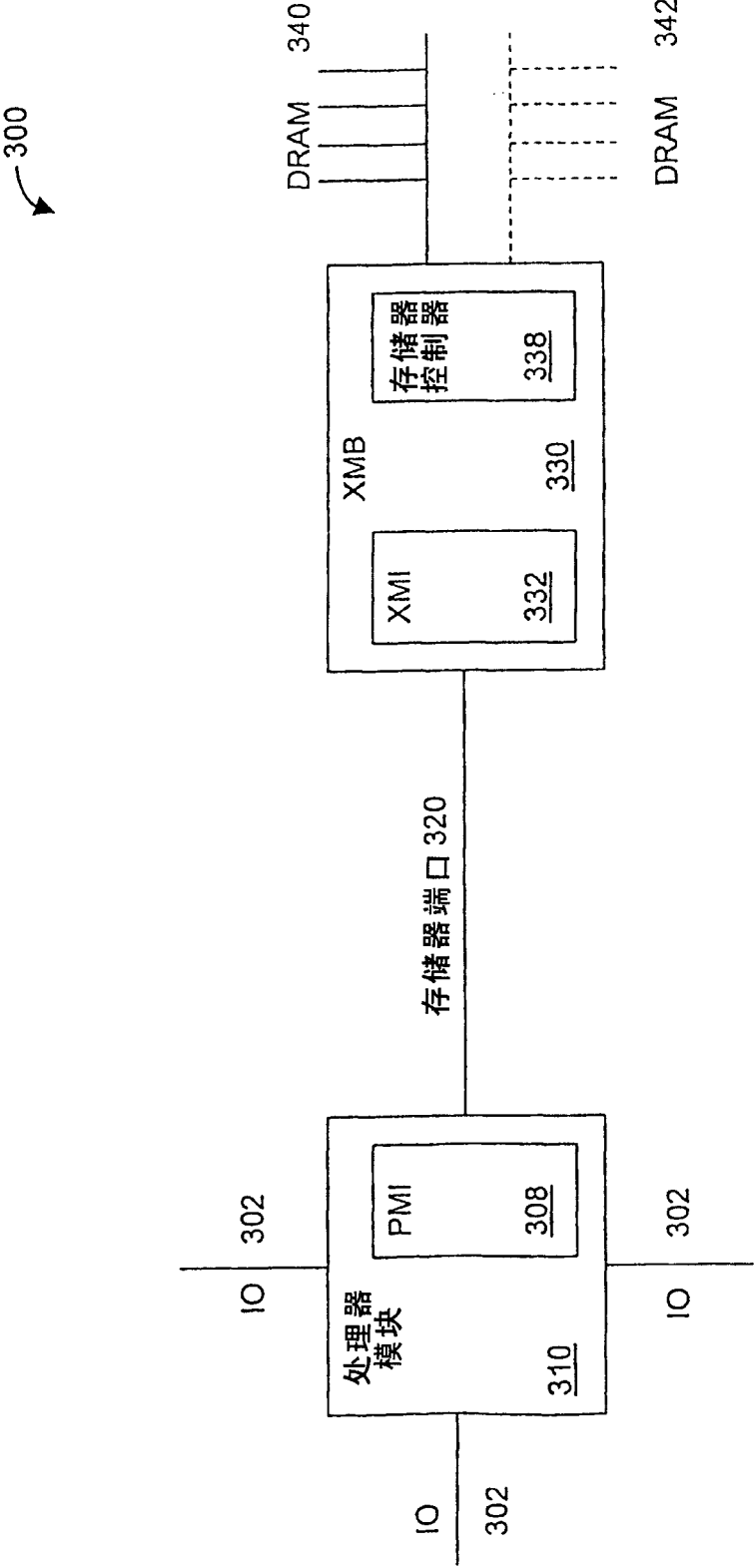


图3

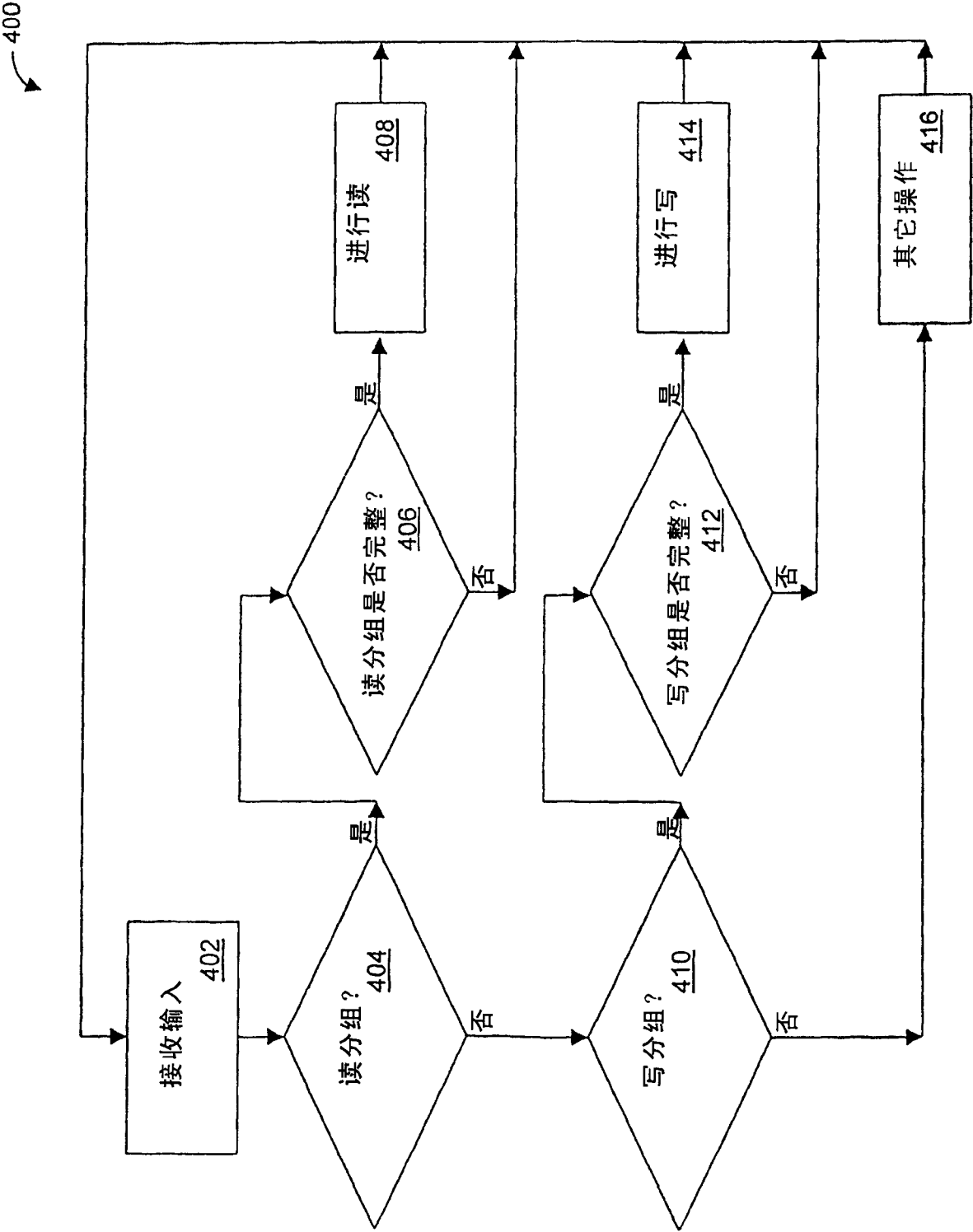


图4A

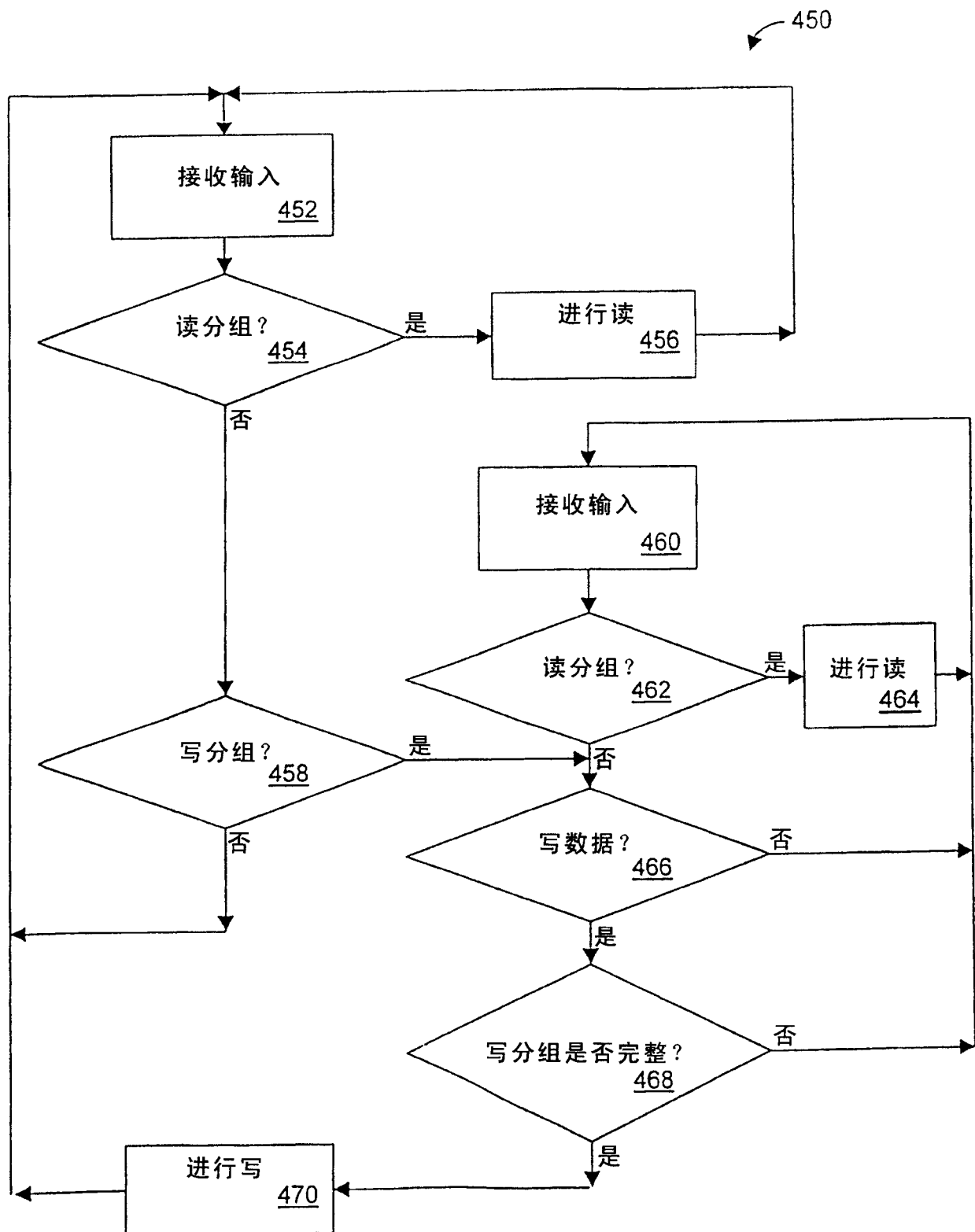


图4B

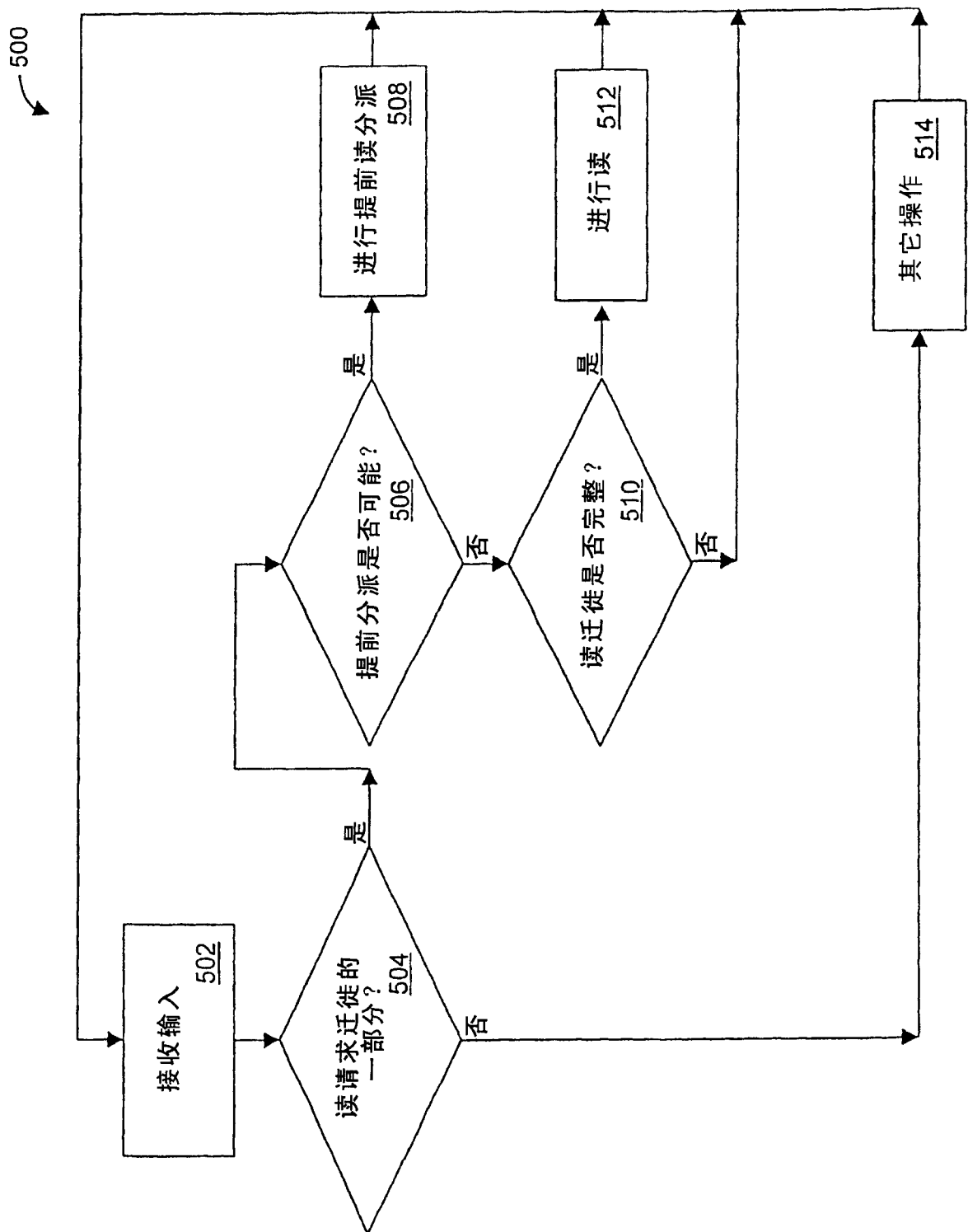


图5

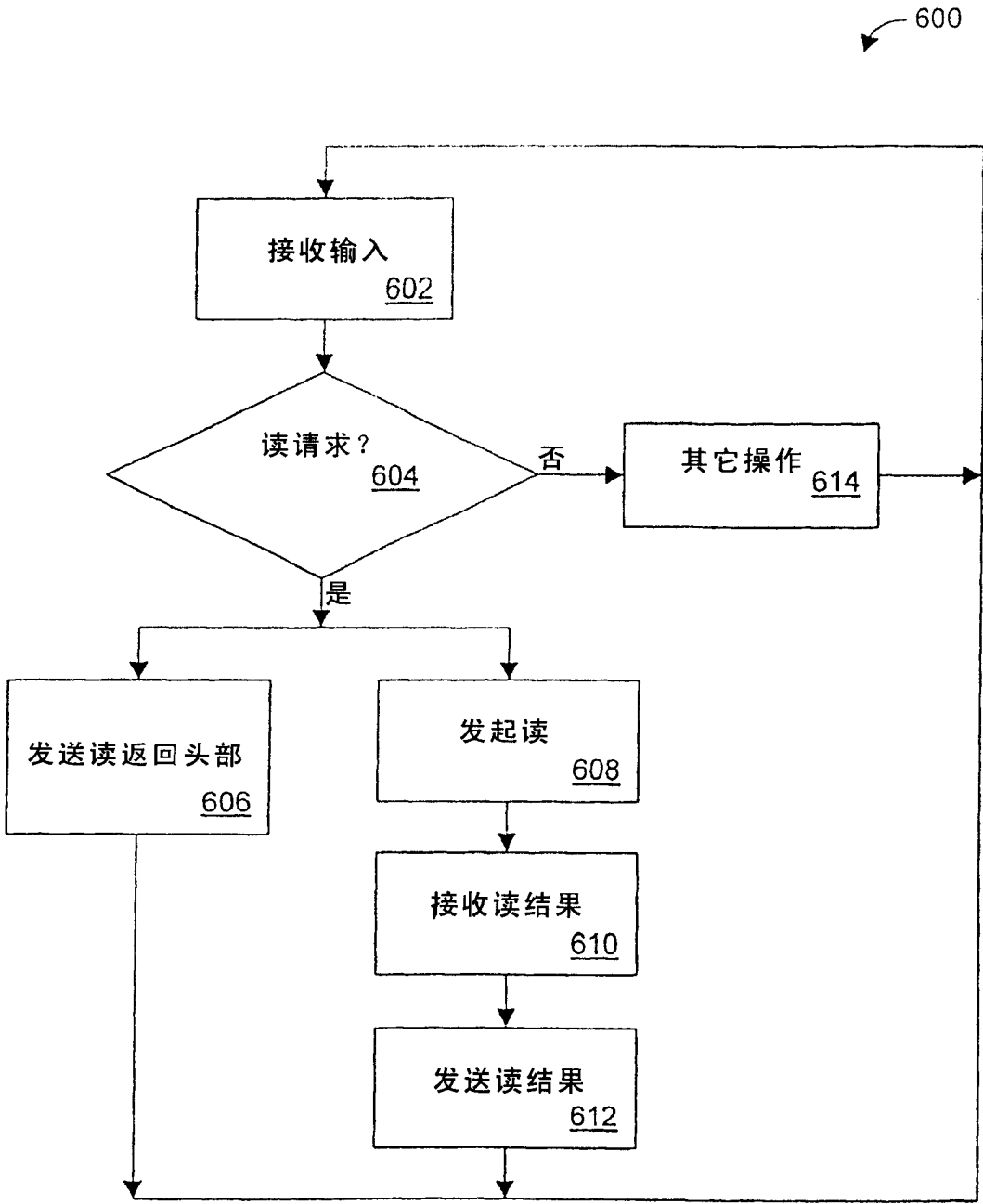


图6A

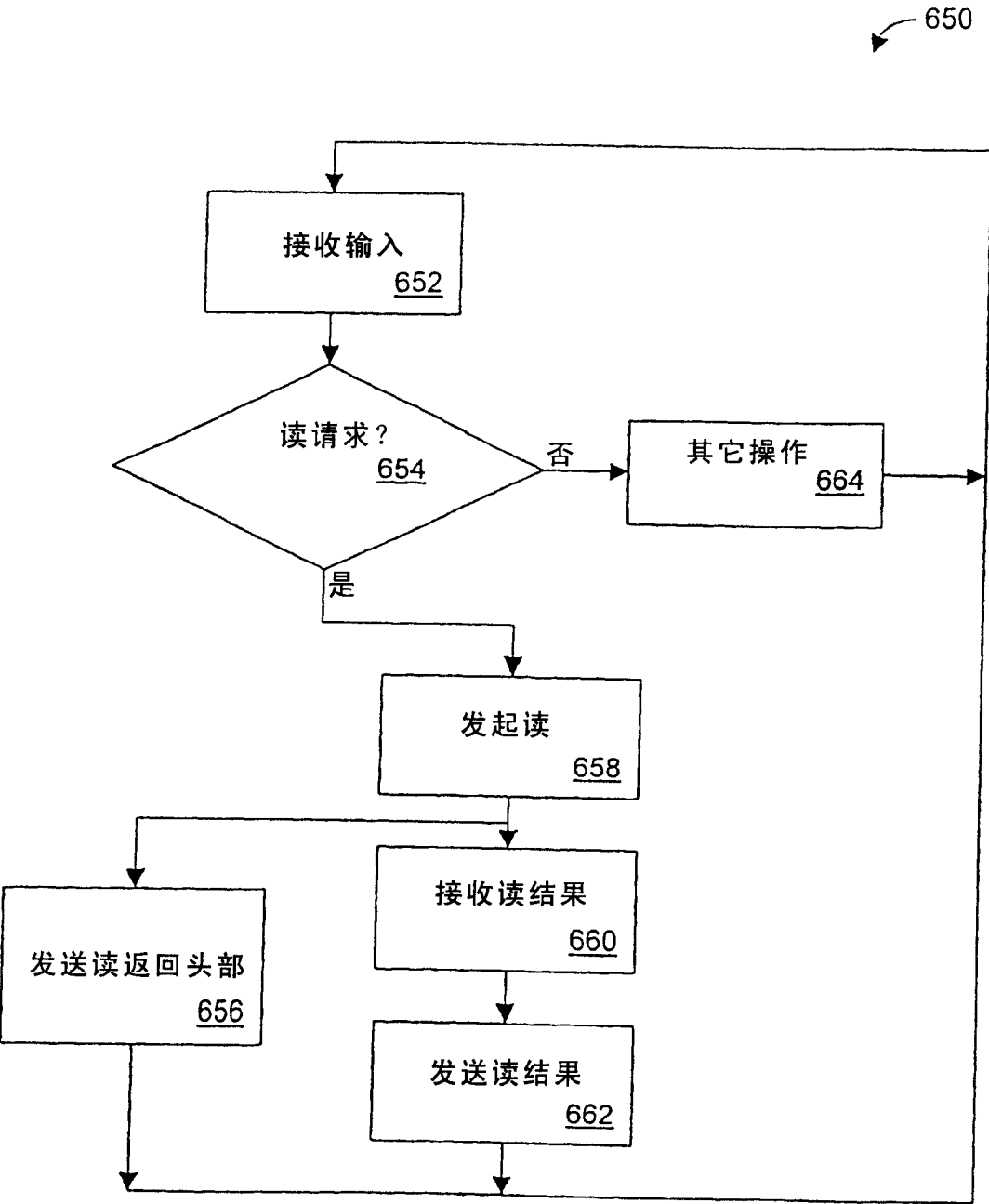


图6B

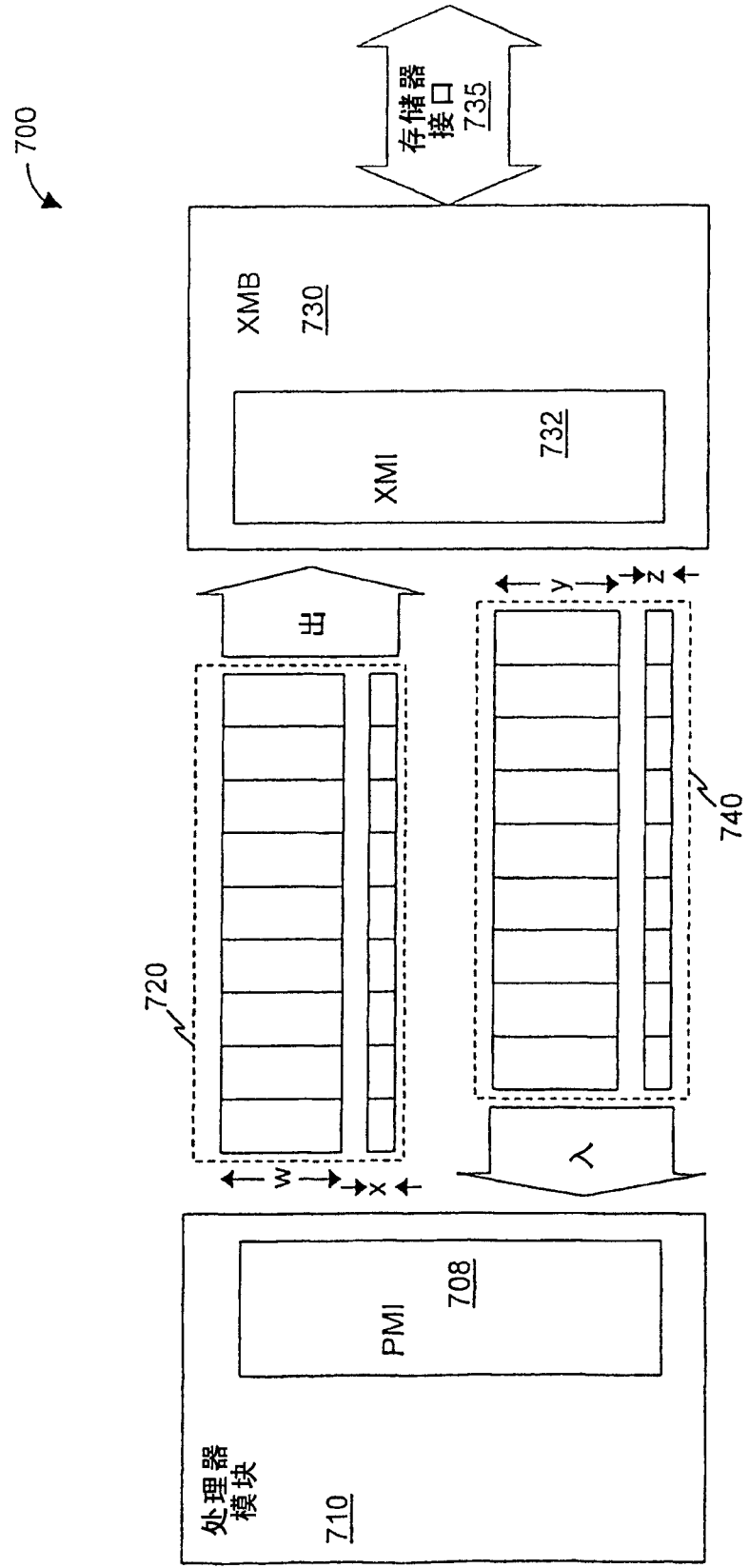


图7

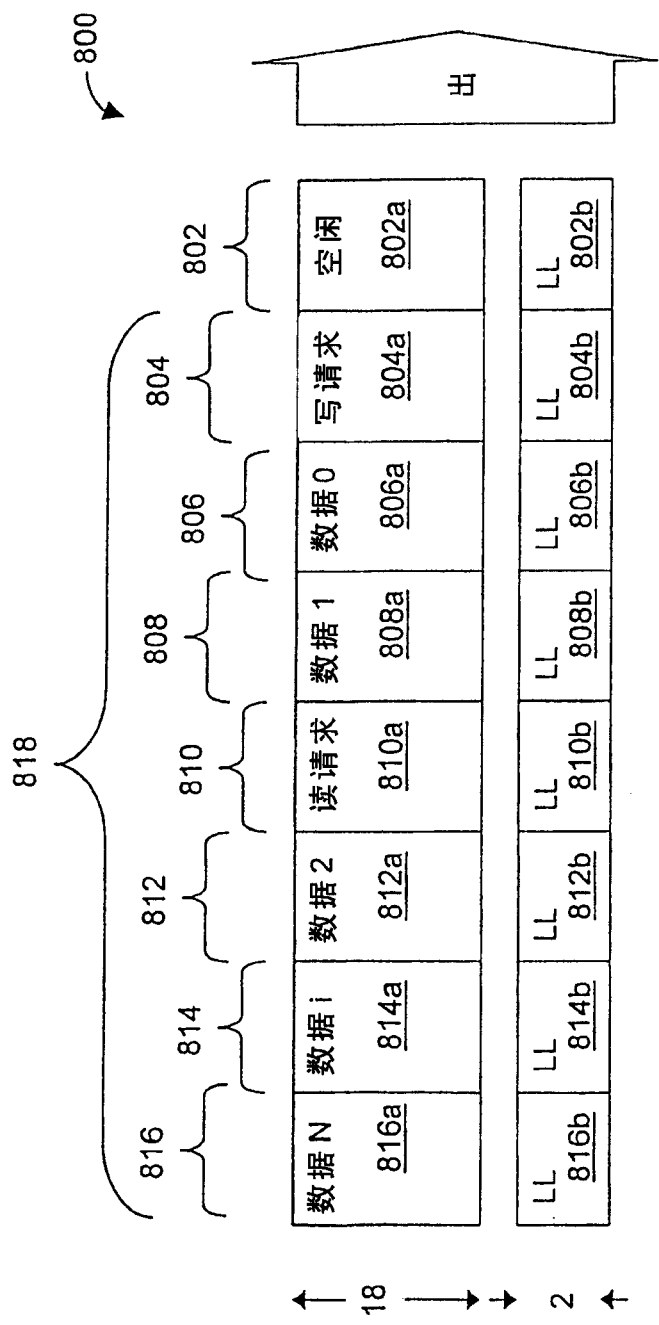


图8

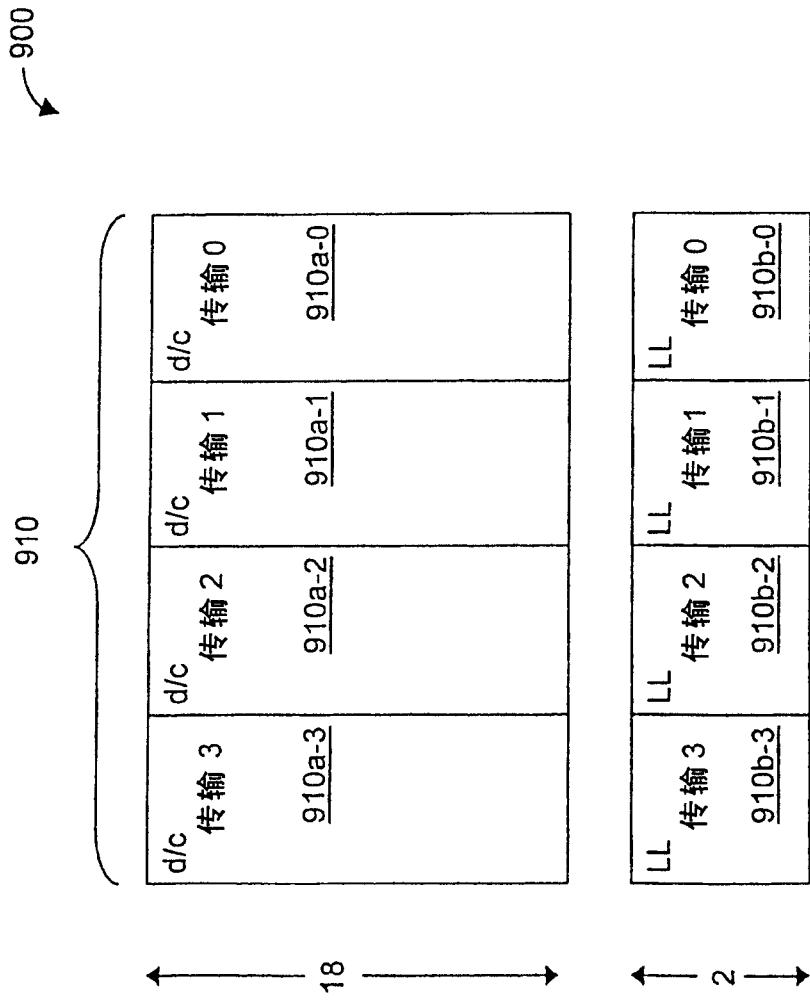


图9

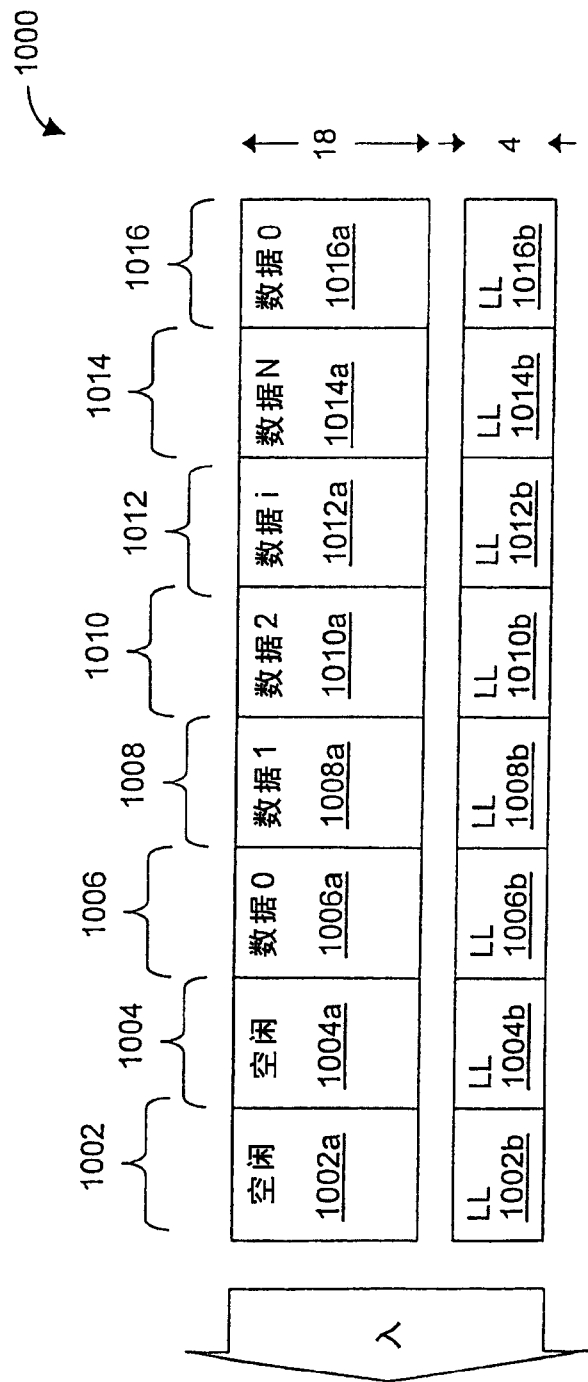


图10

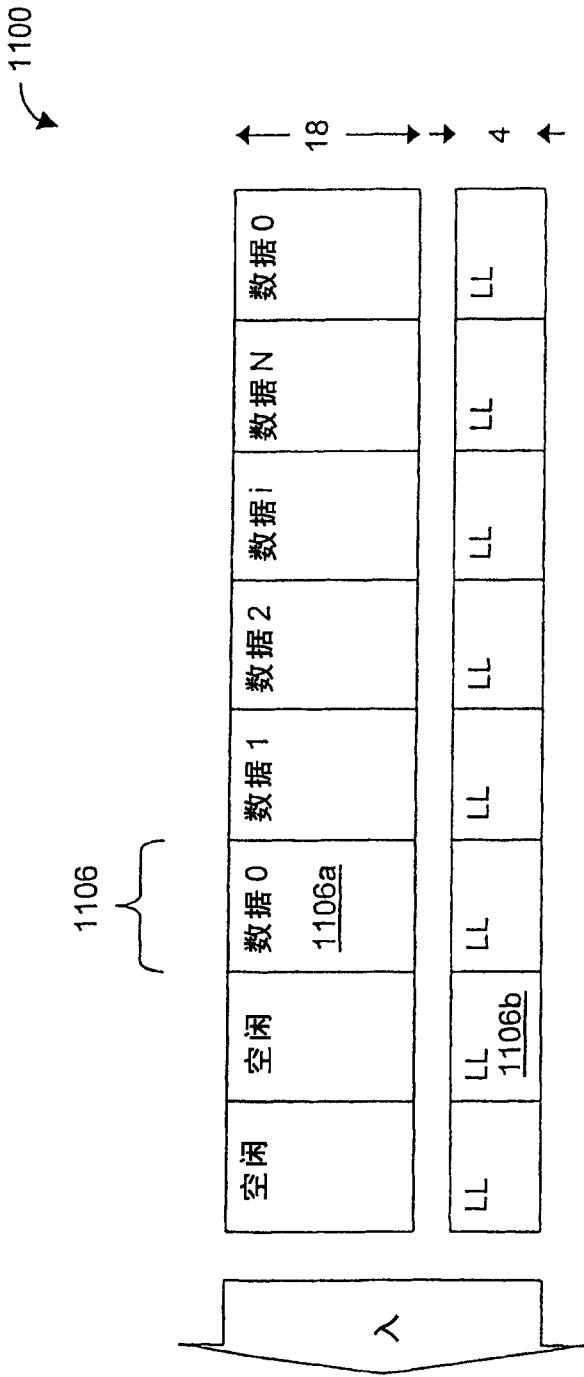


图11

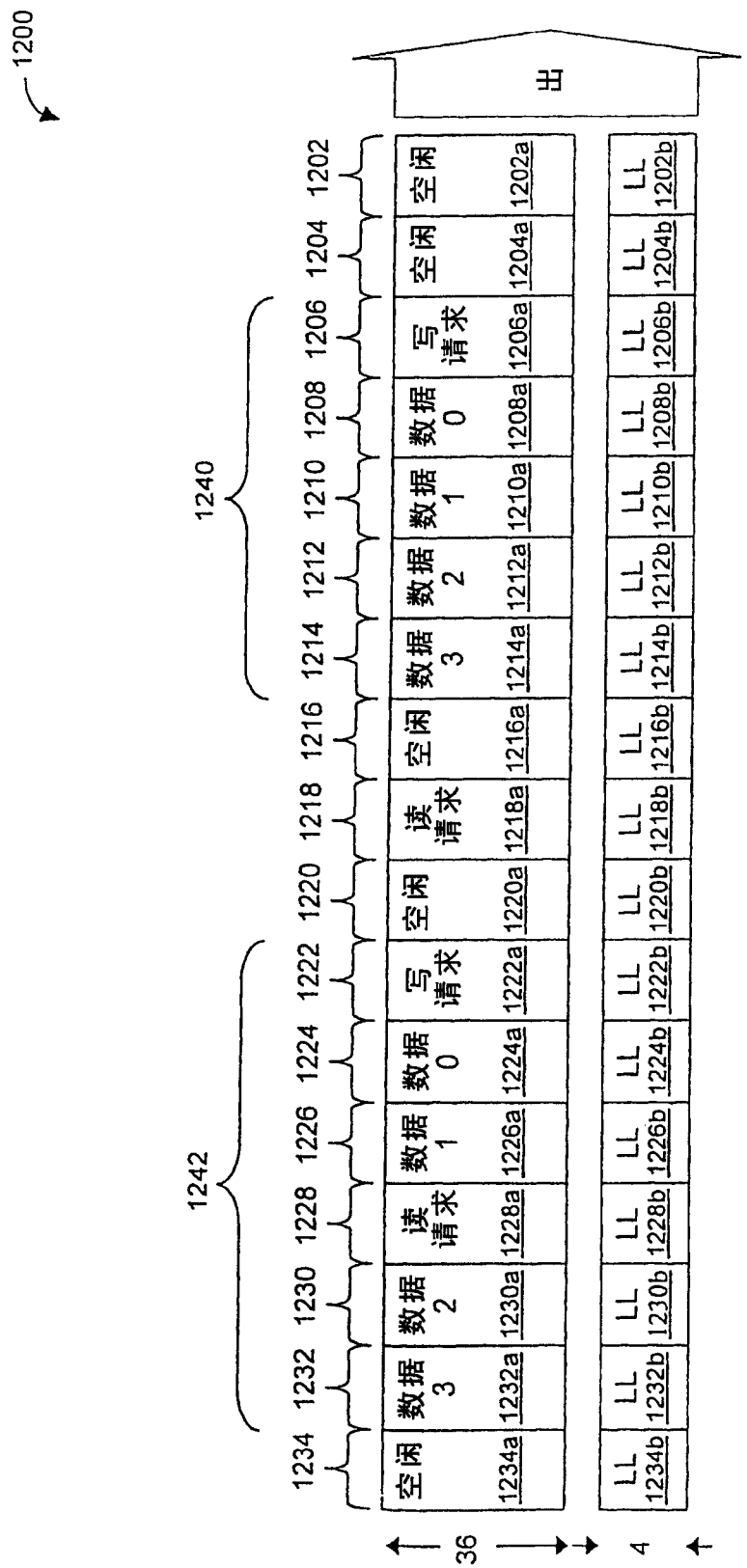


图12

1300

↓ 传输	位 17:0
0	较低地址位和读/写命令
1	较高地址位和提前读指示符
2	命令目的地、从地址起的偏移量、事务ID、校验位、掩码位、流ID、大小位、取消命令、优先级等等
3	

图13

1400
↙

↓ 传输	位 17:0
0	较低地址位和读/写命令
1	较高地址位和提前读指示符
2	命令目的地、从地址起的偏移量、事务ID、校验位、掩码位、大小位等等
3	

图14

1500

↓ 传输	位17:0
0	较低配置地址位和读/写命令
1	较高配置地址位和提前读指示符
2	命令目的地、从地址起的偏移量、事务ID、校验位、掩码位等等
3	

图15

1600

LL 信号	传输 0:3
0	信息和校验位
1	头部、尾部和校验位
2	扩展模式位
3	

图16

1700

LL 信号	传输 0:3
0	类型和校验位
1	
2	信息和校验位
3	

图17

1800

LL 信号	传输 0:3
0	标签、控制和校验位
1	
2	标签、信息和校验位
3	

图18

1900

LL 信号	传输 0:3
0	标签和校验位
1	
2	标签、信息和校验位
3	

图19

2000

LL 信号	传输 0:3
0	信号和校验位
1	
2	信号、信息、停止和校验位
3	

图20